UNIVERSIDAD NACIONAL AUTÓNONA DE MÉXICO

FACULTAD DE INGENIERÍA

“ENLAZAR LOS PROGRAMAS DE ESTUDIO POR MEDIO DE UNA APLICACIÓN DE 4 ASIGNATURAS DEL MÓDULO DE BASE DE DATOS”

T E S I S

Q U E P A R A O P T E N E R E L T I T U L O D E:

INGENIERO EN COMPUTACIÓN

P R E S E N T A N:

ROBERTO CARLOS GAMA GÓMEZ
ARELI MONSERRAT PÉREZ JIJÓN
TANIA REYES LEÓN

DIRECTORA DE TESIS:

ING. GABRIELA BETZABÉ LIZÁRRAGA RAMÍREZ

CUIDAD UNIVERSITARIA ENERO 2010

AGRADECIMIENTOS

Agradecemos a la Ing. Gabriela Betzabé Lizárraga Ramírez por su apoyo a lo largo de nuestra formación académica y dirección de tesis, gracias por su tiempo y tolerancia.

A nuestros sinodales por su gran y pronto apoyo para la realización de este trabajo.

Al M.I. Luis César Vázquez Segovia y M.I. Adolfo Millán Nájera por su ayuda y comprensión incondicional.

A la guardería 44, por brindarnos la información necesaria para la realización de este trabajo: a la directora Perla del Rosario Montero García, la pedagoga Leticia Martínez Juárez, la psicóloga Ma. Del Socorro Ibarra García, La trabajadora Social Martha Lilia Gómez Pacheco y la enfermera Rita Ibarra Ramírez.

A la Facultad de Ingeniería por abrirnos las puertas y prepararnos para nuestra vida profesional

A la Universidad Nacional Autónoma de México por habernos dado la oportunidad de formar parte de ella, por enriquecernos de conocimientos, cultura y ética.

DEDICATORIAS

A mis padres por haberme dado una gran educación, por su ejemplo diario de superación y por el apoyo que me dan en los días en que el sol no sale del todo.

A mi hermano por ser mi maestro, mi compañero y mi amigo.

A mi familia por apoyarme directa o indirectamente en cada paso que doy en la vida.

A mis amigos por los días de estudio y convivencia que han dado como fruto a mi segunda familia.

A los profesores que lograron hacerme estudiar sin que fuera obligatorio y a los que después de terminar el curso se convirtieron en amigos.

A Dios y a la Virgen de Guadalupe por llenar mi vida de dicha y bendiciones.

ROBERTO CARLOS GAMA GÓMEZ.

A Dios por que me ha acompañado a lo largo de toda mi vida.

A mis padres Arturo y Amelia que amo con todo mi corazón, son mi razón de ser, la fuerza que me impulsa a salir adelante. Quiero que este trabajo lo sientan como suyo, el fruto de todos sus esfuerzos, sacrificios y desvelos.

A mi hermano por apoyarme y hacerme ver que el mejor estado en la vida es ser feliz, Te amo.

A Luis mi mejor amigo, mi confidente, mi apoyo en los momentos más difíciles, mi luz, mi felicidad, el amor de mi vida.

A mis Abuelitos Arturo y Lupe, Proto y Adela por todo el cariño y amor que me dieron desde pequeña.

Les dedico este trabajo con todo mi amor.

ARELI MONSERRAT PEREZ JIJON

A mi Nanis que Siempre está conmigo incondicionalmente,

A mi Tatis, Daría lo que fuera por ser al menos la mitad de lo que usted es!,

A mi Tío David a quien admiro y respeto mucho,

A mi Yadi, eres mi alma gemela,

A Betito, sin tí no hubiera sido posible nada de esto,

A mis hermanos, por su apoyo Gracias,

A Fermín, quién me acompaña en esta aventura,

A tío Sebastián, Gracias por sus enseñanzas,

A los profesores que contribuyeron en mi formación académica,

A toda mi familia, Gracias!, les dedico este presente, Los amo!

TANIA REYES LEÓN.

ÍNDICE

1. INTRODUCCIÓN:

El presente trabajo tiene como objetivo poner en práctica los conocimientos teóricos de las asignaturas del módulo de bases datos, así como filtrar las preguntas más relevantes de las prácticas de las asignatura, proporcionando así un material de apoyo a los alumnos que opten por elegir dicho módulo.

La problemática principal en la que nos basamos fue la siguiente:

- Las asignaturas del modulo de Bases de Datos no tienen un orden preestablecido de cursarlas, pero realmente están seriadas.
- Las asignaturas del modulo de Bases de Datos no tienen horas prácticas, a pesar de que la profesora de las asignaturas se ve en la necesidad de combinar en el semestre la teoría y la práctica, con versiones libres de software en su mayoría.

La intención de esta tesis, es apoyar a los alumnos que se inscriban en próximos semestres en la elaboración de sus proyectos propuestos en dichas asignaturas que cursen. Y a los que estén a punto de egresar.

Se espera que sea una tesis altamente consultada por su aspecto práctico.

Para el desarrollo de este trabajo se utilizaron las siguientes técnicas:

- En un 90% metodología CASE, porque es una manera muy práctica de enlazar la parte de la aplicación y la de Bases de Datos hasta lograr terminar un sistema, se emplearon 3 diagramas: Diagrama Entidad Relación, Diagrama Jerárquico Funcional y Diagrama de Flujo de Datos.

- En un 10 % metodología UML, aquí se emplearon 2 diagramas: Casos de uso, actividad, estados, clases.

Las siguientes materias se utilizaron para el desarrollo de esta tesis son:

- Bases de Datos, correspondiente al Capítulo 1
- Bases de Datos Avanzadas, correspondiente al Capítulo 2
- Depósitos de Datos, correspondiente al Capítulo 3
- Minería de datos, correspondiente al Capítulo 4

Capítulo 1
Bases de Datos Teoría y Práctica.

Introducción

Este capítulo tiene como objetivo mostrar los conceptos y principios en los que se fundamenta la teoría de bases de datos y ejemplificar, mediante una aplicación enfocada a una estancia infantil, la metodología Case (Computer Assisted Software Engineering) y RUP (Rational Unified Process), diseñando e implementado la Base de Datos que utilizaremos como fuente para los próximos capítulos y en los que se fundamenta la creación de una aplicación, la cual, en nuestro caso, será programada sólo en un 10% y sus datos serán ficticios.

Metodología case:

ESTRATEGIA: Etapa inicial en el desarrollo de un sistema a través de la cual se conocerá el objetivo del negocio, sus funciones y necesidades, todo con apoyo del usuario. El esquema en esta etapa se le denomina TOP-DOWN (general a lo particular)
ANÁLISIS: Verificar Diagramas. Descripción más detallada que garantice el conocimiento del mundo real.
DISEÑO: Determinar la Base de Datos, procedimientos y programas.
CONSTRUCCIÓN: Codifican y prueban los programas
DOCUMENTACIÓN: Manual de operación y técnicas que sirvan de apoyo para la correcta operación y aclaración de dudas.
TRANSICIÓN: Implantación del sistema, período inicial de soporte del mismo.
PRODUCCIÓN: El sistema se utiliza en mundo real. Plan para mantenimiento y mejoras a futuro.

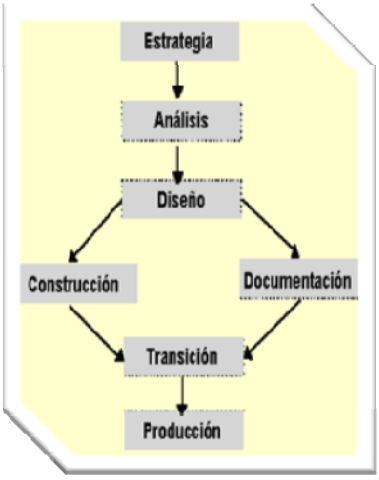


Fig.1.1.
Diagrama usado en metodología CASE.

En el siguiente diagrama se muestran los pasos correspondientes para obtener un sistema operacional, haciendo uso de la metodología case:

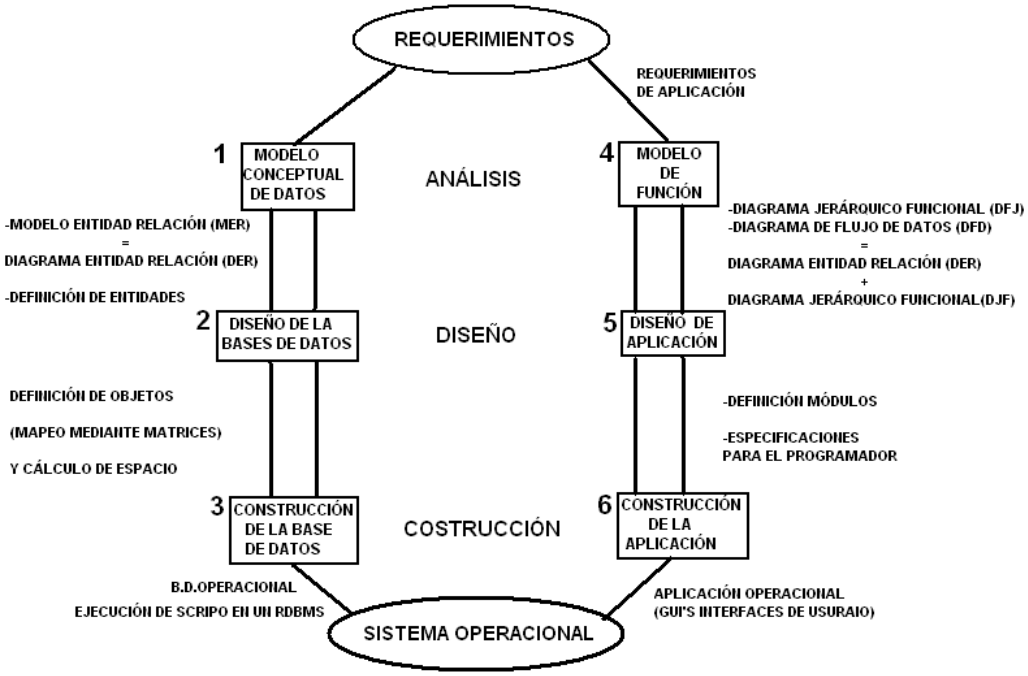


Fig.1.2. Diagrama que muestra el flujo de información en la metodología CASE.

Uso de Metodología CASE

El desarrollo es un enfoque top-down es decir que parte de lo general a lo particular, va transformando los requerimientos de información en un Sistema Operacional.

Requerimientos. Es importante tener todas las especificaciones, ya que de lo contrario la Base de Datos no cumplirá con las necesidades que se tienen en la aplicación final, éstos serán proporcionados por el usuario.

Ejemplo de Requerimientos para administrar la información de una Guardería (Estancia infantil).

- Personal Involucrado
    - (intendentes, cocineras, profesoras, directora, doctora, psicólogas, enfermera, trabajadora social, mantenimiento, seguridad, practicantes, dietista, dentista)

- *Horario de alimentación*
    - hr. Desayuno
    - hr. Comida

- Se deberá anotar de 3 diferentes formas el consumo de alimento del pequeño separado en 4 partes en
    - *desayuno* (fruta, leche, alimento fuerte, pan) y en
    - *comida* (entrada, plato fuerte, fruta o postre , agua) CT= Comió Todo PR=Probó CM=Comió Mitad NC=No Comió de cada componente.

- Para entrada y salida de los pequeños, aparte de los
    - *padres o tutores* pueden incluirse en la credencial del menor, 2 adultos autorizados.

- Se contemplarán los *datos generales* de los Padres o Tutores y del menor.

- Por cada nivel no se evalúa solo hay un comentario final.
- Al ingresar el menor se realizan las entrevistas para formar su expediente,
    - Revisión médica con *antecedentes de enfermedades, peso al nacer, estatura al nacer, cartilla de vacunación.*

- Si el niñ@ enferma en la estancia se entrega una constancia para ausentarse determinado número de dias.

- Si el niñ@ *falta* debe ser justificado con receta medica y Si el pequeño necesita un medicamento para suministrarlo hay que indicar en recepción indicaciones de la dosis.

- Deberá involucrar Estados y Municipios de nuestro país. (identificando en donde se relacionarían) en donde involucren *domicilios**

- El pequeño puede *permanecer* hasta los 6 años

- La prestación la tienen las madres al 100% y alguna excepción padres, su horario de trabajo puede variar.

Desarrollo de la Base de Datos (Rama izquierda)

Cubriremos los siguientes elementos que se requieren en la creación de la Base de Datos:

1 ***Análisis*** de los requerimientos de información del usuario. Modelaremos (Esquematizaremos, Diagramaremos) los aspectos importantes de la información que el negocio necesita saber o tener y las relaciones entre dicha información. y se desarrolla un Modelo Entidad-Relación para expresar esos requerimientos.

2 ***Diseño*** de Base de Datos Se mapearán las entidades, del modelo Entidad Relación, en tablas, los atributos opcionales en columnas nulas, los no opcionales en obligatorias y los identificadores únicos en llaves primarias, finamente, las relaciones en llaves foráneas.

3 ***Construcción*** de la Base de Datos. Se crearán físicamente las tablas en la Base de Datos Relacional, implementándolas de acuerdo al diseño de Bases de Datos.

Desarrollo de la aplicación (Rama Derecha)

4 ***Análisis*** En base a las necesidades del cliente creamos 2 diagramas:

1. el *Diagrama Jerárquico Funcional*
2. el *Diagrama de Flujo de Datos*.

5 ***Diseño*** especificaciones del sistema para el programador de cada función definitiva del *Diagrama Jerárquico Funcional*.

6 ***Construcción*** Aquí se programan la interfaz gráfica de usuario (GUI'S).

1.1 Análisis de la información para la Base de Datos (1)

El modelo de datos entidad-relación Se basa en la percepción del mundo real, que consiste en un conjunto de objetos llamados entidades y las características de cada una, llamados atributos y las relaciones entre las entidades. Representa la estructura lógica general de la Base de Datos gráficamente.

Realizaremos los siguientes tres pasos:

1.- Identificar entidades
Mediante los siguientes pasos se puede llegar a identificar y modelar las entidades de las notas de entrevistas a los usuarios y de formatos, etc.

1. Examinar los sustantivos
2. Poner un nombre a cada entidad
3. ¿Existe información de interés para la compañía acerca de la entidad?
4. ¿Cada entidad es identificable de manera única? ¿Cuál o cuáles atributos sirven como Identificador único (UID)?

2.- Descripción de entidades
Escribir la descripción de la entidad.

3.- Establecer las relaciones entre entidades

Es importante resaltar que una entidad puede ser identificada a través de una o múltiples relaciones.

NORMALIZACIÓN

Es aquí donde aplicamos la *normalización* para el Modelo De Datos, entendiendo por normalización: *Validar su ubicación de los atributos de cada entidad y en caso de no encontrarse en el lugar correcto, reubicarlos*. Su objetivo es eliminar redundancia.
Se usarán las siguientes reglas de normalización.

1FN: Todos los atributos deben tener un sólo valor
2FN: Un atributo debe ser dependiente del identificador único completo
3FN: Ningún atributo no-UID puede ser dependiente de otro atributo no-UID.

Ejemplo:

En la siguiente tabla tenemos los atributos:

curp_profesora que viola la **1FN** pues un Infante puede tener diferentes profesoras (música, ingles), así que no tendrá un solo valor, también viola la **2FN** pues no depende del curp_infante.

nombre_profesora este atributo viola la **1FN**, la **2FN** por lo antes mencionado y también viola la **3FN** pues se trata de un atributo no-UID que depende de otro atributo no-UID curp_profesora.

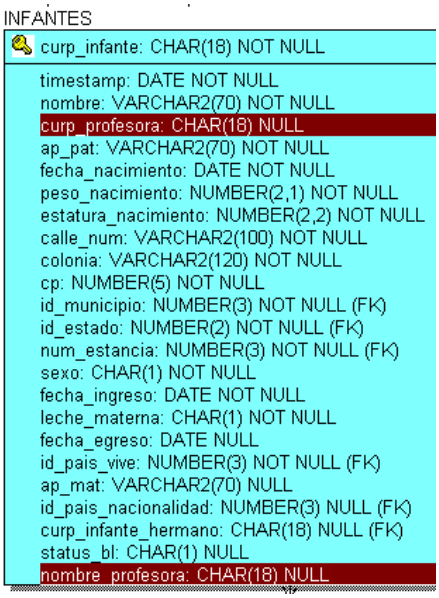


Fig.1.3. Tabla Infantes. Ejemplo de Violación

Dichas violaciones se solucionan colocando estos atributos en otra tabla, llamada empleados, como se muestra a continuación.

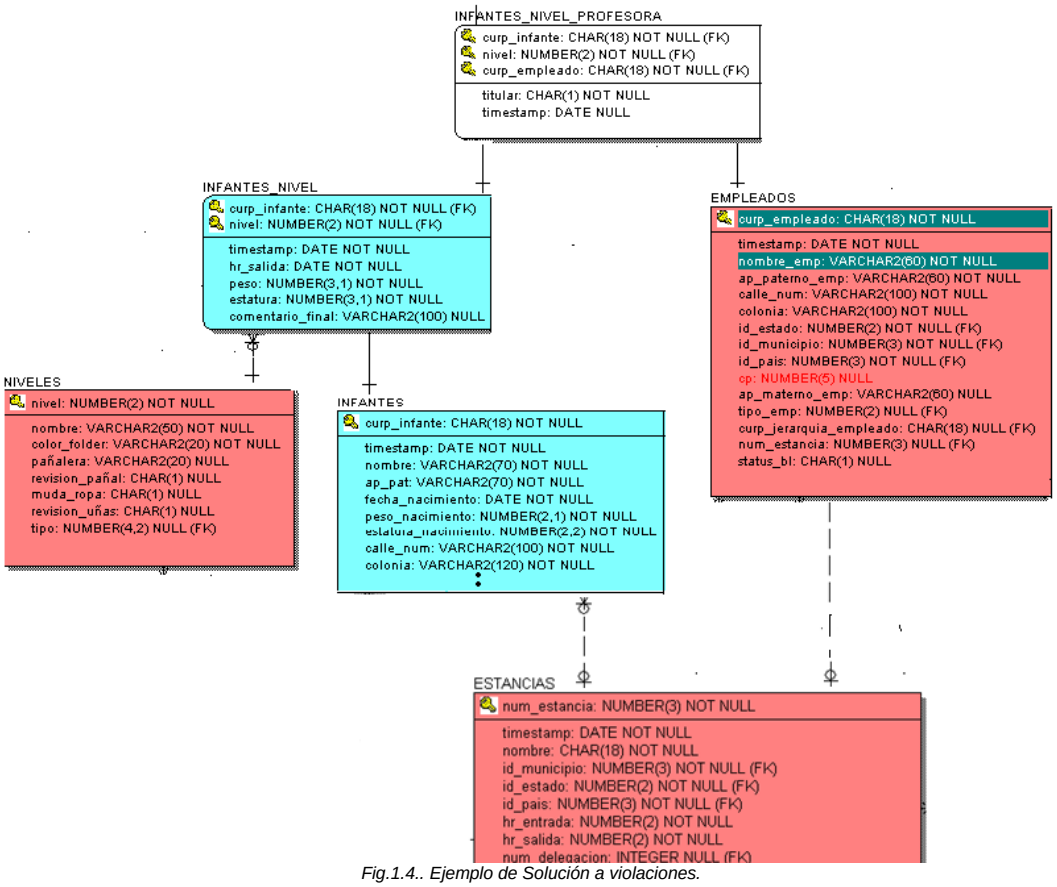


Fig.1.4.. Ejemplo de Solución a violaciones.

Un modelo de datos **Físico** entidad-relación normalizado se traslada automáticamente en un diseño de Base de Datos.

TIPOS DE RELACIONES.

Una **relación uno a uno (1 a 1 ó 1:1)** tiene un grado uno y solamente uno en ambas direcciones.

Ejemplo:

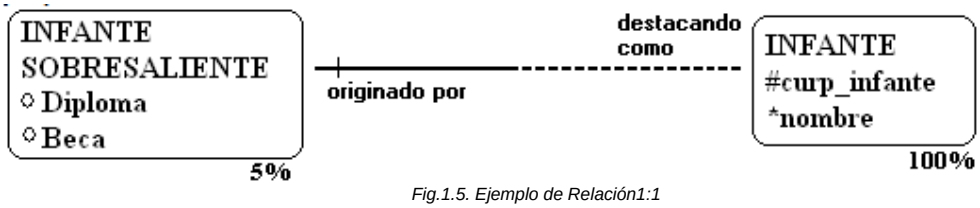


Fig.1.5. Ejemplo de Relación1:1

* El 5 % solo son sobresalientes, por tanto INFANTE tendría mucha recepción de NULOS para los atributos de INFANTE SOBRESALIENTE.
Cada INFANTE puede estar destacando como uno y sólo un INFANTE SOBRESALIENTE
Cada INFANTE debe ser originado por uno y sólo un INFANTE.

Notas * Las relaciones 1:1 Son muy escasas.
* Una relación 1:1 que es obligatoria en ambas direcciones es muy difícil de encontrar.
* Las entidades que tienen una relación 1:1 pueden ser en realidad la misma entidad.

Ejemplo:
Cada estancia tiene una y solo una directora:

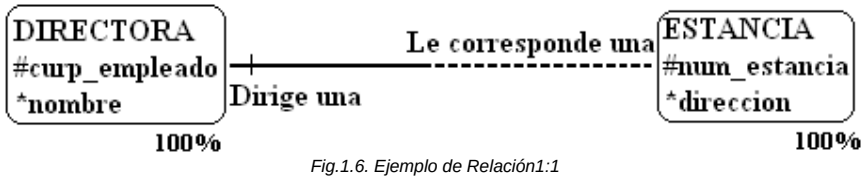
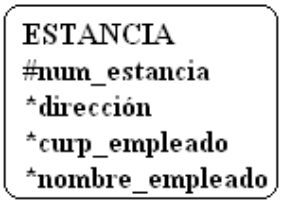


Fig.1.6. Ejemplo de Relación1:1

Y la tabla ESTANCIA quedaría así:

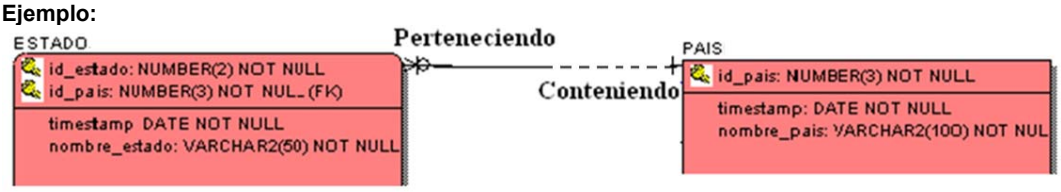


Donde el atributo **curp_empleado** es UNIQUE y NOT NULL

Entonces podemos ver que **curp_empleado** viola la 2FN, pues no depende de estancia, **nombre** violan la 2FN por la misma razón y también viola la 3FN pues depende del curp_empleado.

Pero es mejor tener una relación 1:1, sin embargo analizando el flujo de datos vemos que a través del tiempo son muchos directores y sólo tendríamos el actual, por lo que es mejor tener en EMPLEADOS, el director.

Relaciones M:1
Una relación muchos a uno (M a 1 ó M:1) tiene el grado de uno ó más en una dirección y el grado de uno y solamente uno en la otra dirección. Estas Relaciones son las más comunes.
Ejemplo:



Relaciones M:M
Donde originalmente se tiene:

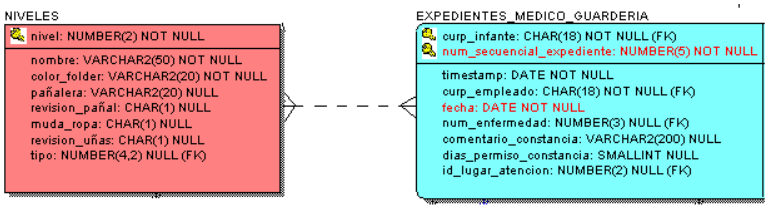


Fig.1.7. Ejemplo de Relación M:1.

Solución de **Relación M:M**

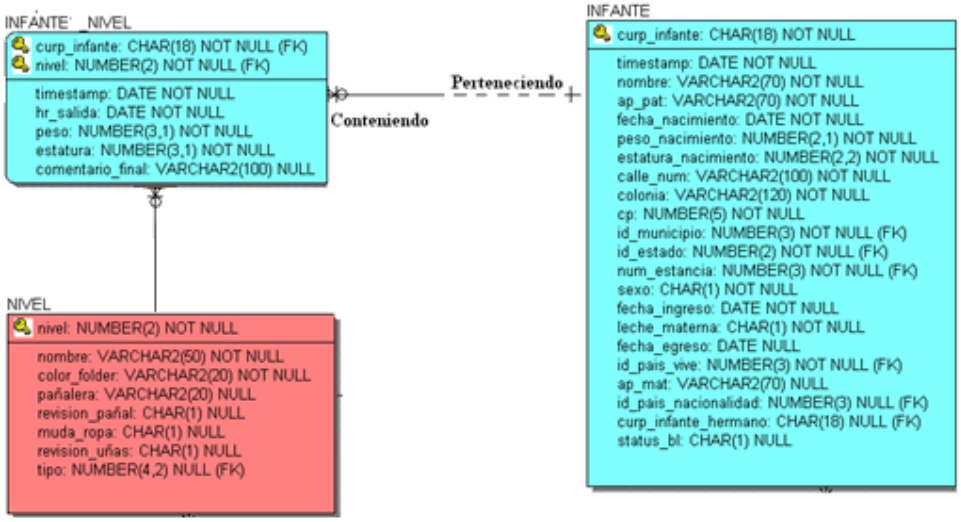


Fig.1.8. Ejemplo de Relación M:M.

Las Relaciones M:1 obligatorias de ambas direcciones son poco comunes y se valida su integridad referencial con constraints diferidos, es decir después del commit de los inserts de ambas tablas.

Dependencia por existencia: Cuando un entidad no existe, sin que exista previamente otra por ejemplo; *Expediente e Infante*; No existe un o varios registros a través del tiempo en un expediente sin el infante, por lo que Registro en Expediente depende de Infante.

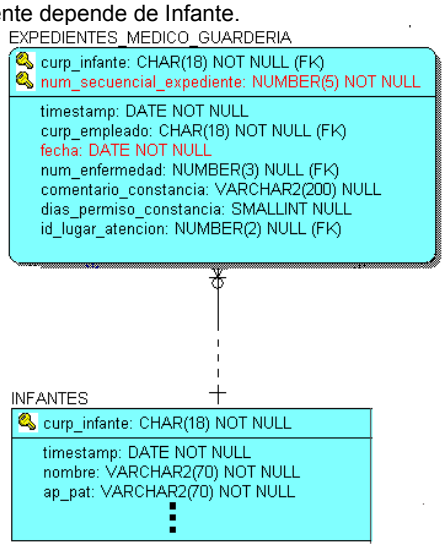


Fig.1.9. Ejemplo de Dependencia por existencia.

Relaciones recursivas: Se trata de las relaciones jerárquicas, sobre la misma entidad.
Ejemplo: En este caso un Empleado es Jefe de otros y en ocasiones Subordinado de otro Empleado

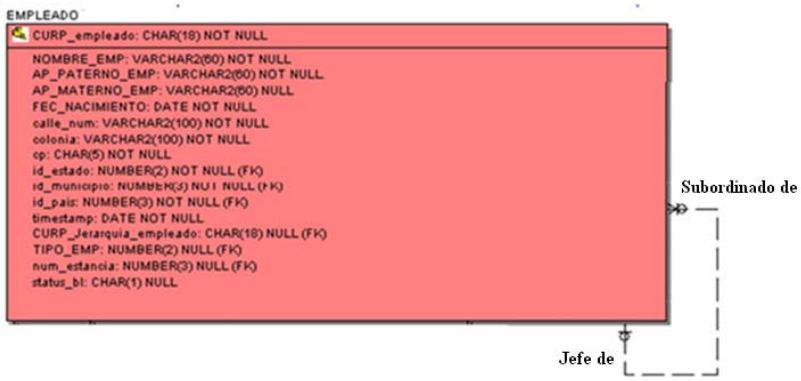


Fig.1.10. Ejemplo de Relaciones cursivas.

Modelado de relaciones complejas.
Aquellos identificadores únicos de una entidad que se forman por más de dos identificadores a través de las relaciones (es decir débil de 3).

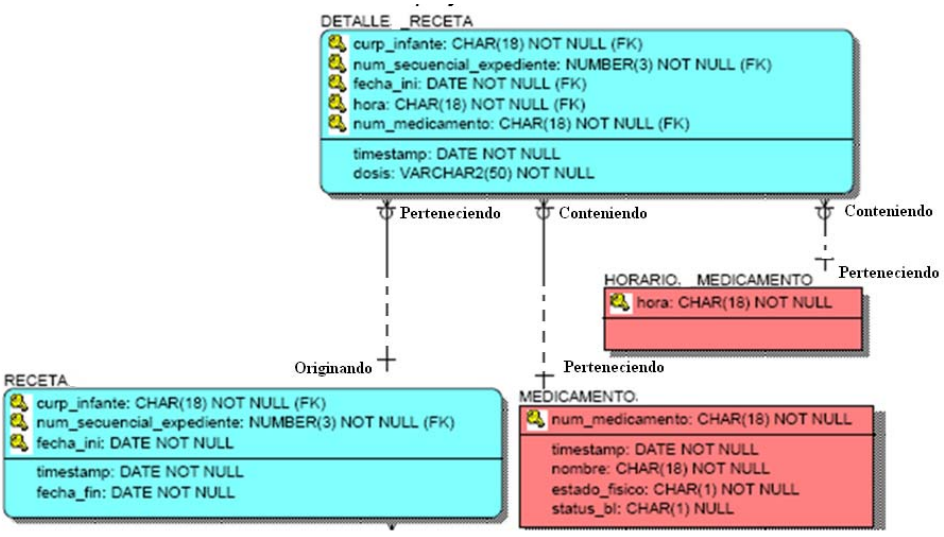


Fig.1.11. Ejemplo de Modelado de relaciones complejas.

En este caso, el infante puede requerir en diferentes fechas el mismo medicamento y en el mismo horario.

Relaciones que modelan roles.
Aquí las relaciones permiten que una sola instancia de entidad asuma múltiples roles.

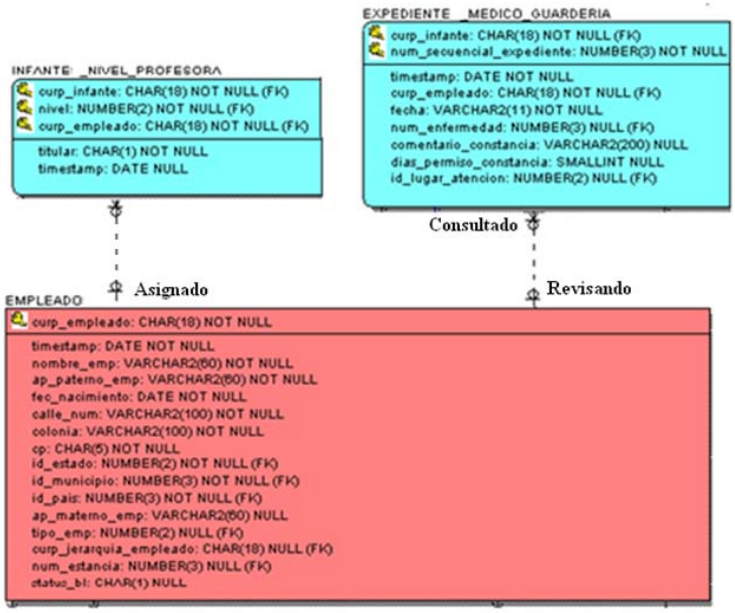


Fig.1.12. Ejemplo de Relaciones que modelan roles

En este ejemplo EMPLEADO tomará los roles de *empleado de la estancia* para ser el tipo de empleado profesora y/o *padre o tutor* para registrar alguna enfermedad de su hijo en la misma guardería.

Subtipos o subentidades y supertipos o superentidades.
Un *supertipo* es una entidad que tiene subtipos. Un supertipo puede ser separado en dos o más subtipos mutuamente excluyentes.
Los *subtipos* sirven para modelar exclusivamente tipos de entidad que tienen atributos o relaciones, y comparten atributos y relaciones generales.

Ejemplo:
Un nivel tiene dos tipos de subniveles: lactante y maternal. Para todos los niveles se guarda su nivel, nombre y color_folder asignado. Para los lactantes y maternales se guardan atributos particulares.

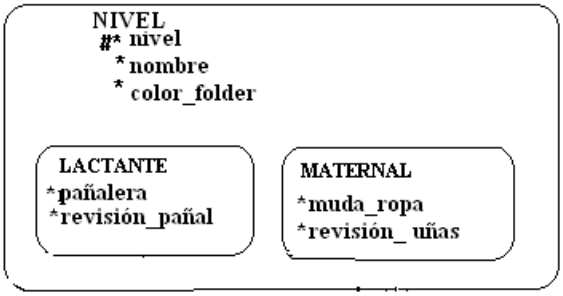


Fig.1.13. Ejemplo de subtipos y supertipos

*No olvidar que el supertipo tiene #.
NOTA: Siempre se utilizan subtipos otros cuando no se está seguro de que sean los únicos subtipos, sino pueden surgir más.

Relaciones excluyentes

Modelar dos o más relaciones mutuamente excluyentes de la misma entidad usando un *arco*, es decir hacer modelos de relaciones exclusivas.

Ejemplo: Se usa un arco para modelar dos relaciones excluyentes con CONSUMO_DESAYUNO dentro de los que se clasifican JUGOS_FRUTAS Y FRUTAS_YOGOURTH_POSTRE.

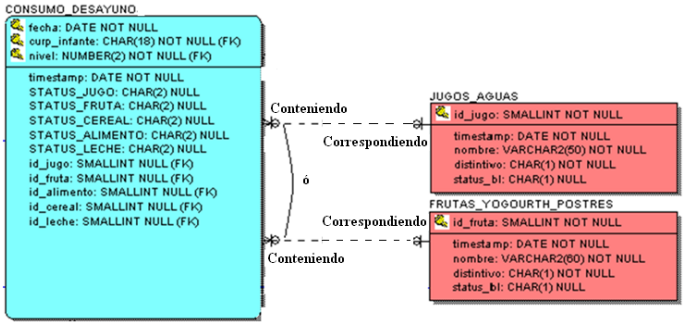


Fig.1.14. Ejemplo de Relaciones excluyentes

Cada CONSUMO_DESAYUNO debe tener a un y sólo un JUGO_AGUA o tener un y sólo un FRUTA_YOGOURTH_POSTRE.

Modelar datos de tiempo
Ejemplo:
Necesitamos mantener información acerca del detalle de receta y en qué **fecha** dosificó al menor. Cada detalle_receta contempla la dosis del medicamento prescrito.
Inicialmente fue modelada DETALLE_RECETA.
Pero la receta con medicamentos suministrados requieren llevar el registro histórico.

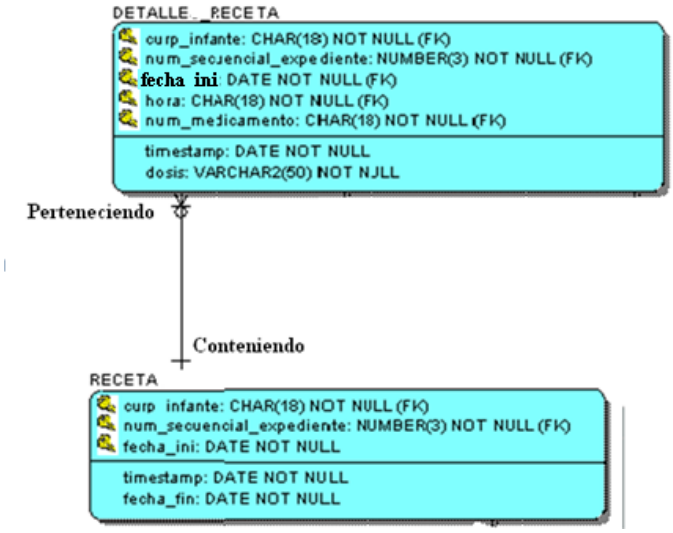


Fig.1.15. Ejemplo de Modelado de Datos de tiempo.

A continuación se muestra el Diagrama Jerárquico Funcional (DJF) que emerge del objetivo principal que es *Controlar la información de la estancia*, a partir de él, se observan las diferentes tareas que se estarán realizando para cumplir dicho objetivo, hasta este nivel solo se trata de una clasificación de las tareas o funciones. Bajando de nivel en nuestro DJF se encuentran las tareas más específicas que se llevarán a cabo para controlar la información, como son mantenimiento de catálogos, registros de los niños, mantener al día información, mostrarla, realizar respaldo, recuperación, y finalmente la administración de usuarios que tendrán acceso al sistema.

Diagrama Jerárquico Funcional:

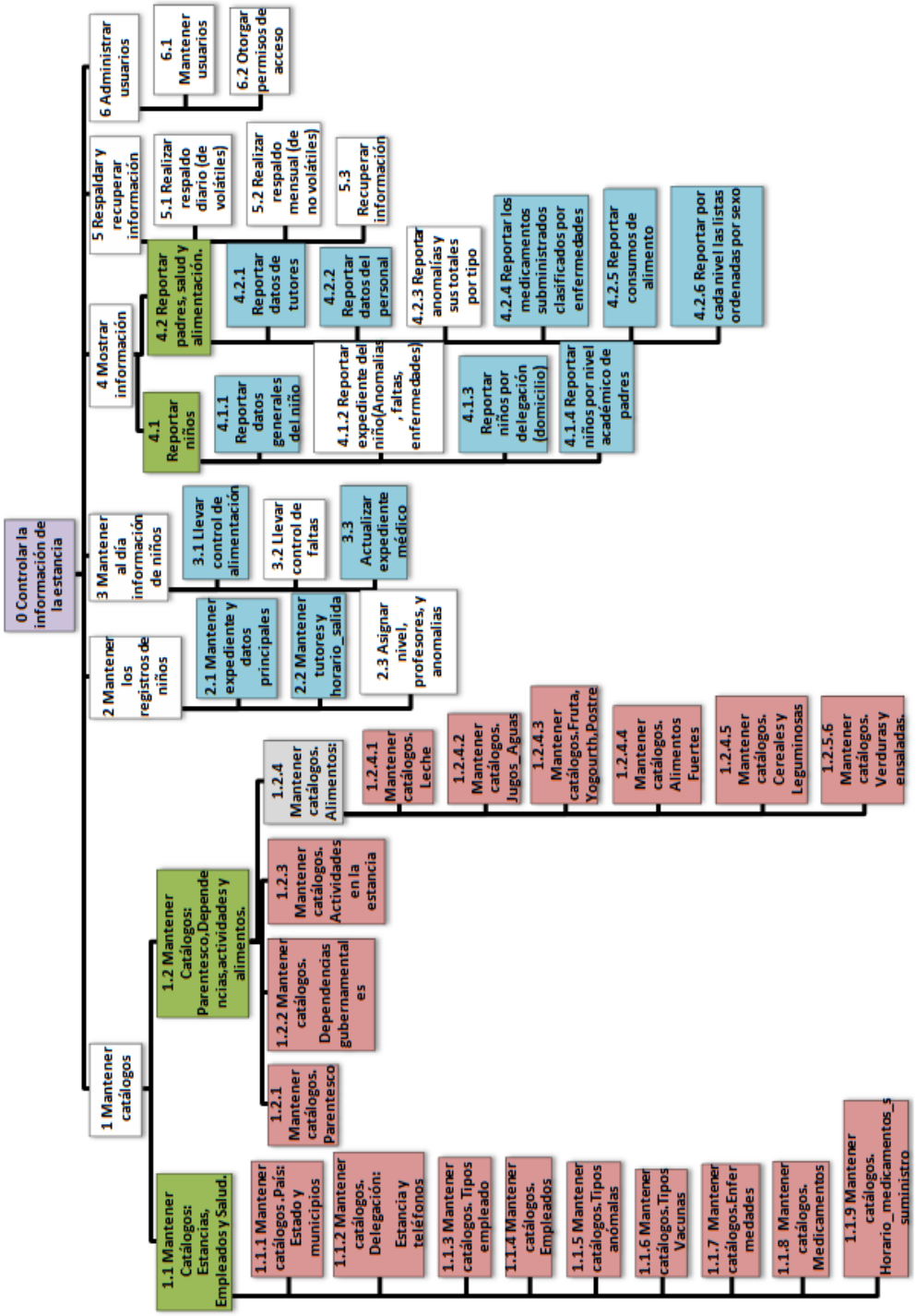


Fig.1.17. Diagrama Jerárquico funcional.

NOTA: Las funciones 2.3, 3.2, 4.1.2, 5.1, 5.2, 5.3, 6.1 y 6.2 No se involucran con la Base de Datos en esta tesis.

Diagrama de Flujo de Datos:

En este diagrama se muestra el flujo de información que habrá a lo largo de la tareas o funciones que se asignaron en nuestro DJF; es decir las flechas (flujos) indican si entra o sale información a las tablas de la Base de Datos, así como muestra la entidades externas, las cuales usan la información.

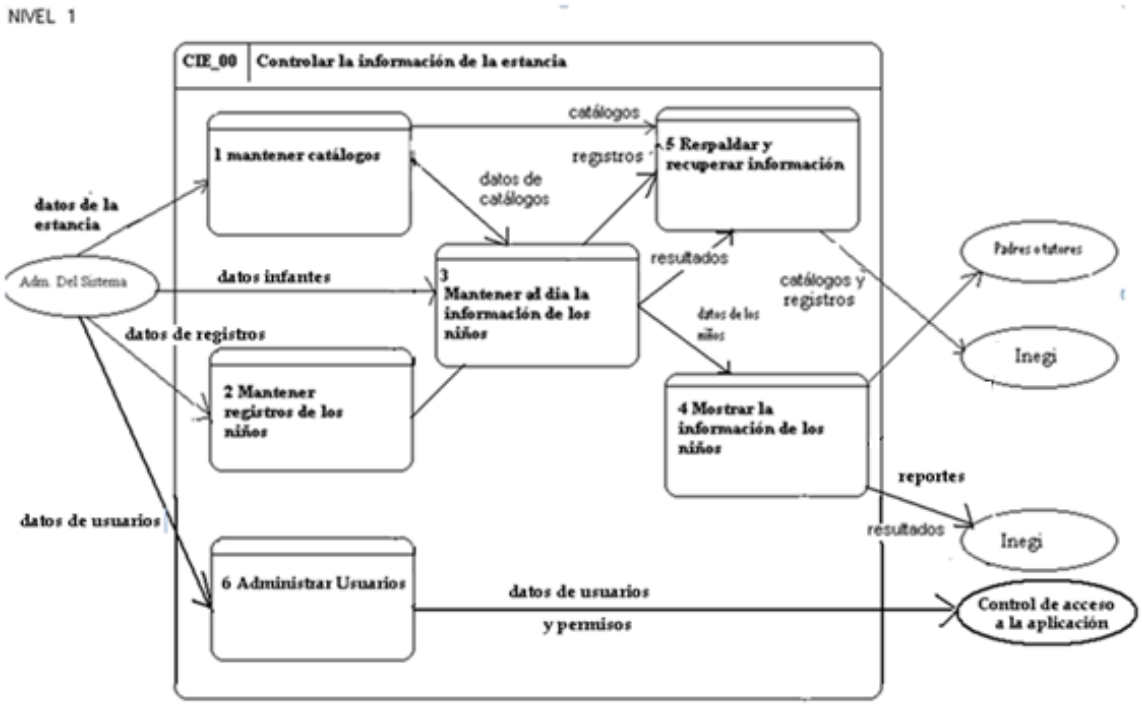


Fig.1.18. Diagrama de Flujo de Datos. Nivel1: Controlar la información de la estancia.

Observar que en los niveles siguientes, se muestran las funciones del DJF y las Entidades del DER.

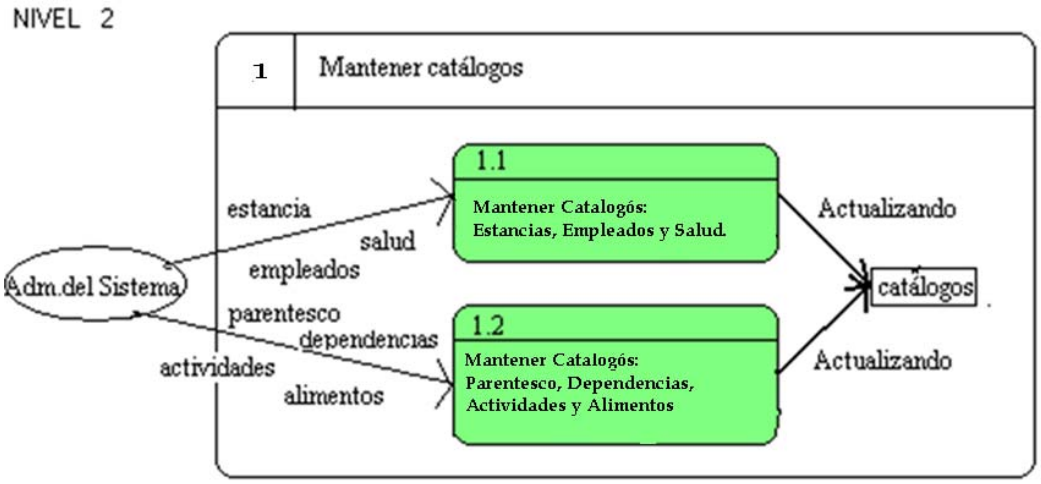


Fig.1.19. Diagrama de Flujo de Datos. Nivel2: Mantener catálogos

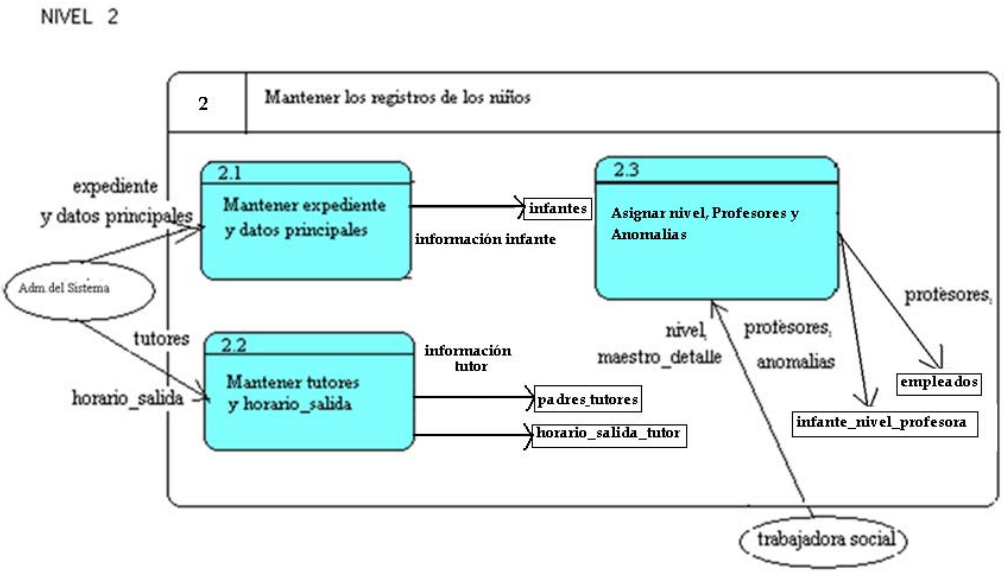


Fig.1.20. Diagrama de Flujo de Datos. Nivel2: Mantener registros de los niños.

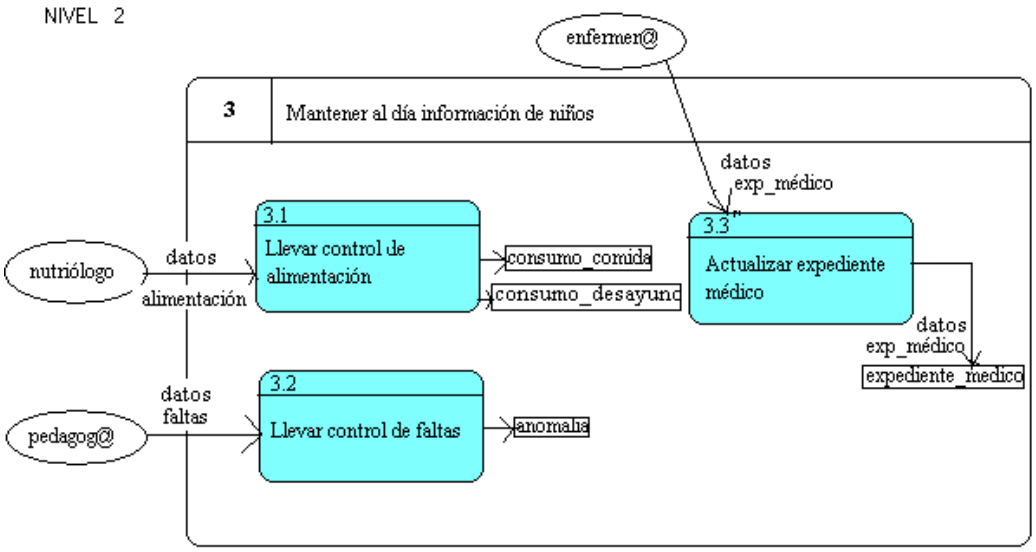


Fig.1.21. Diagrama de Flujo de Datos. Nivel2: Mantener al día información de los niños.

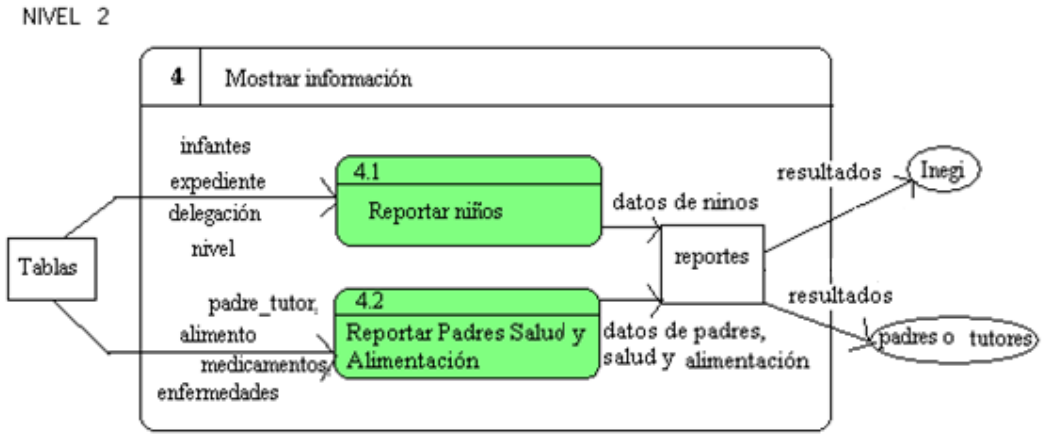


Fig.1.22. Diagrama de Flujo de Datos. Nivel2: Mostrar información

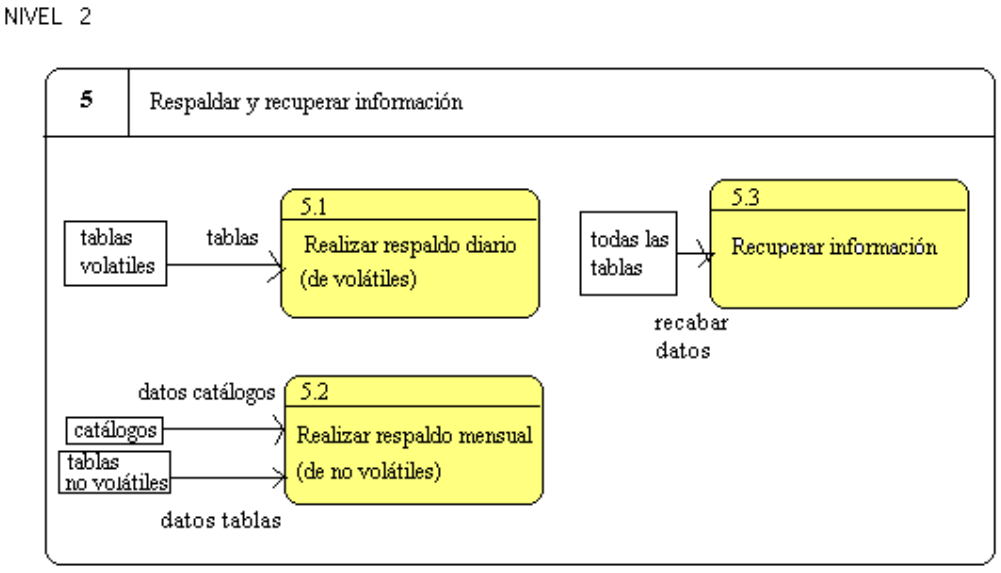


Fig.123.. Diagrama de Flujo de Datos. Nivel2: Respaldar y recuperar información.

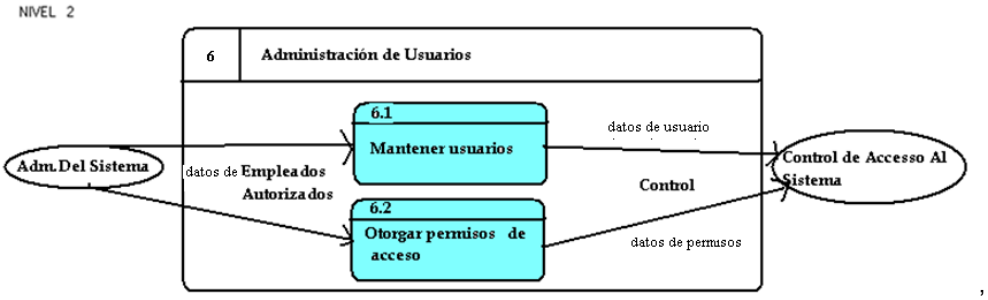


Fig.1.24. Diagrama de Flujo de Datos. Nivel2: Administración de usuarios.

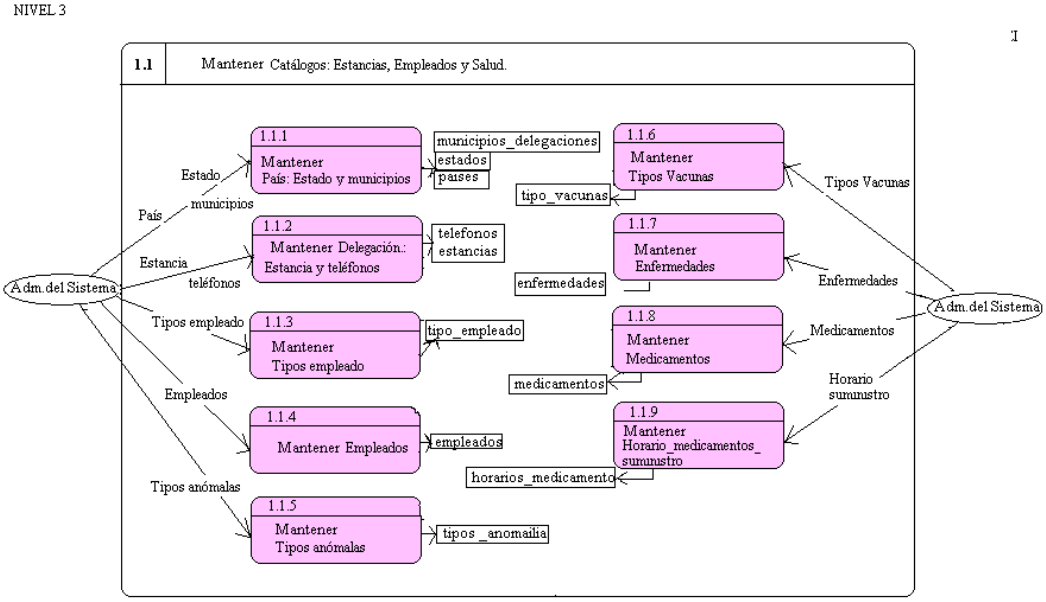


Fig.1.25. Diagrama de Flujo de Datos. Nivel3: Mantener catálogos

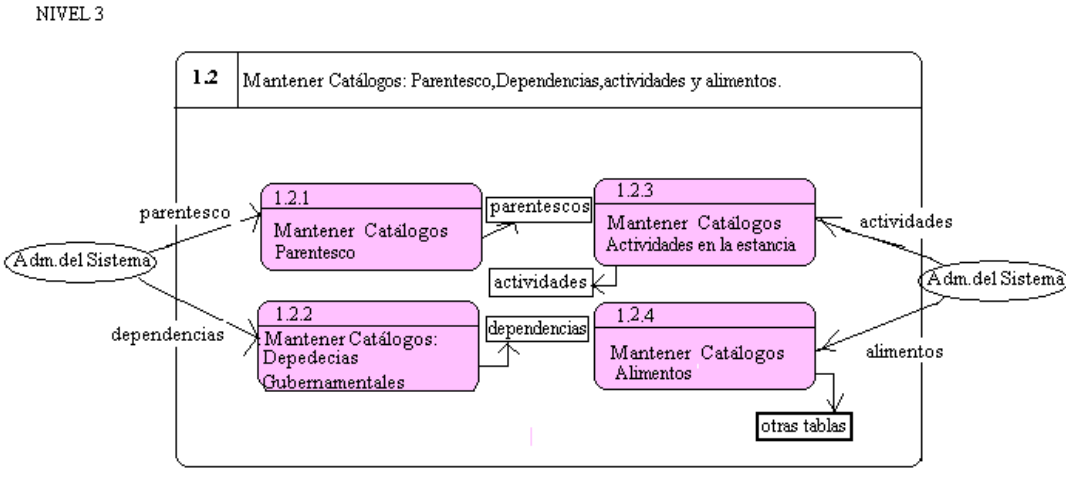


Fig.1.26. Diagrama de Flujo de Datos. Nivel3: Mantener catálogos

NIVEL 3

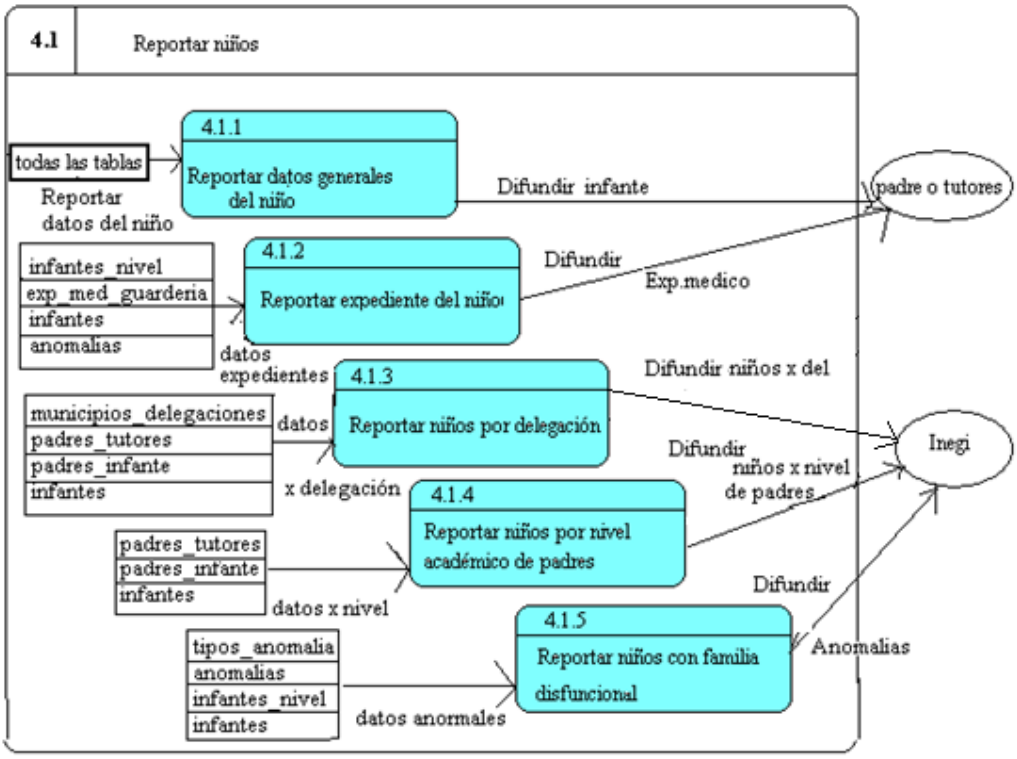


Fig.1.27. Diagrama de Flujo de Datos. Nivel 3: Reportar Niños

NIVEL 3

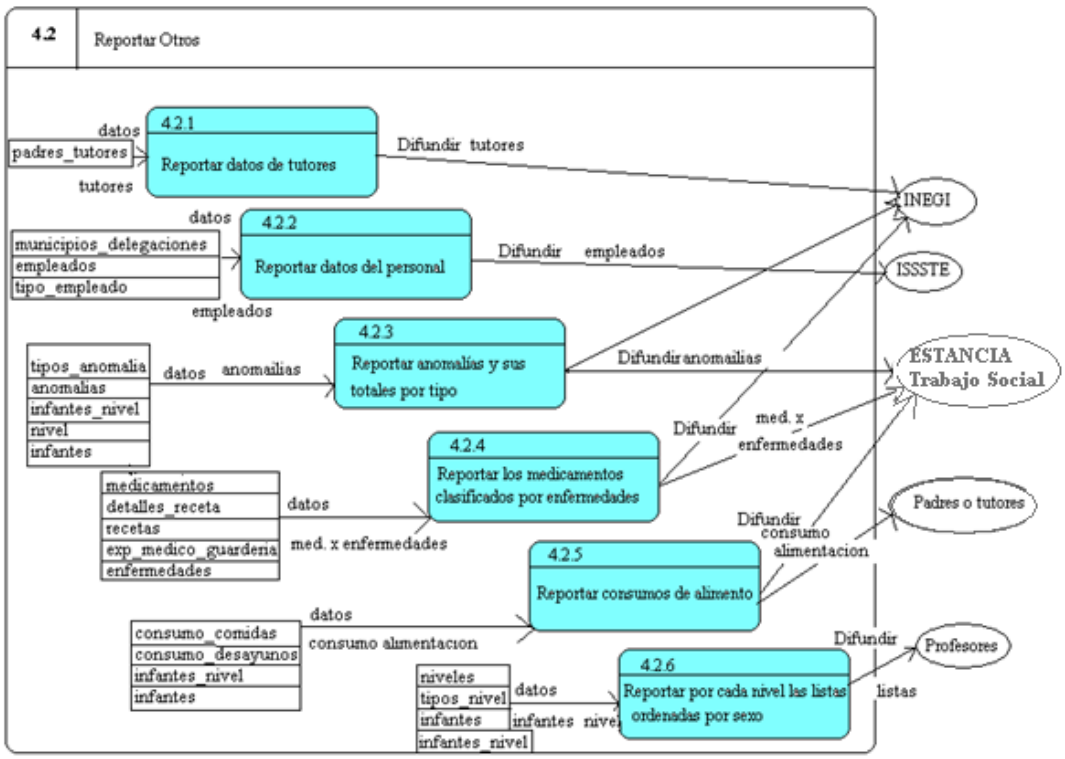


Fig.1.28. Diagrama de Flujo de Datos. Nivel3: Reportar Otros.

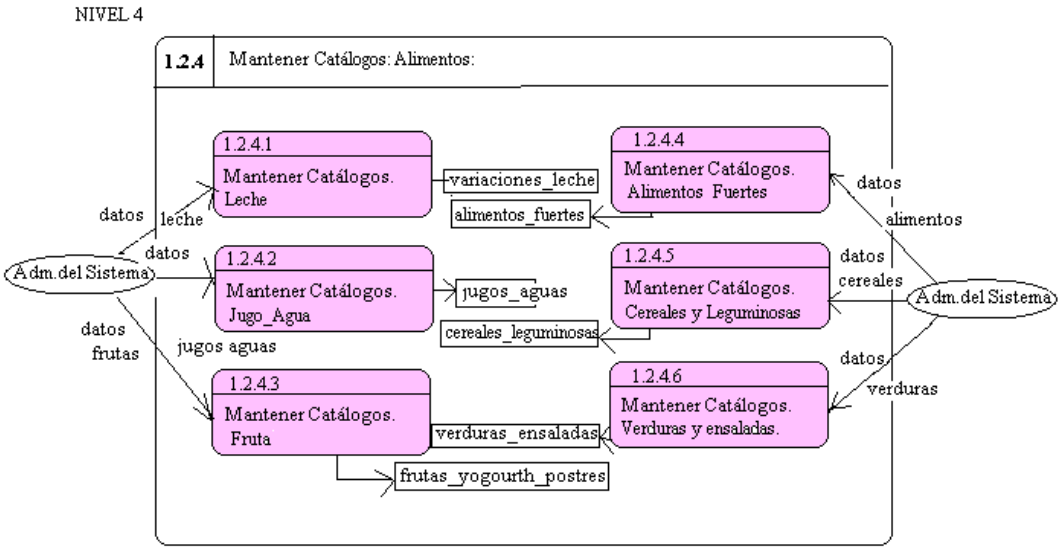


Fig.1.29. Diagrama de Flujo de Datos. Nivel4: Mantener catálogos alimentos.

Uso de la metodología RATIONAL UNIFIED PROCESS (RUP)

El Rational Unified Process (RUP) es un proceso iterativo de desarrollo de software creado por el Rational Software Corporation, una división de IBM desde 2003.

La metodología RUP, llamada así por sus siglas en inglés Rational Unified Process, divide en 4 fases el desarrollo del software:

• **Inicio:**	El objetivo en esta etapa es determinar la visión del proyecto. (puesta en marcha)
• **Elaboración:**	Su objetivo es determinar la arquitectura óptima. (Definición, análisis, diseño)
• **Construcción:**	Su objetivo es llegar a obtener la capacidad operacional inicial. (Implementación)
• **Transición:**	El objetivo es llegar a obtener la liberación del proyecto. (Fin del proyecto y puesta en marcha)

Cada uno de estas etapas es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada, Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes.

Los elementos de RUP son los encargados de ir describiendo lo que se ha producido, las habilidades necesarias y el paso a paso la explicación que describe cómo los objetivos concretos de desarrollo se han van logrando.

- **Funciones (quien)** – Su papel es definir un conjunto de habilidades, competencias y responsabilidades.
- **Productos de trabajo (qué)** - Representa algo como resultado de una tarea, incluidos todos los documentos y modelos producidos mientras se trabaja a través del proceso.
- **Tareas (cómo)** - Un grupo describe una unidad de trabajo asignado a una función que proporciona un resultado significativo.
  Cabe mencionar que el ciclo de vida que se desarrolla por cada iteración se debe de llevar mediante nueve disciplinas:

DISCIPLINAS DE LA INGENIERÍA	
1. Modelado de Negocios:	Entendiendo las necesidades del negocio
2. Requisitos:	Trasladando las necesidades del negocio a un sistema automatizado.
3. Análisis y diseño:	Trasladando los requisitos dentro de la arquitectura de software.
4. Ejecución o implementación:	Creando software que se ajuste a la arquitectura y que tenga un comportamiento deseado.
5. Pruebas:	Asegurándose que el comportamiento requerido es el correcto y que todos lo solicitado está presente.
6. Despliegue:	Implementado el sistema.
DISCIPLINAS DE SOPORTE	
7. Configuración y Gestión del Cambio:	Guardar todas las versiones del proyecto.
8. Gestión de Proyectos:	Administrar horarios y recursos.
9. Medio Ambiente:	Administrando el ambiente de desarrollo

Es recomendable que a cada una de estas iteraciones se les clasifique y ordene según su prioridad, y que cada una se convierta luego en un entregable al cliente.

Esto trae como beneficio la retroalimentación que se tendría en cada entregable y en cada iteración.

La metodología RUP es más adaptable para proyectos de largo plazo.

Uso de la metodología UNIFIED MODELING LANGUAGE (UML)

Lenguaje Unificado de Modelado es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software.
Es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos.

UML es el lenguaje de modelado estándar para crear planos de software. Permite:

- **Visualizar** la solución y facilitar la comunicación
- **Especificar** modelos precisos, completos y sin ambigüedad
- **Construir** código base mediante herramientas en varios lenguajes de programación (php,asp,java,c,c#,c++ etc..)
- **Documentar** arquitectura, requerimientos, pruebas, procesos de negocio, páginas web

Clasificación de los Diagramas:

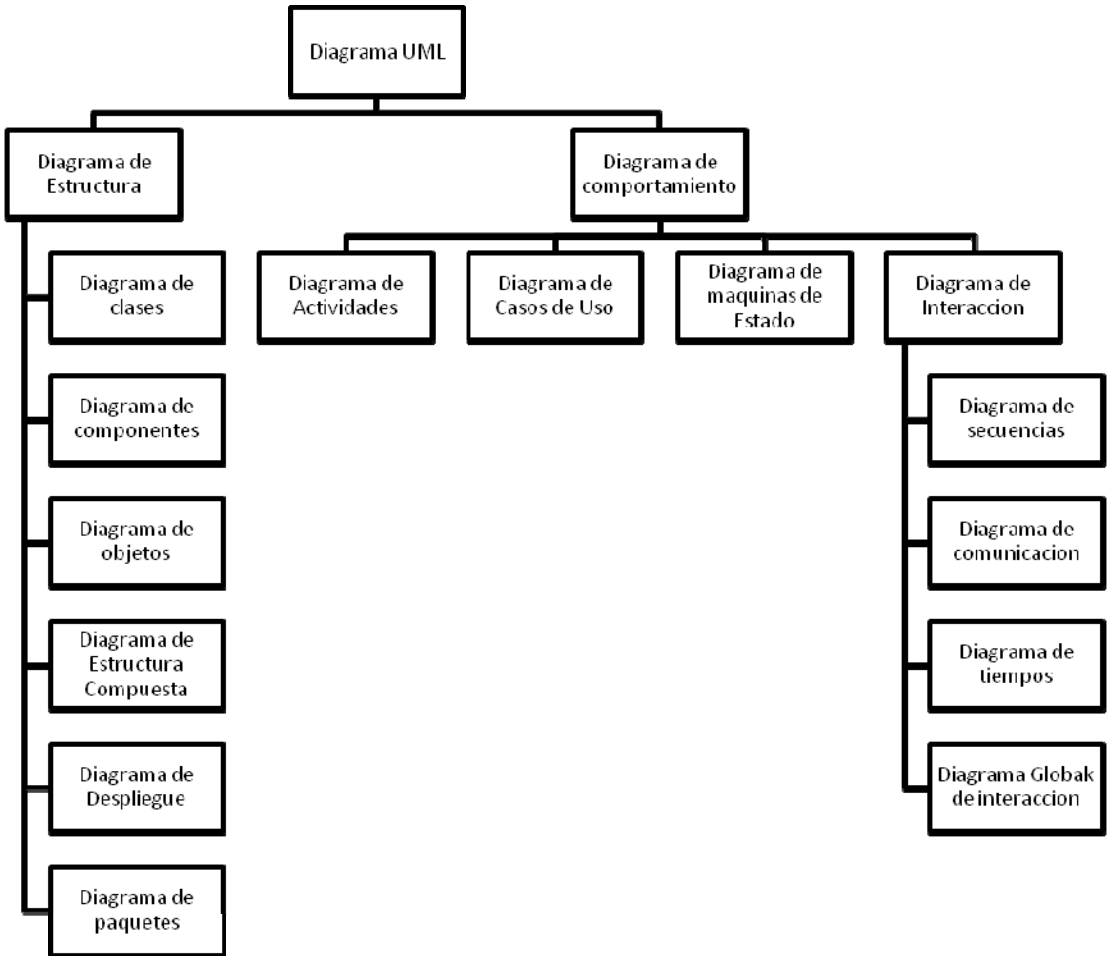


Fig.1.30. Diagrama en metodología UML.

- Los ***Diagramas de Estructura*** enfatizan en los elementos que deben existir en el sistema modelado
- Los ***Diagramas de Comportamiento*** enfatizan en lo que debe suceder en el sistema modelad
- Los ***Diagramas de Interacción*** son un subtipo de diagramas de comportamiento, que enfatiza sobre el flujo de control y de datos entre los elementos del sistema modelado

DIAGRAMA DE ACTIVIDADES

1 Mantener catálogos del sistema

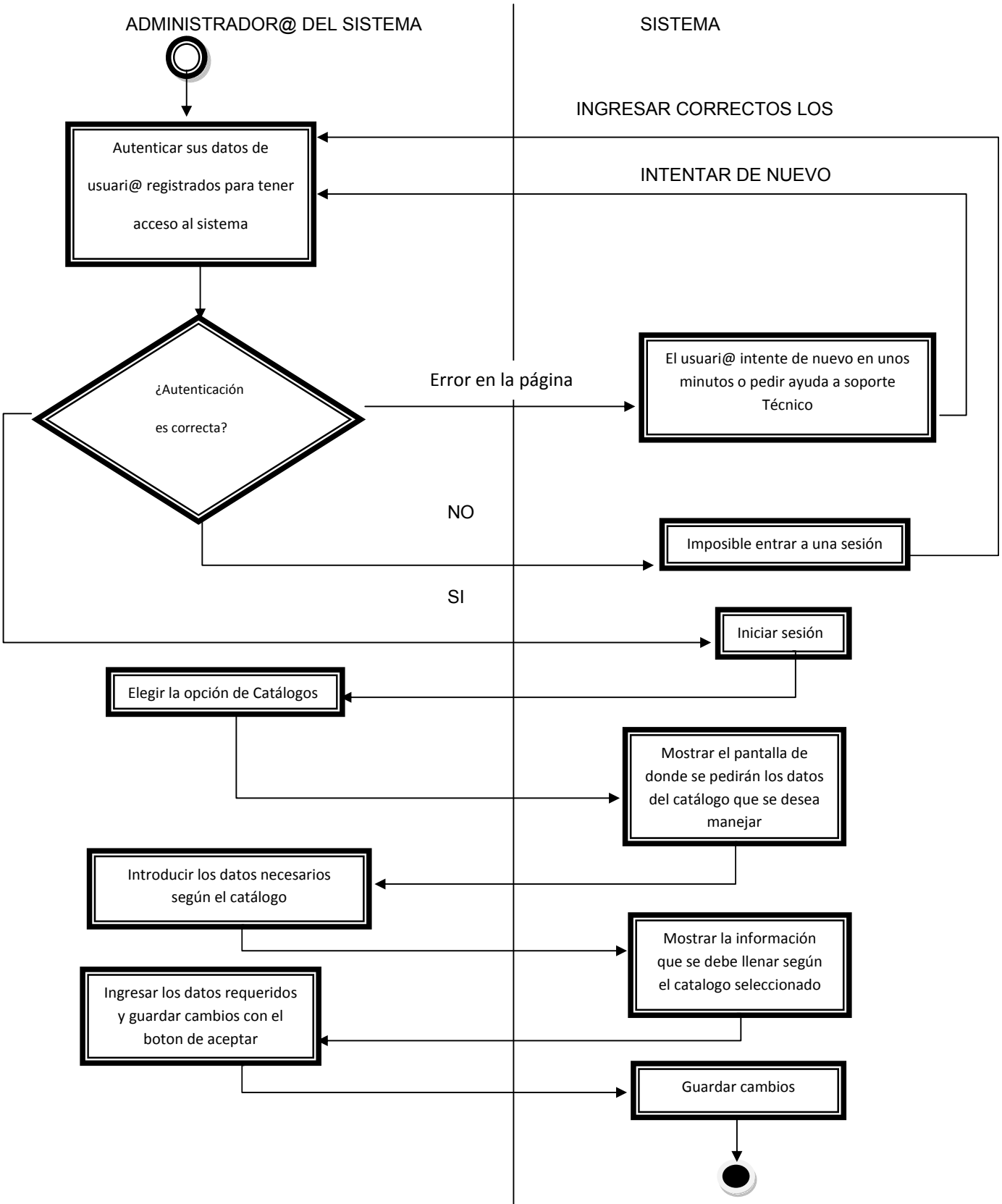


DIAGRAMA DE ACTIVIDADES

2 Mantener el registro de l@s niñ@s

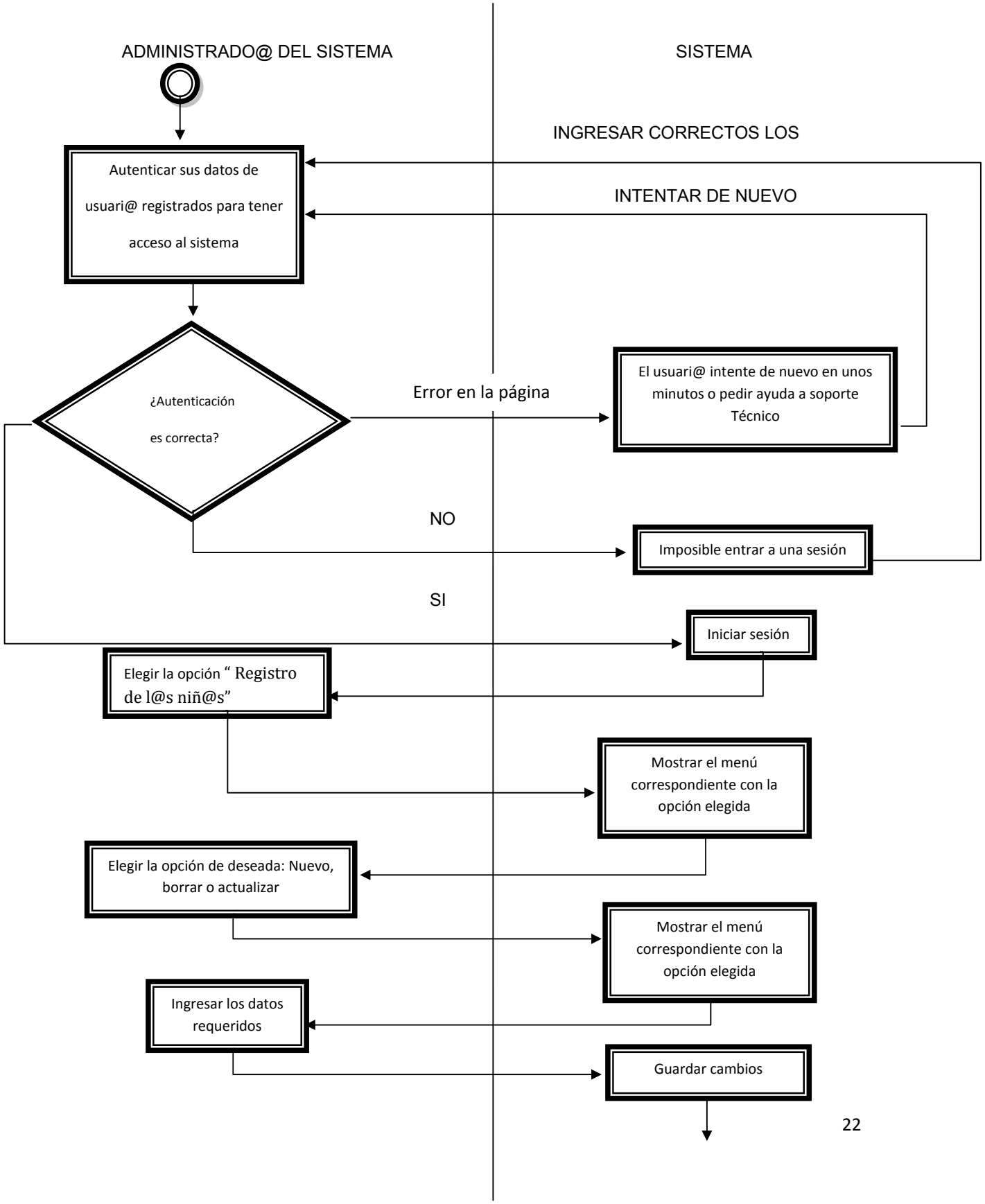


DIAGRAMA DE ACTIVIDADES

3.1 Llevar control de alimentación

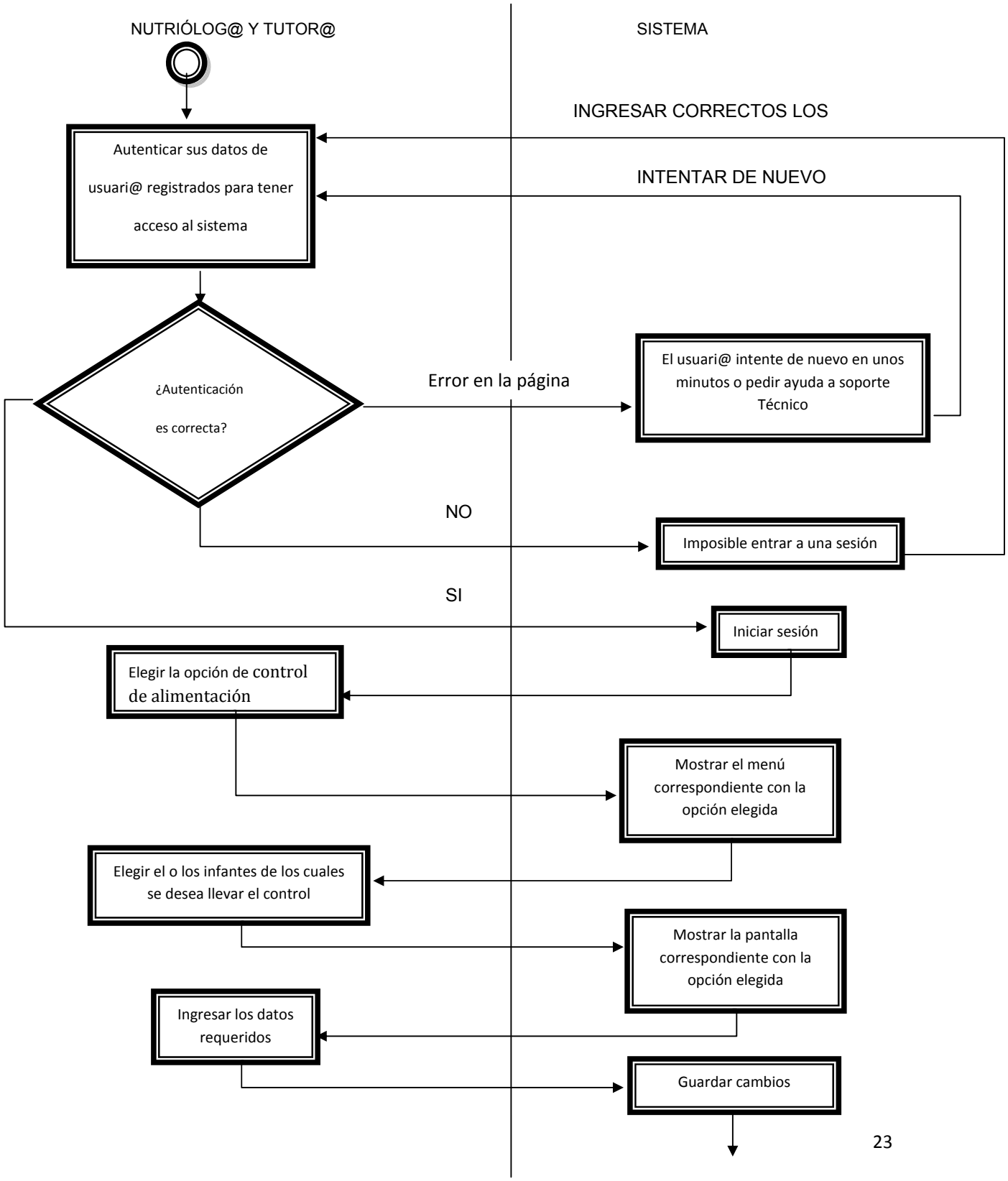


DIAGRAMA DE ACTIVIDADES

3.2 Llevar control de Faltas

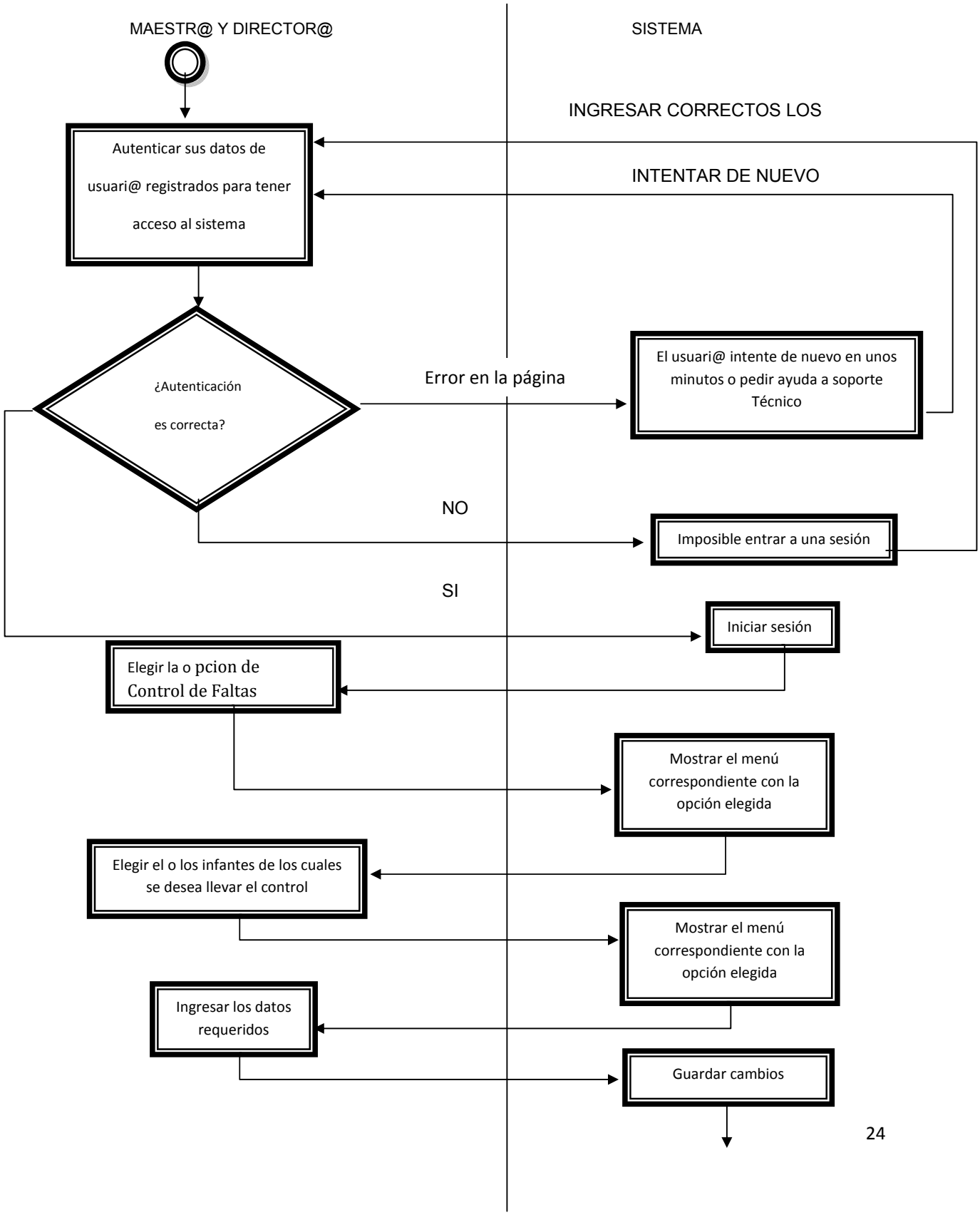


DIAGRAMA DE ACTIVIDADES

3.3 Actualizar expediente médico

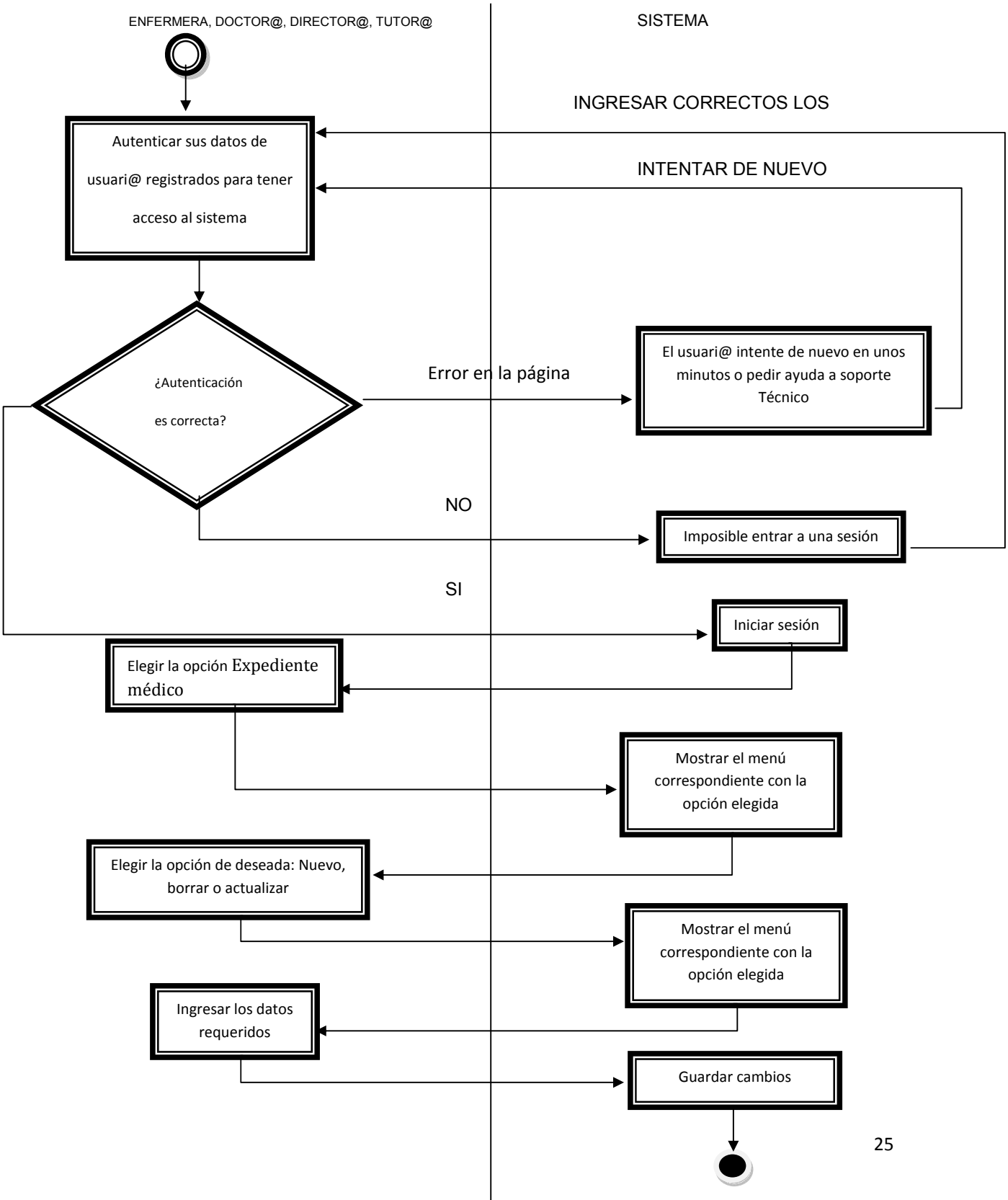


DIAGRAMA DE ACTIVIDADES

4.1.1 Reportar datos generales del niñ@

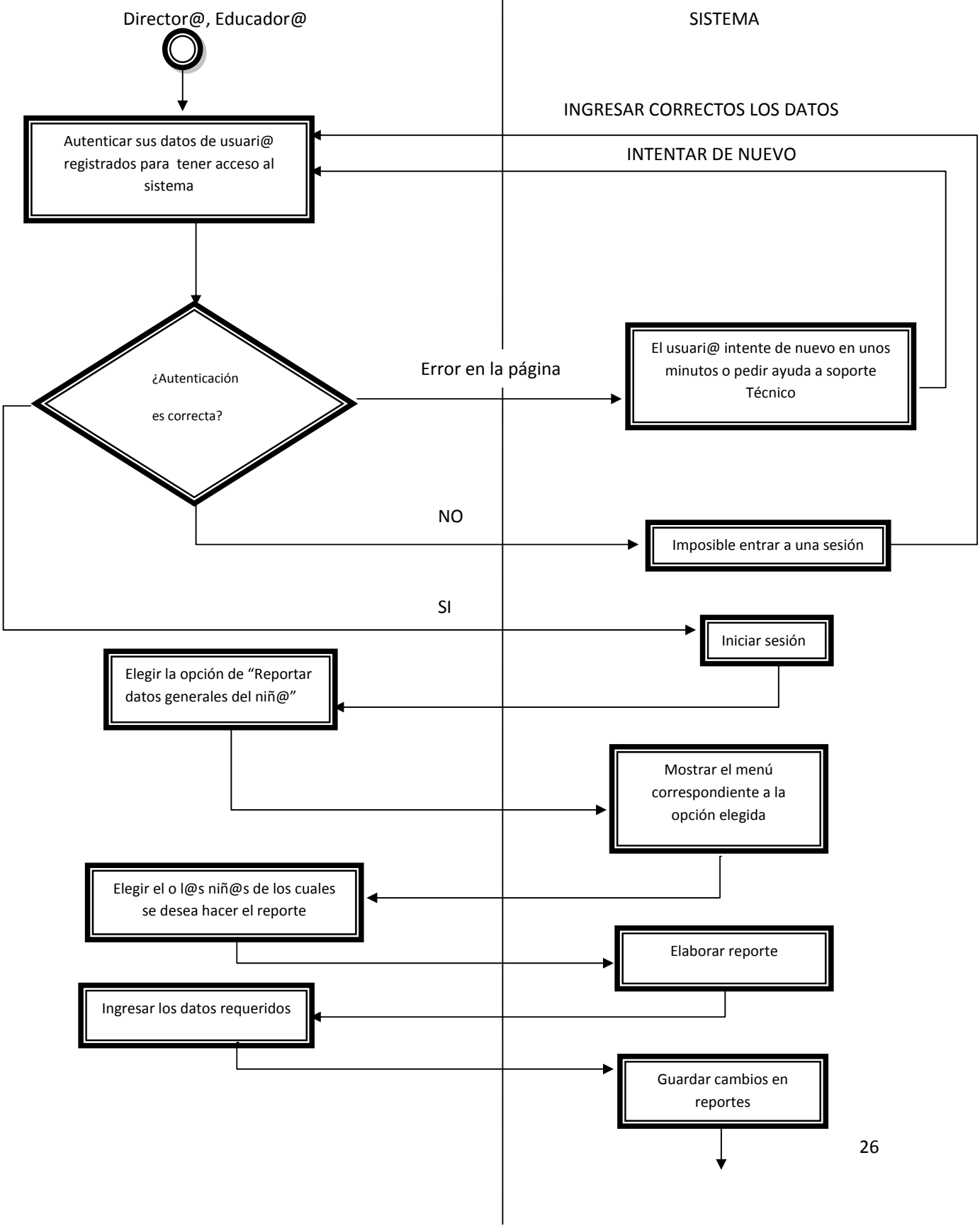


DIAGRAMA DE ACTIVIDADES

4.1.2 Reportar expediente del niñ@

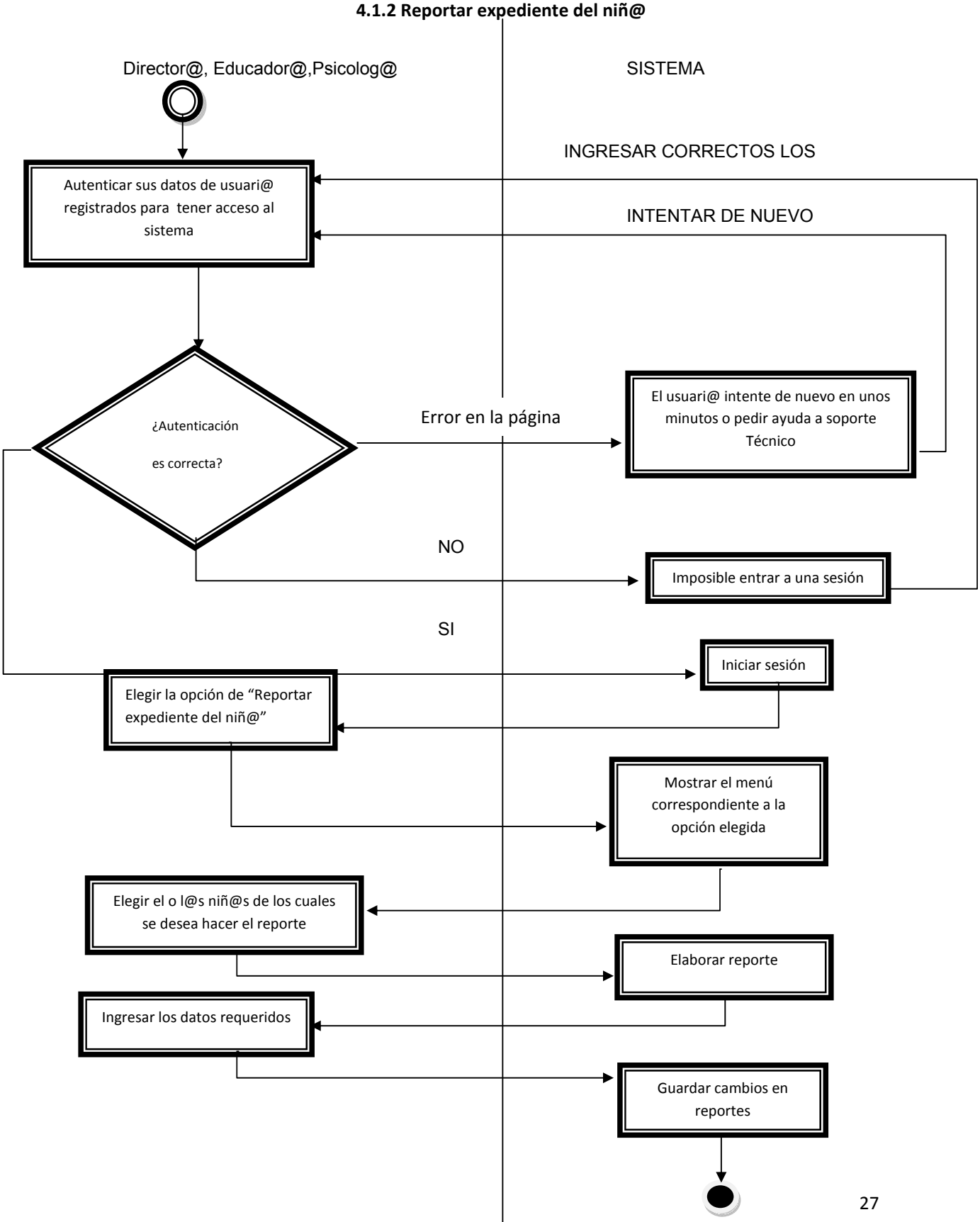


DIAGRAMA DE ACTIVIDADES

4.1.3 Reportar niños por delegación

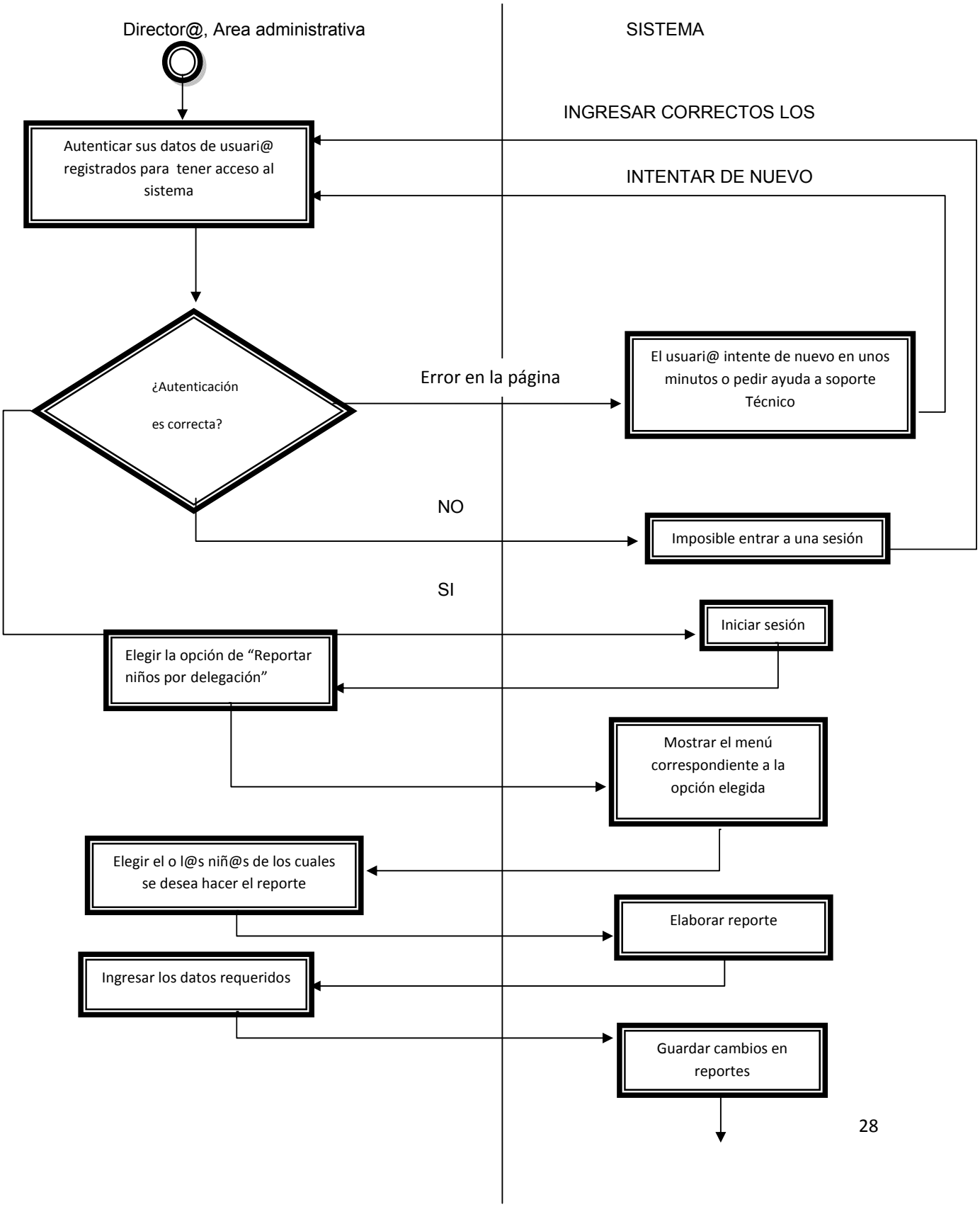


DIAGRAMA DE ACTIVIDADES

4.1.4 Reportar niños por nivel académico de tutores

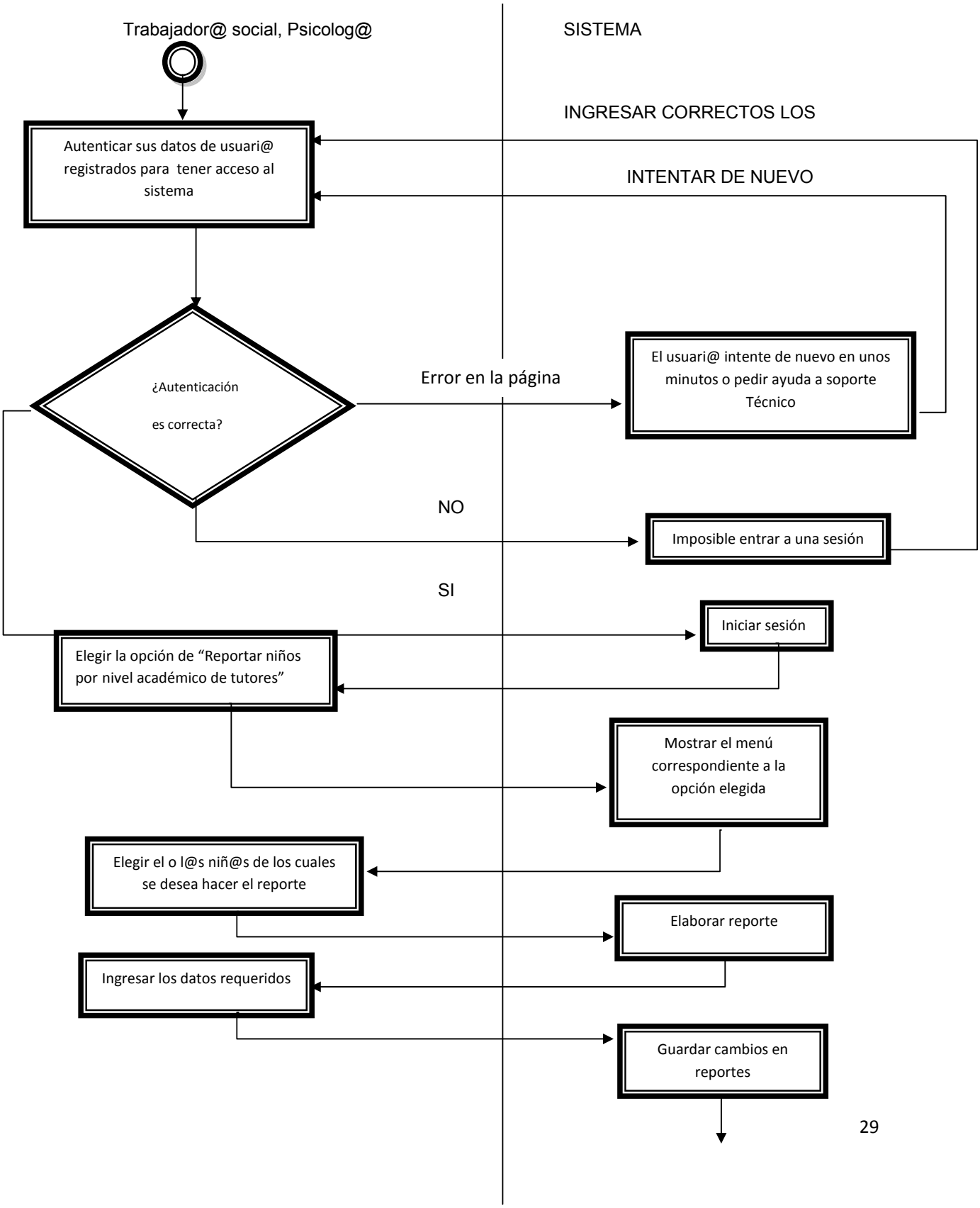


DIAGRAMA DE ACTIVIDADES

4.2.1 Reportar datos de tutores

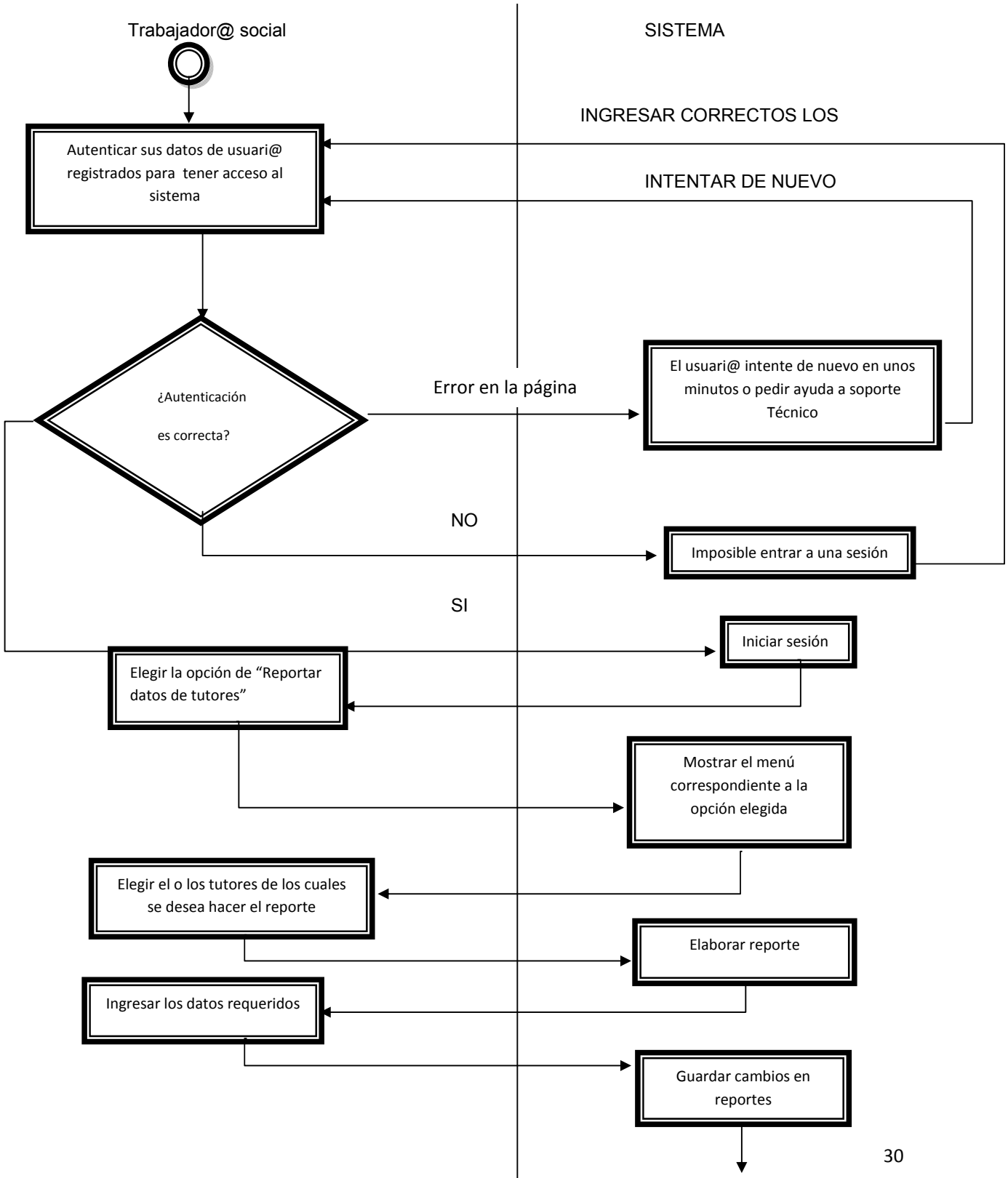


DIAGRAMA DE ACTIVIDADES

4.2.2 Reportar datos del personal de delagación del ISSSTE

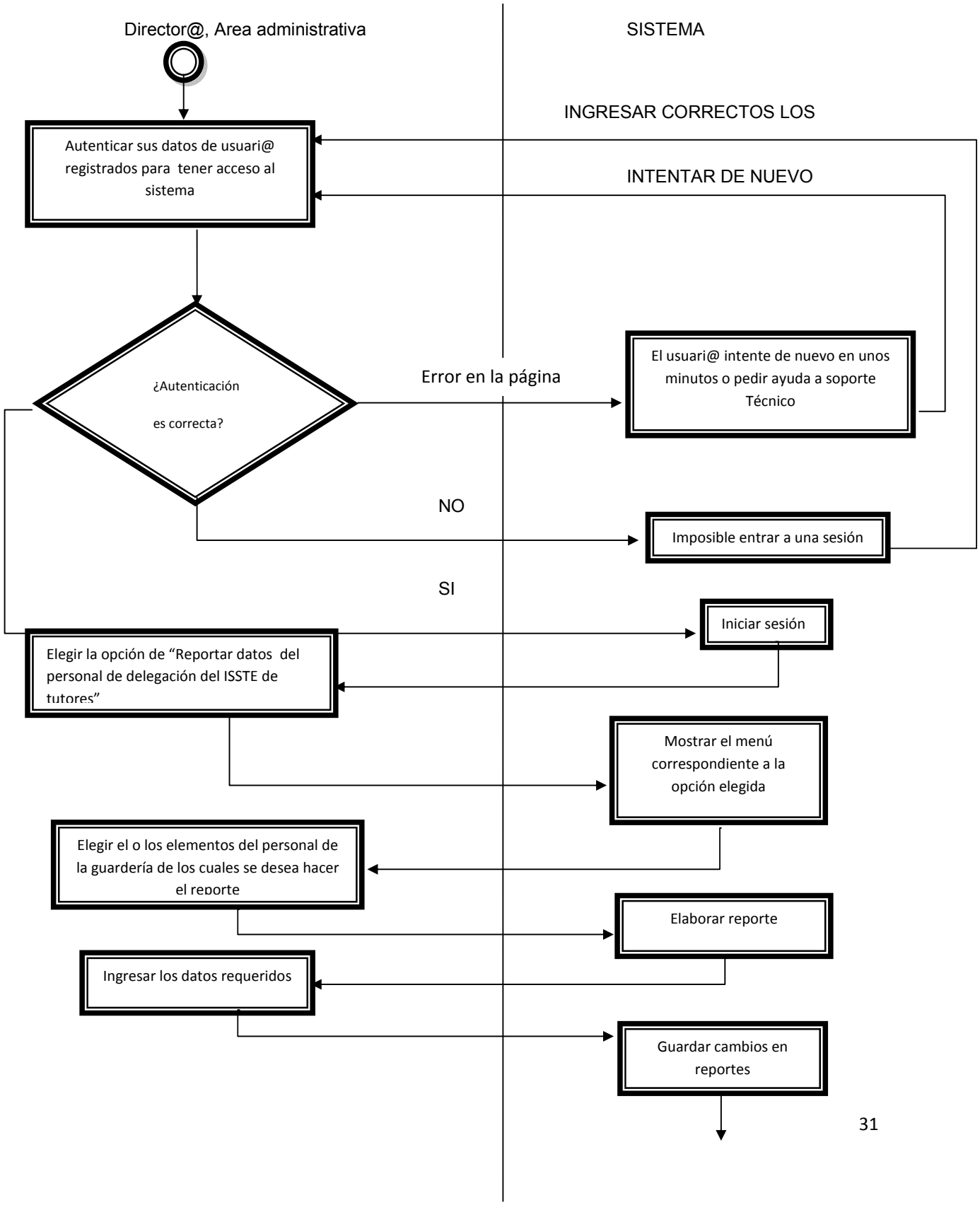


DIAGRAMA DE ACTIVIDADES

4.2.3 Reportar anomalías y sus totales por tipo de anomalía

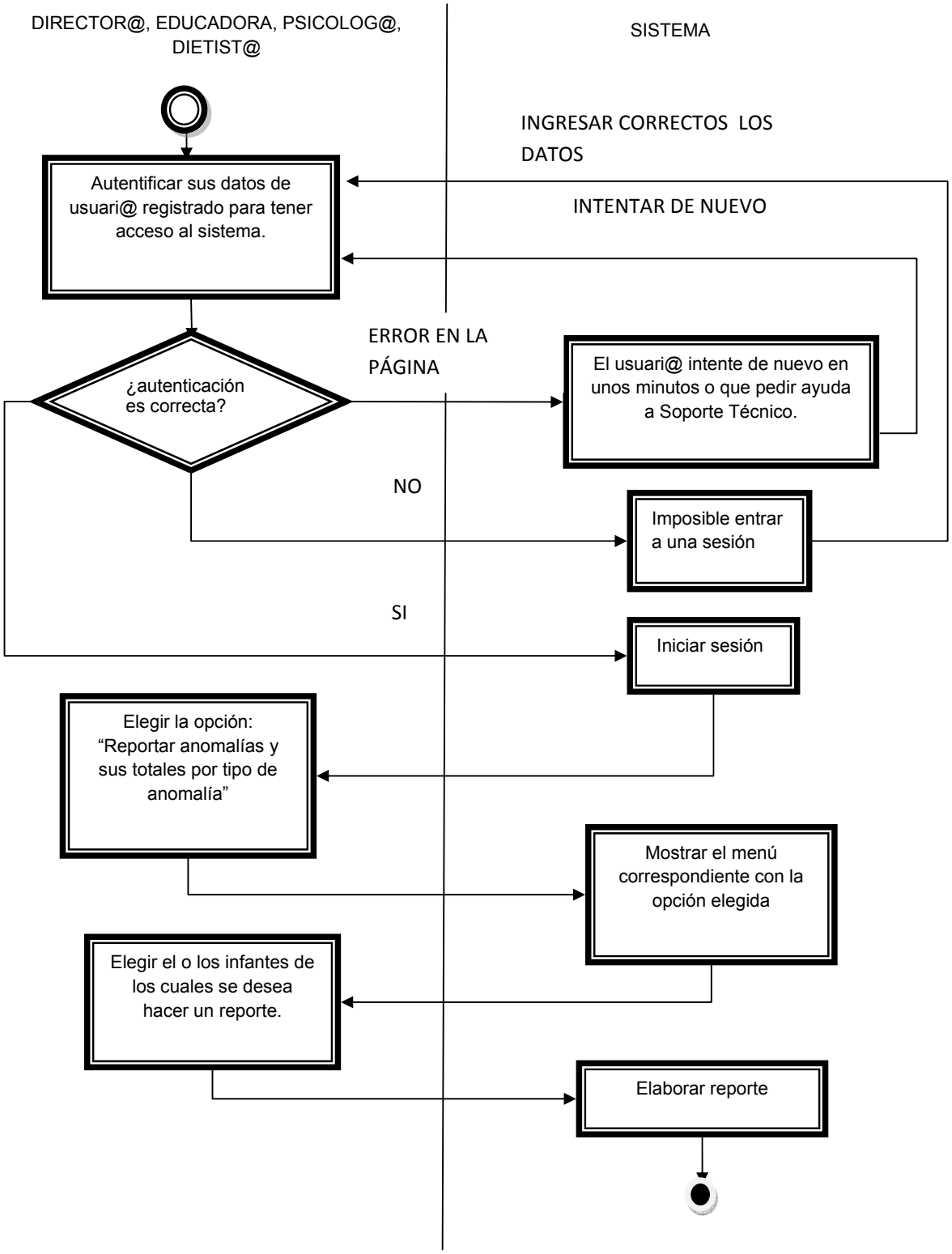


DIAGRAMA DE ACTIVIDADES

4.2.4 Reportar los medicamentos suministrados clasificados por enfermedades

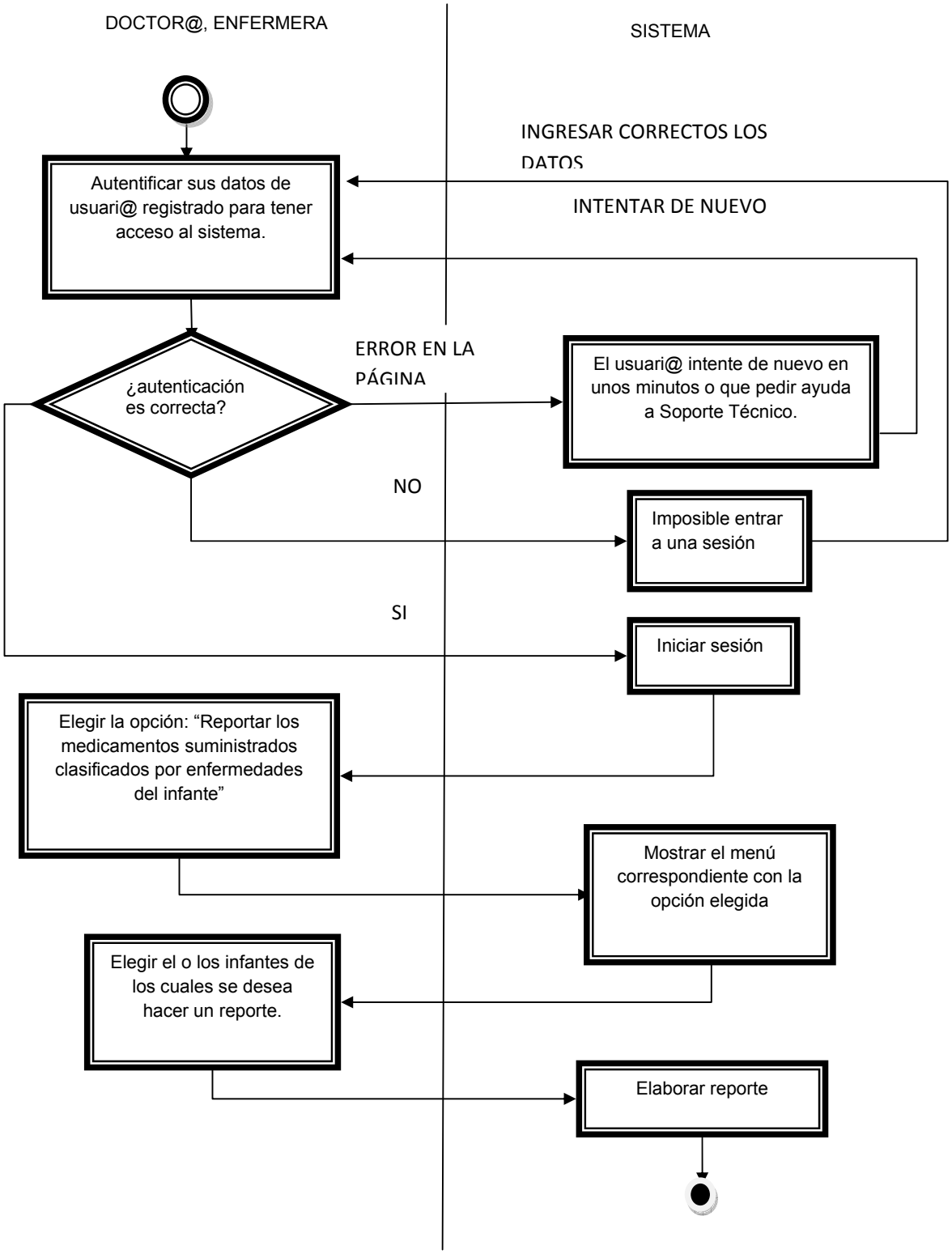


DIAGRAMA DE ACTIVIDADES

4.2.5 Reportar consumo de alimentos

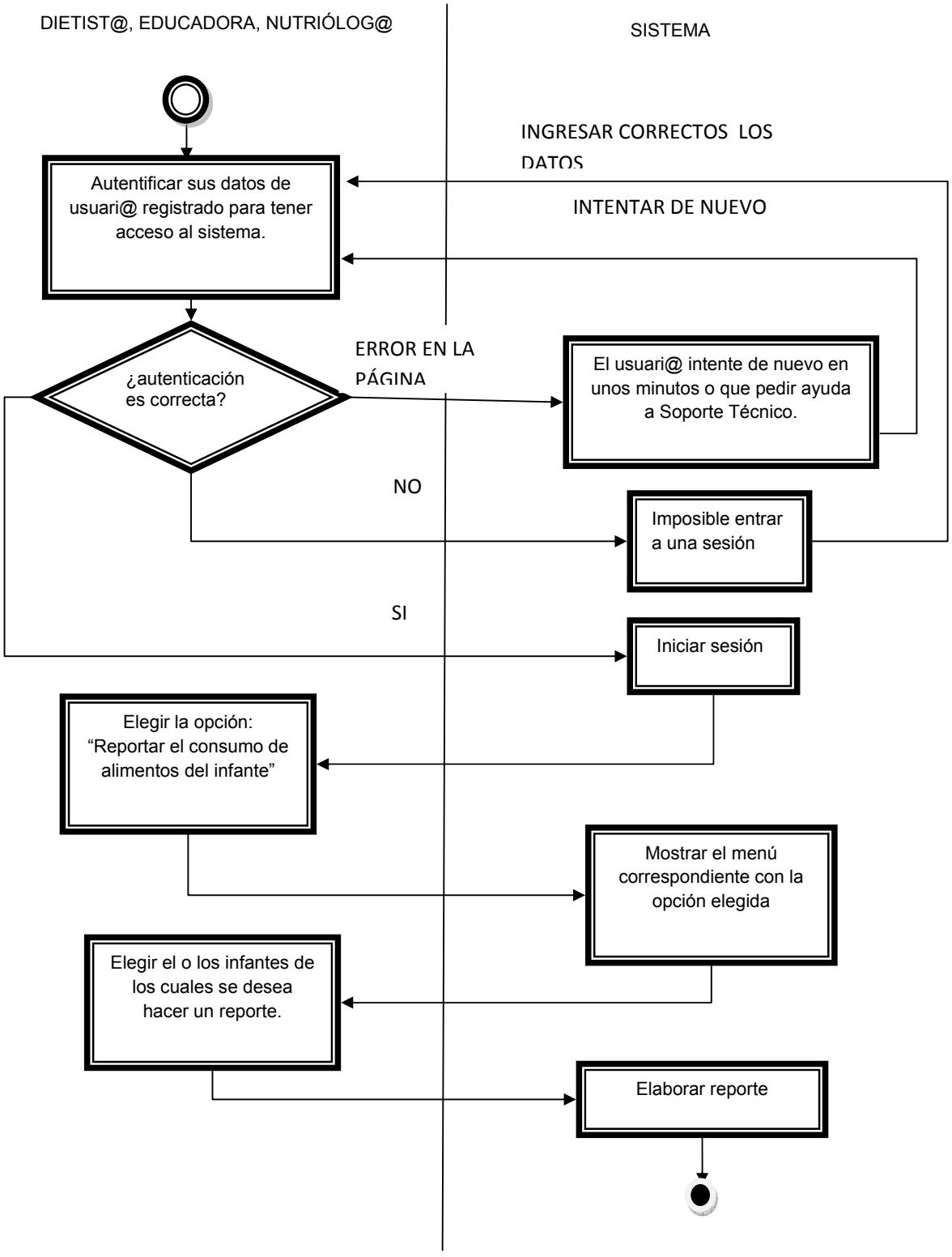


DIAGRAMA DE ACTIVIDADES

4.2.6 Reportar por cada nivel las listas ordenadas por sexo

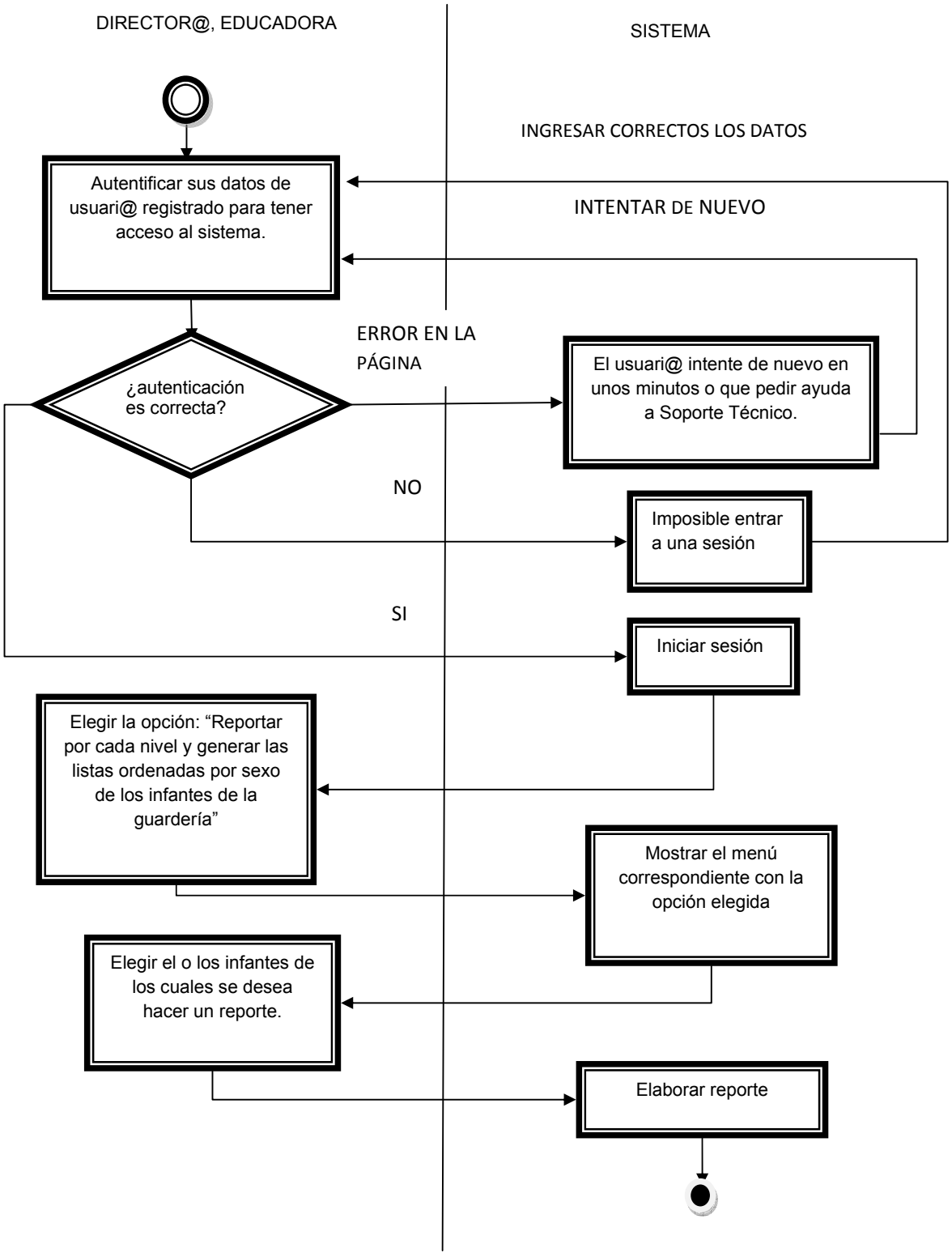


DIAGRAMA DE ACTIVIDADES

5 Respaldar y recuperar información

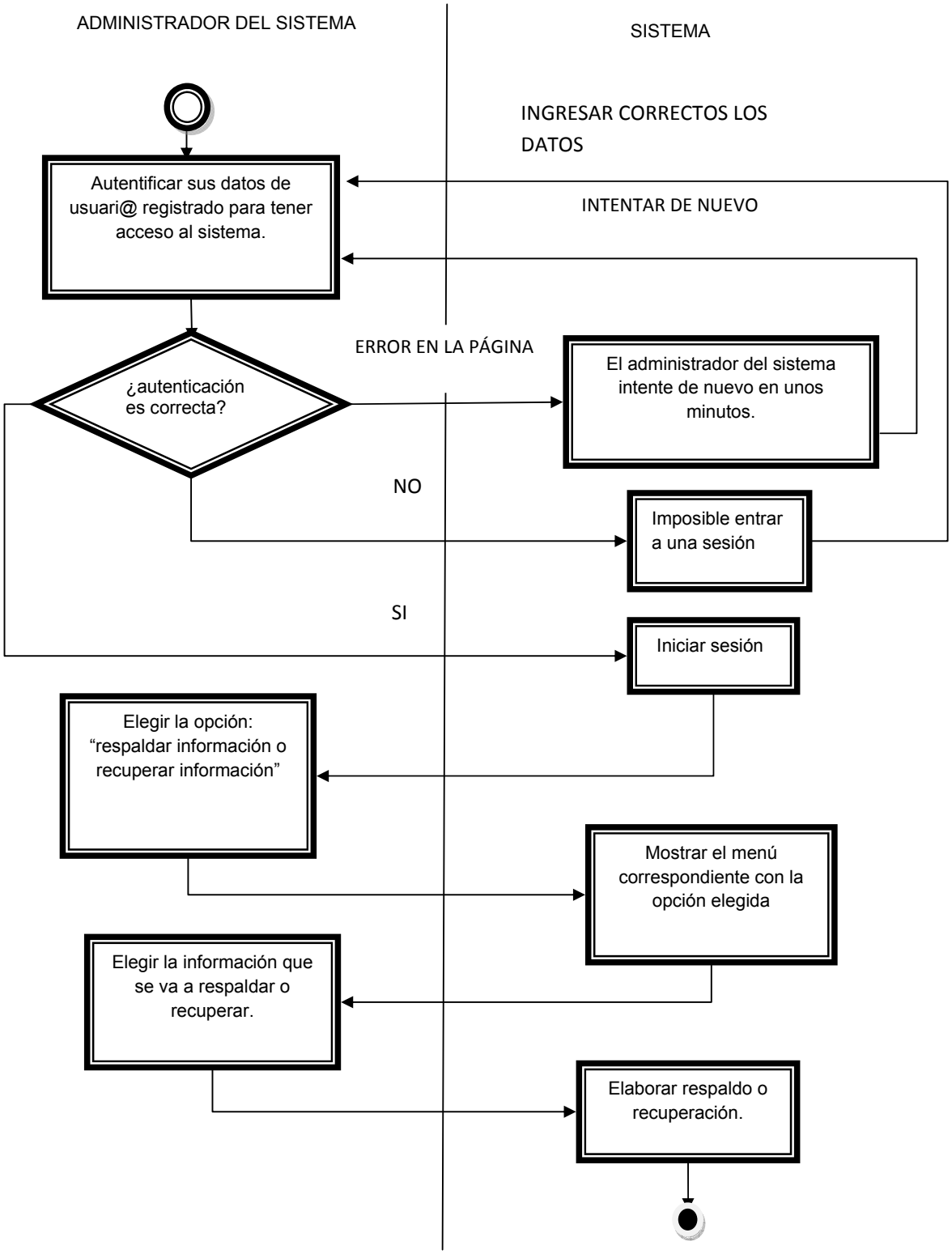
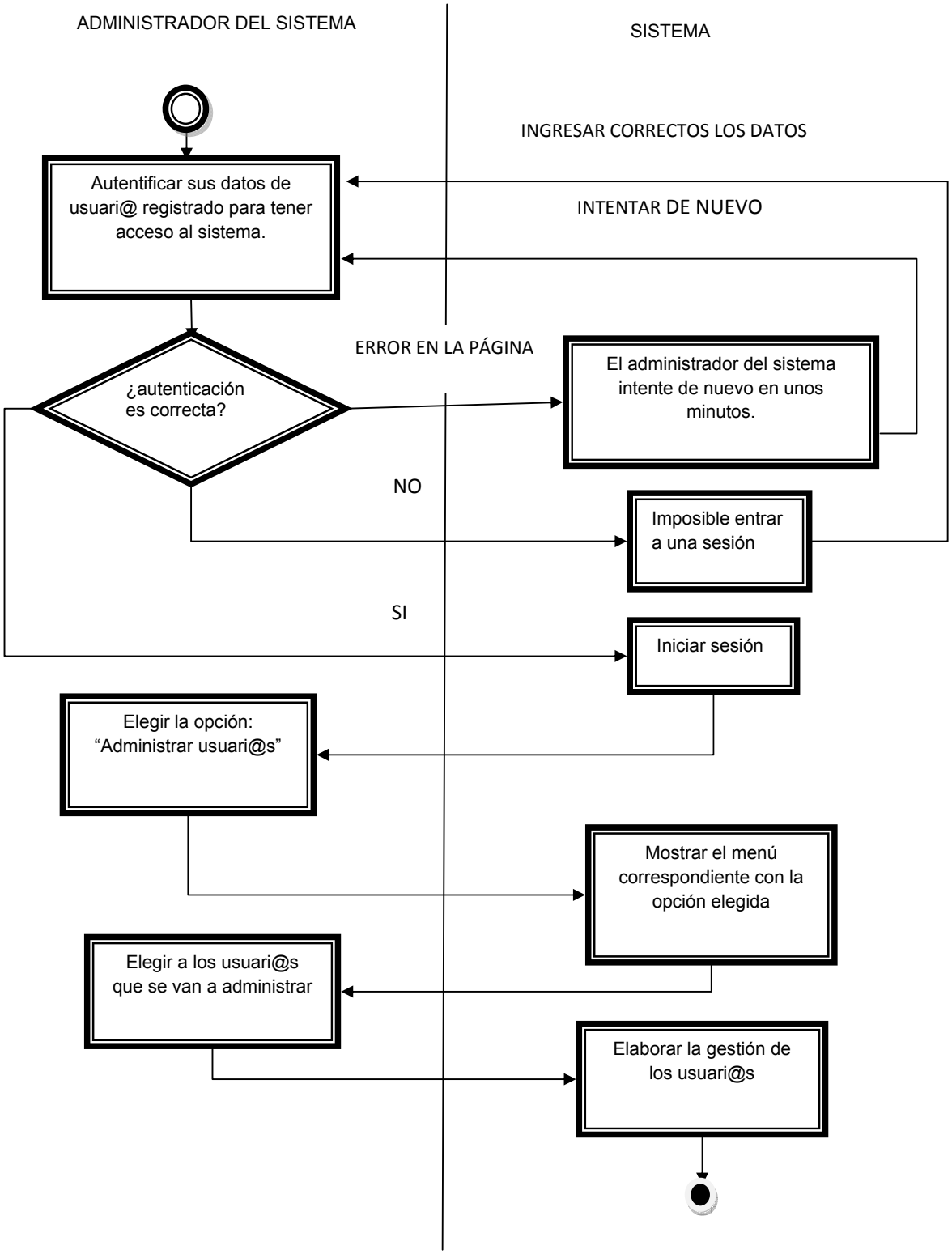


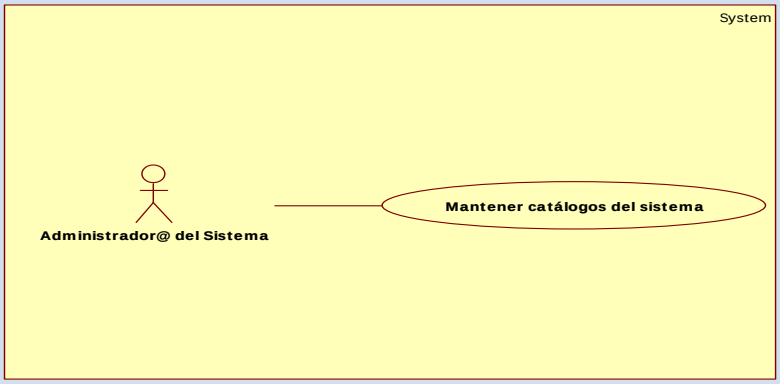
DIAGRAMA DE ACTIVIDADES

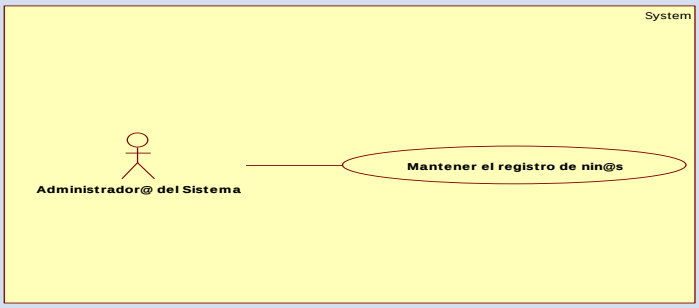
6 Administrar Usuari@s

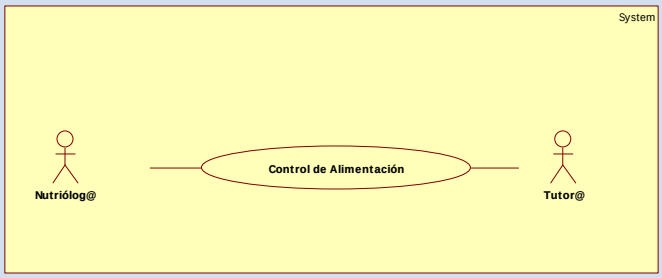


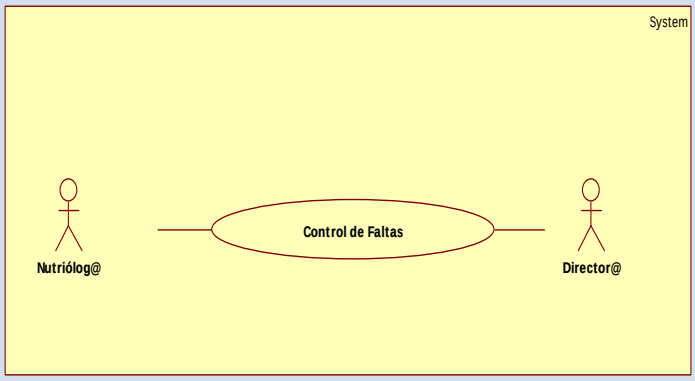
Ahora se muestran los diagramas de casos de uso que nos permitirán especificar y modelar los requerimientos de nuestro sistema.

CASOS DE USO

<table>
<tr><td>ID</td><td>1</td></tr>
<tr><td>Nombre</td><td>Mantener catálogos del sistema</td></tr>
<tr><td>Actores</td><td>Administrador@ del Sistema</td></tr>
<tr><td>Propósito</td><td>Mantener actualizados los catálogos que conforman al sistema.</td></tr>
<tr><td>Descripción</td><td>Administrador@ creará los registros de cada uno de los catálogos que forman el sistema, además de realizar las actualizaciones y mantenimiento correspondiente</td></tr>
<tr><td>Observaciones</td><td>Los catálogos a los que le dará mantenimiento son: Estancia, empleados, salud, parentesco, dependencias, actividades y alimentos</td></tr>
<tr><td>Diagrama</td><td> 
</td></tr>
<tr><td>Salidas</td><td>➢ Identificación de los registros de cada catálogo.</td></tr>
<tr><td>Precondiciones</td><td>➢ Tener la información correspondiente a cada catálogo</td></tr>
<tr><td>Postcondiciones</td><td>➢ Los catálogos estarán listos para su acceso</td></tr>
<tr><td>Flujo principal</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>1. El caso de uso inicia cuando el administrador@ cuenta con la información necesaria para crear los registros de los catálogos .una vez que entro al menú principal selecciona la opción del catalogo que desea modificar, actualizar o crear</td><td>2. El sistema despliega una pantalla donde se pedirá los datos del catálogo que se desea manejar</td></tr>
<tr><td>3. El administrador@ del sistema, introduce los datos necesarios, según el catálogo</td><td>4. El sistema presenta en los formularios sólo la información que se deberá llenar, según el catálogo seleccionado.</td></tr>
<tr><td>5. El administrador@ del sistema después de revisar los datos capturados y si están correctos los guarda para terminar el mantenimiento de catálogos, con el botón de “aceptar”.</td><td>6. El sistema valida que los datos sean correctos ver FA1 y FA2. en caso de que no lo sean ver FE1 y FE2.</td></tr>
</table>
</td></tr>
<tr><td>Flujos Alternos</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FA1. Introdujo datos correctos</td><td>FA2.El sistema actualiza las tablas para guardar la modificación que se le hizo a los catálogos</td></tr>
</table>
</td></tr>
<tr><td>Flujos de Excepción</td><td>
<table>
<tr><th>*Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FE1. El administrador del sistema introduce los datos en forma incompleta o incorrecta.</td><td>FE2. El sistema envía un mensaje que hace falta datos o datos incorrectos y se posiciona el puntero en el primer dato incorrecto o incompleto.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>ID</td><td>2</td></tr>
<tr><td>Nombre</td><td>Mantener el registro de l@s niñ@s</td></tr>
<tr><td>Actores</td><td>Administrador@ del Sistema</td></tr>
<tr><td>Propósito</td><td>Mantener actualizados los registros de l@s niñ@s</td></tr>
<tr><td>Descripción</td><td>Administrador@ creará los registros de cada niñ@, además de realizar las actualizaciones y mantenimiento correspondiente</td></tr>
<tr><td>Observaciones</td><td>Los registros a los que se les dará mantenimiento son: expediente y datos principales del niñ@</td></tr>
<tr><td>Diagrama</td><td> 
</td></tr>
<tr><td>Salidas</td><td>➢ Registros de l@s niñ@s.</td></tr>
<tr><td>Precondiciones</td><td>➢ Tener la información correspondiente de cada niñ@.</td></tr>
<tr><td>Postcondiciones</td><td>➢ Los registros estarán listos para su manipulación</td></tr>
<tr><td>Flujo principal</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>1. El caso de uso inicia cuando el/la administrador@ cuenta con la información necesaria para crear los registros de los niños. Una vez que entro al menú principal selecciona la opción registrar niñ@</td><td>2. El sistema despliega una página pidiendo los datos del niñ@.</td></tr>
<tr><td>3. El/La administrador@ del sistema, introduce los datos necesarios, según el catálogo</td><td>4. El sistema presenta en los formularios sólo la información que se deberá llenar, según el catálogo seleccionado.</td></tr>
<tr><td></td><td>5. El sistema presenta en un formulario la información que se deberá llenar: CURP, nombre, apellido paterno, apellido materno, fecha de nacimiento, sexo, dirección [calle con número, colonia, código postal, delegación o municipio, estado], peso de nacimiento, estatura de nacimiento y CURP hermano en caso de contar con un gemelo o hermano mayor y fecha de ingreso.</td></tr>
</table>
</td></tr>
<tr><td>Flujos Alternos</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FA1. Introdujo datos correctos</td><td>FA2.El sistema actualiza las tablas</td></tr>
</table>
</td></tr>
<tr><td>Flujos de Excepción</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FE1. Introdujo los datos en forma incompleta o incorrecta.</td><td>FE2. El sistema envía un mensaje que hace falta datos o datos incorrectos y se posiciona el puntero en el primer dato incorrecto o incompleto</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>ID</td><td>3.1</td></tr>
<tr><td>Nombre</td><td>Llevar control de alimentación</td></tr>
<tr><td>Actores</td><td>Nutriólog@ y Tutor@</td></tr>
<tr><td>Propósito</td><td>Llevar un control de la alimentación de cada niñ@ de la estancia, y el/la Tutor@ pueda conocer esta información</td></tr>
<tr><td>Descripción</td><td>Llevar un control de lo que come cada niñ@ en el transcurso del día, desde el desayuno hasta la comida</td></tr>
<tr><td>Observaciones</td><td></td></tr>
<tr><td>Diagrama</td><td> 
</td></tr>
<tr><td>Salidas</td><td>➢ Identificación de la alimentación de cada uno de l@s niñ@s</td></tr>
<tr><td>Precondiciones</td><td>➢ Tener la información correspondiente de cada niñ@, además de la comida que se les dará cada día.</td></tr>
<tr><td>Postcondiciones</td><td>➢ Se contara con la información de lo que cada niñ@ comió en el transcurso del día de cada día</td></tr>
<tr><td>Flujo principal</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>1. El caso de uso inicia cuando nutriólog@ cuenta con la información necesaria para crear los registros de lo que comió durante el día el infante cada profesora los proporciona de desayuno y de comida.</td><td>2. El sistema despliega una página pidiendo los datos de lo que comió el infante durante el día.</td></tr>
<tr><td>3. La nutrióloga, introduce los datos necesarios, según el infante.</td><td>4. El sistema presenta en los formularios sólo la información que se deberá llenar, según el infante y la comida.</td></tr>
<tr><td></td><td>5. El sistema valida que los datos sean correctos ver FA1. en caso de que no lo sean ver FE1.</td></tr>
</table>
</td></tr>
<tr><td>Flujos Alternos</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FA1. Introdujo datos correctos</td><td>FA2.El sistema actualiza las tablas para el control de alimentación</td></tr>
</table>
</td></tr>
<tr><td>Flujos de Excepción</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FE1. Introdujo los datos en forma incompleta o incorrecta.</td><td>FE2. El sistema envía un mensaje que hace falta datos o datos incorrectos y se posiciona el puntero en el primer dato incorrecto o incompleto</td></tr>
</table>
</td></tr>
</table>

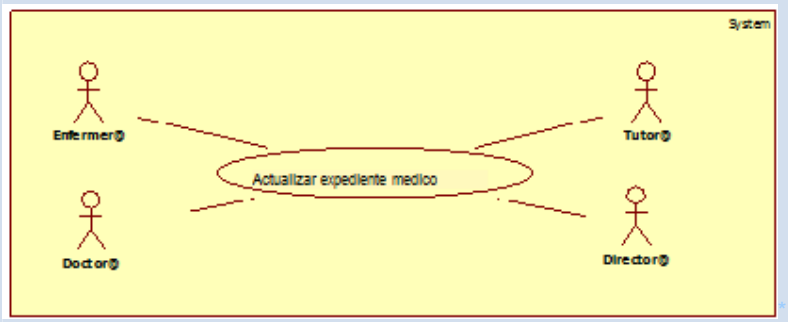
<table>
<tr><td>ID</td><td>3.2</td></tr>
<tr><td>Nombre</td><td>Llevar control de Faltas</td></tr>
<tr><td>Actores</td><td>Maestr@ y Direct@r</td></tr>
<tr><td>Propósito</td><td>Llevar un control de faltas de cada niñ@ de la estancia, y el Director@ pueda conocer esta información</td></tr>
<tr><td>Descripción</td><td>La Maestra llevará un control de las inasistencias de cada niñ@</td></tr>
<tr><td>Observaciones</td><td></td></tr>
<tr><td>Diagrama</td><td> 
</td></tr>
<tr><td>Salidas</td><td>➢ Identificar a niñ@s que faltan, para verificar el motivo y justificarlas</td></tr>
<tr><td>Precondiciones</td><td>➢ Tener la información correspondiente a las asistencias de cada niñ@</td></tr>
<tr><td>Postcondiciones</td><td>➢ Se contará con la información de las faltas de cada niñ@</td></tr>
<tr><td>Flujo principal</td><td>
<table>
<tr><th>*Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>1. caso de uso inicia cuando la maestra cuenta con la información necesaria para crear los registros de las inasistencias del niño</td><td>2. El sistema despliega una página pidiendo los datos de los niños que han faltado</td></tr>
<tr><td>3. La maestra, introduce los datos necesarios, según el infante.</td><td>4. El sistema presenta en los formularios sólo la información que se deberá llenar, según el infante.</td></tr>
<tr><td></td><td>5. El sistema valida que los datos sean correctos ver FA1. en caso de que no lo sean ver FE1.</td></tr>
</table>
</td></tr>
</table>

Flujos Alternos	

Actor	Acciones del Sistema
FA1. Introdujo datos correctos	FA2. El sistema actualiza las tablas para guardar la modificación que se le hizo a control de faltas

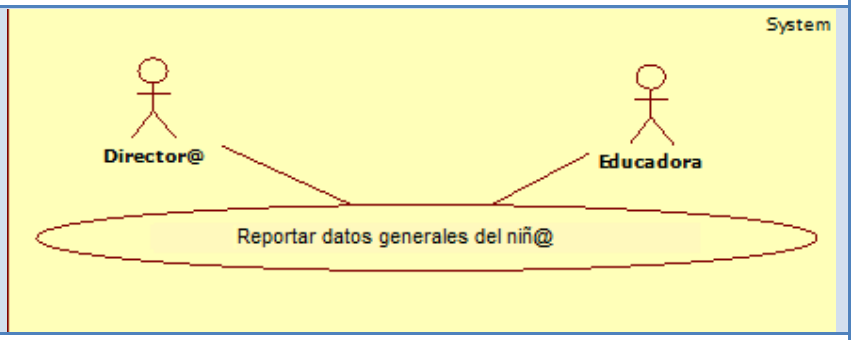
Flujos de Excepción	

Actor	Acciones del Sistema
FE1. Introdujo los datos en forma incompleta o incorrecta.	FE2. El sistema envía un mensaje que hace falta datos o datos incorrectos y se posiciona el puntero en el primer dato incorrecto o incompleto

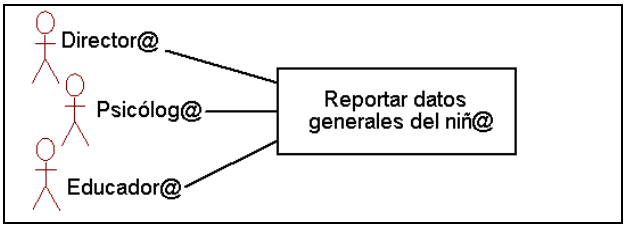
ID	**3.3**
Nombre	**Actualizar expediente médico**
Actores	Enfermera, Doctor@, Director@, Tutor@
Propósito	Actualizar el expediente medico de cada niñ@ de la estancia
Descripción	Enfermer@ y doctor@ actualizaran los datos del expediente médico de cada uno de los infantes, para que tutor@ y director@ cuenten con esta información
Observaciones	
Diagrama	
Salidas	➢ Identificación de las enfermedades de cada niñ@s
Precondiciones	➢ Tener la información correspondiente de cada niñ@, además de las enfermedades o anomalías que se puedan presentar
Postcondiciones	➢ Se contara con un expediente médico de cada uno de los infantes
Flujo principal	

Actor	Acciones del Sistema
1. El caso de uso inicia cuando la enfermera o el doctor, cuenten con la información necesaria para crear el expediente médico	2. El sistema despliega una página pidiendo los datos del expediente médico
3. el doctor o la enfermera, introduce los datos necesarios, según el infante.	**4.** El sistema presenta en los formularios sólo la información que se deberá llenar, correspondiente al infante y la enfermedad o anomalía
	5. El sistema valida que los datos sean correctos ver **FA1**. en caso de que no lo

	sean ver **FE1**.
Flujos Alternos	**Actor** / **Acciones del Sistema** FA1. Introdujo datos correctos / FA2. El sistema actualiza las tablas para guardar la modificación que se le hizo a control de faltas
Flujos de Excepción	**Actor** / **Acciones del Sistema** FE1. Introdujo los datos en forma incompleta o incorrecta. / FE2. El sistema envía un mensaje que hace falta datos o datos incorrectos y se posiciona el puntero en el primer dato incorrecto o incompleto

ID	4.1.1
Nombre	**Reportar datos generales del niñ@**
Actores	Director@, educadora
Propósito	Generar un reporte de los datos generales de cada uno de l@s niñ@s
Descripción	Proceso para obtener un reporte general por cada niñ@
Observaciones	
Diagrama	
Salidas	
Precondiciones	- Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	**Actor** / **Acciones del Sistema** **1.** Autentificar sus datos de usuari@ registrado para tener acceso al sistema. / **2.** Si la autenticación es correcta: - Iniciar sesión. **3.** Elegir la opción: "Reportar datos generales del niñ@" / **4.** Mostrar el menú correspondiente con la opción elegida **5.** Elegir el o los niños de los cuales se desea hacer un reporte. / **6.** Elaborar reporte.
Flujos Alternos	**Actor** / **Acciones del Sistema** **FA1.** Autentificar sus datos de usuari@ registrado para tener acceso al sistema. / **FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.
Flujos de Excepción	**Actor** / **Acciones del Sistema** **FE1.** Autentificar sus datos de usuari@ registrado para tener / **FE2.** Si el acceso a una sesión no esta disponible por alguna razón

	acceso al sistema.	distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.1.2**
Nombre	**Reportar expediente del niñ@ (Anomalías, faltas, enfermedades)**
Actores	Director@, Educador@, Psicolog@
Descripción	Proceso para obtener un reporte sobre aspectos de la vida del niñ@ como son: anomalías presentadas antes y durante su estancia en la guardería, faltas a ella y registro de enfermedades que le aquejaron.
Observaciones	
Diagrama	
Precondiciones	- Tener registrado en la Base de Datos al niñ@ del cual se realizará el reporte. - Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	

Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar expediente del niño”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir el o los niños de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

Flujos Alternos	

Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.

Flujos de Excepción	

Actor	Acciones del Sistema

	FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FE2.** Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.1.3**
Nombre	**Reportar niñ@s por delegación (domicilio)**
Actores	Director@, Área administrativa de delegación ISSSTE
Descripción	Proceso para obtener un reporte sobre la relación existente entre l@s niñ@s y la delegación a la que pertenece su domicilio registrado en la guardería.
Observaciones	
Diagrama	
Precondiciones	**- Tener acceso a la Base de Datos.** **- Tener los permisos de sistema necesarios.**
Postcondiciones	
Flujo principal	

Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar niñ@s por delegación”	**4.**Mostrar el menú correspondiente con la opción elegida
5. Elegir el o l@s niñ@s de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

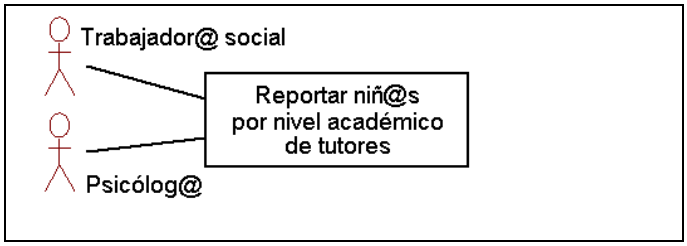
Flujos Alternos

Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.

Flujos de Excepción

Actor	Acciones del Sistema
FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FE2.** Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de

	autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.1.4**
Nombre	**Reportar niños por nivel académico de tutores**
Actores	Trabajador@ social, Psicólog@
Descripción	Proceso para obtener un reporte sobre la relación existente entre l@s niñ@s y el nivel académico de sus padres.
Observaciones	
Diagrama	
Precondiciones	**- Tener registrado en la Base de Datos al niñ@ del cual se realizará el reporte.** **- Tener registrados los datos de sus tutores.** **- Tener acceso a la Base de Datos.** **- Tener los permisos de sistema necesarios.**
Postcondiciones	
Flujo principal	

Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar niñ@s por nivel académico de padres”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir el o l@s niñ@s de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

Flujos Alternos	

Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.

Flujos de Excepción	

Actor	Acciones del Sistema

	FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.2.1**
Nombre	**Reportar datos de tutores**
Actores	Trabajador@ social
Descripción	Proceso para obtener un reporte que contenga información relevante sobre los tutores registrados en la guardería.
Observaciones	
Diagrama	
Precondiciones	**- Tener registrado al tutor del cual se realizara el reporte.** **- Tener acceso a la Base de Datos.** **- Tener los permisos de sistema necesarios.**
Postcondiciones	
Flujo principal	

Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar datos de tutores”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir el o los tutores de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

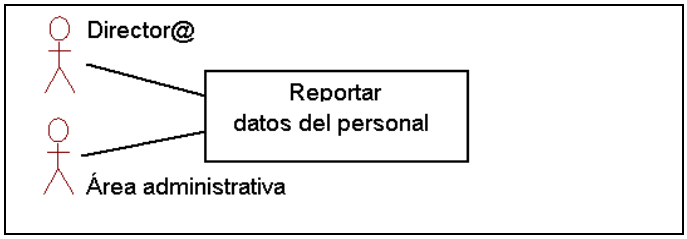
Flujos Alternos	

Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión por clave incorrecta.

Flujos de Excepción	

Actor	Acciones del Sistema
FE1. Autentificar sus datos de	**FE2.** Si el acceso a una sesión no

	usuari@ registrado para tener acceso al sistema.	esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.2.2**
Nombre	**Reportar datos del personal de delegación del ISSSTE**
Actores	Director@, Área administrativa
Descripción	Proceso para obtener un reporte que contenga datos relevantes sobre el personal que labora en la guardería.
Observaciones	
Diagrama	
Precondiciones	**- Tener registrado en la Base de Datos al elemento del personal del cual se realizará el reporte.** **- Tener acceso a la Base de Datos.** **- Tener los permisos de sistema necesarios.**
Postcondiciones	
Flujo principal	

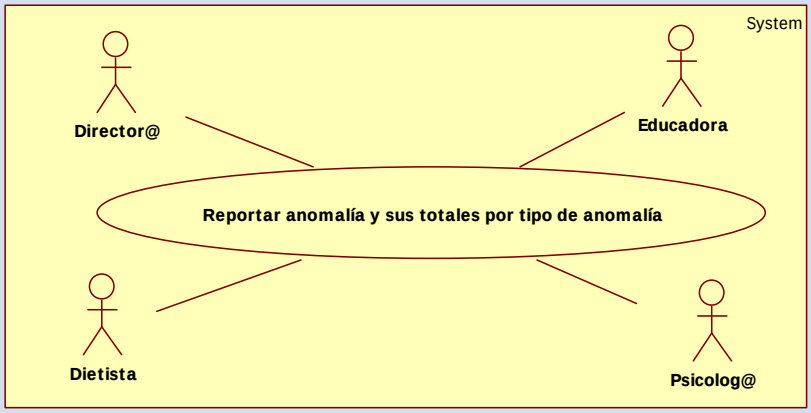
Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar datos del personal”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir el o elementos del personal de la guardería de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

Flujos Alternos	

Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.

Flujos de Excepción	

Actor	Acciones del Sistema
FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FE2.** Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.2.3**
Nombre	**Reportar anomalías y sus totales por tipo de anomalía**
Actores	Director@, Educadora, Psicolog@, Dietist@
Propósito	Generar un reporte de las anomalías del infante y sus totales por tipo de anomalía.
Descripción	Proceso para obtener un reporte sobre anomalías y sus totales por tipo del infante para continuar tratamiento durante su estancia en la guardería.
Observaciones	
Diagrama	
Salidas	
Precondiciones	- Tener registrado en la Base de Datos al infante del cual se realizará el reporte. - Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	

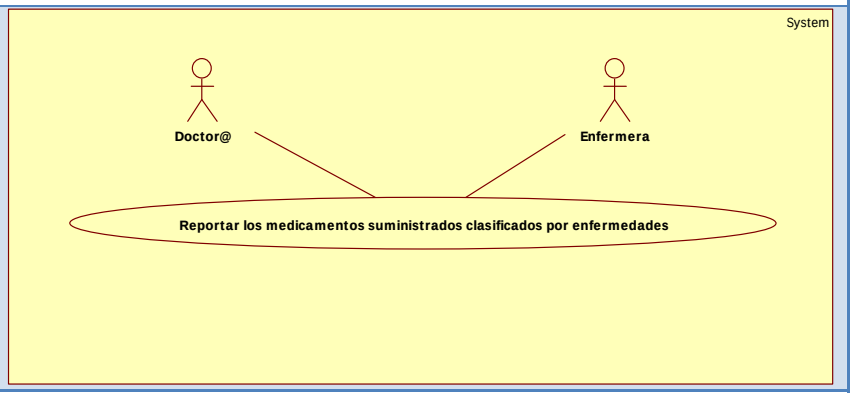
Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar anomalías y sus totales por tipo de anomalía”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir el o los infantes de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

Flujos Alternos

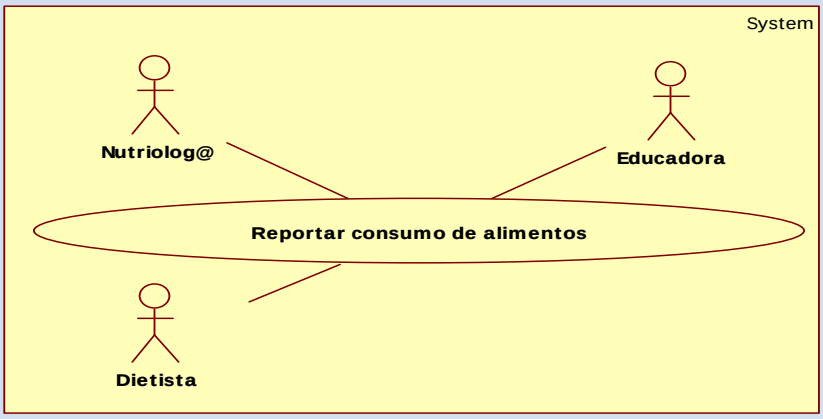
Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.

Flujos de Excepción

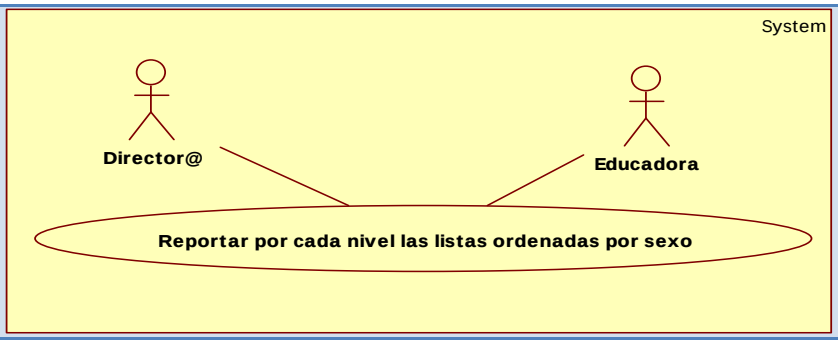
	<table><tr><th>Actor</th><th>Acciones del Sistema</th></tr><tr><td>FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>FE2. Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.</td></tr></table>

ID	**4.2.4**
Nombre	**Reportar los medicamentos suministrados clasificados por enfermedades**
Actores	Doctor@, enfermera
Propósito	Generar un reporte de los medicamentos suministrados clasificados por enfermedades del infante.
Descripción	Proceso para obtener un reporte sobre los medicamentos suministrados clasificados por enfermedades del infante para continuar tratamiento durante su estancia en la guardería.
Observaciones	
Diagrama	
Salidas	
Precondiciones	- Tener registrado en la Base de Datos al infante del cual se realizará el reporte. - Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	<table><tr><th>Actor</th><th>Acciones del Sistema</th></tr><tr><td>1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>2. Si la autenticación es correcta: - Iniciar sesión.</td></tr><tr><td>3. Elegir la opción: “Reportar los medicamentos suministrados clasificados por enfermedades del infante”</td><td>4. Mostrar el menú correspondiente con la opción elegida</td></tr><tr><td>5. Elegir el o los infantes de los cuales se desea hacer un reporte.</td><td>6. Elaborar reporte.</td></tr></table>
Flujos Alternos	<table><tr><th>Actor</th><th>Acciones del Sistema</th></tr><tr><td>FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>FA2. Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.</td></tr></table>
Flujos de Excepción	

	<table><tr><th>Actor</th><th>Acciones del Sistema</th></tr><tr><td>FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>FE2. Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.</td></tr></table>

ID	**4.2.5**
Nombre	**Reportar consumo de alimentos**
Actores	Dietista, educadora, nutriólog@
Propósito	Generar un reporte del consumo de alimentos del infante.
Descripción	Proceso para obtener un reporte sobre el consumo de alimentos del infante por una semana, mes o un año, durante su estancia en la guardería.
Observaciones	
Diagrama	
Salidas	
Precondiciones	- Tener registrado en la Base de Datos al infante del cual se realizará el reporte. - Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	<table><tr><th>Actor</th><th>Acciones del Sistema</th></tr><tr><td>1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>2. Si la autenticación es correcta: - Iniciar sesión.</td></tr><tr><td>3. Elegir la opción: “Reportar el consumo de alimentos del infante”</td><td>4. Mostrar el menú correspondiente con la opción elegida</td></tr><tr><td>5. Elegir el o los niños de los cuales se desea hacer un reporte.</td><td>6. Elaborar reporte.</td></tr></table>
Flujos Alternos	<table><tr><th>Actor</th><th>Acciones del Sistema</th></tr><tr><td>FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>FA2. Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.</td></tr></table>
Flujos de Excepción	

Actor	Acciones del Sistema
FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FE2.** Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.

ID	**4.2.6**
Nombre	**Reportar por cada nivel las listas ordenadas por sexo**
Actores	Director@, educadora
Propósito	Generar un reporte por cada nivel y generar las listas ordenadas por sexo de los infantes de la guardería.
Descripción	Proceso para obtener un reporte por cada nivel y generar las listas ordenadas por sexo de los infantes.
Observaciones	
Diagrama	
Salidas	
Precondiciones	- Tener registrado en la Base de Datos al infante del cual se realizará el reporte. - Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	

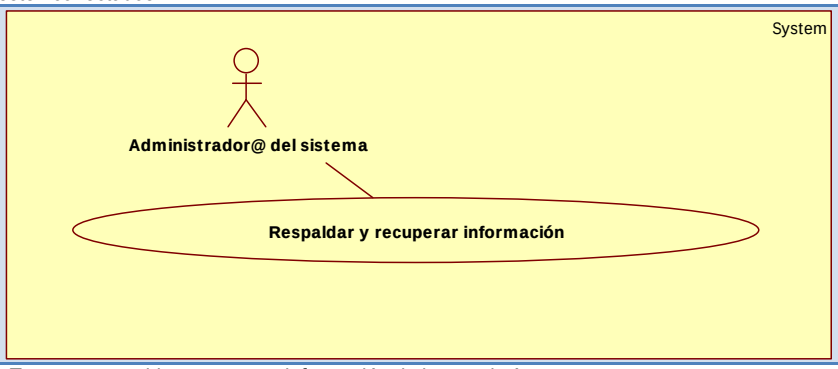
Actor	Acciones del Sistema
1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Reportar por cada nivel y generar las listas ordenadas por sexo de los infantes de la guardería”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir el o los niños de los cuales se desea hacer un reporte.	**6.** Elaborar reporte.

Flujos Alternos

Actor	Acciones del Sistema
FA1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta: - Despliega un mensaje indicando que es imposible entrar a una sesión.

<table>
<tr><td>Flujos de Excepción</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FE1. Autentificar sus datos de usuari@ registrado para tener acceso al sistema.</td><td>FE2. Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos:

- Despliega un mensaje pidiendo al usuari@ que intente de nuevo en unos minutos o que pida ayuda a Soporte Técnico.</td></tr>
</table>
</td></tr>
</table>

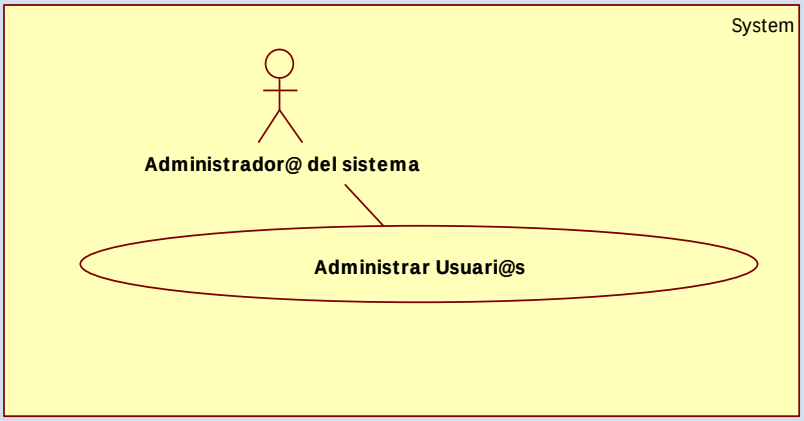
<table>
<tr><td>ID</td><td>5</td></tr>
<tr><td>Nombre</td><td>Respaldar y recuperar información</td></tr>
<tr><td>Actores</td><td>Administrador del sistema</td></tr>
<tr><td>Propósito</td><td>Generar un respaldo y si es necesario recuperación de información de la guardería.</td></tr>
<tr><td>Descripción</td><td>Proceso para realizar un respaldo y si es necesario recuperación de información de la guardería.</td></tr>
<tr><td>Observaciones</td><td>El respaldo se debe hacer en la noche o al fin de actividades diarias y que los usuarios no estén conectados.</td></tr>
<tr><td>Diagrama</td><td> 
</td></tr>
<tr><td>Salidas</td><td>- Tener un respaldo y recuperar información de la guardería</td></tr>
<tr><td>Precondiciones</td><td>- Tener registrado en la Base de Datos a todos los infantes y su respectiva información así como toda la información que se maneja en la guardería.
- Tener acceso a la Base de Datos.
- Tener los permisos de sistema necesarios y verificar que ningún usuario esté conectado.</td></tr>
<tr><td>Postcondiciones</td><td></td></tr>
<tr><td>Flujo principal</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>1. Autentificar sus datos del administrador del sistema registrado para tener acceso al sistema.</td><td>2. Si la autenticación es correcta:

- Iniciar sesión.</td></tr>
<tr><td>3. Elegir la opción: “respaldar información o recuperar información”</td><td>4. Mostrar el menú correspondiente con la opción elegida</td></tr>
<tr><td>5. Elegir la información que se va a respaldar o recuperar.</td><td>6. Elaborar respaldo o recuperación.</td></tr>
</table>
</td></tr>
<tr><td>Flujos Alternos</td><td>
<table>
<tr><th>Actor</th><th>Acciones del Sistema</th></tr>
<tr><td>FA1. Autentificar sus datos del administrador@ del sistema para tener acceso al sistema.</td><td>FA2. Si la autenticación es incorrecta:

- Despliega un mensaje indicando que es imposible entrar a una sesión.</td></tr>
</table>
</td></tr>
</table>

Flujos de Excepción	

Actor	Acciones del Sistema
FE1. Autentificar sus datos del administrador@ del sistema para tener acceso al sistema.	**FE2.** Si el acceso a una sesión no está disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al administrador@ del sistema que intente de nuevo.

ID	**6**
Nombre	**Administrar Usuari@s**
Actores	Administrad@r del sistema
Propósito	Administrar a los usuari@s del sistema.
Descripción	Proceso para administrar a los usuari@s del sistema, clasificados por permiso otorgados desde la creación de la cuenta de usuari@.
Observaciones	Los permisos son clasificados por: -A: Alta -B: Baja -C: Consulta - M: Modificación
Diagrama	
Salidas	
Precondiciones	- Tener registrado a los usuari@s. - Tener acceso a la Base de Datos. - Tener los permisos de sistema necesarios.
Postcondiciones	
Flujo principal	

Actor	Acciones del Sistema
1. Autentificar sus datos del administrador del sistema registrado para tener acceso al sistema.	**2.** Si la autenticación es correcta: - Iniciar sesión.
3. Elegir la opción: “Administrar usuari@s”	**4.** Mostrar el menú correspondiente con la opción elegida
5. Elegir a los usuari@s que se van a administrar	**6.** Elaborar la gestión de los usuari@s

Flujos Alternos	

Actor	Acciones del Sistema
FA1. Autentificar sus datos del administrador del sistema para tener acceso al sistema.	**FA2.** Si la autenticación es incorrecta:

		- Despliega un mensaje indicando que es imposible entrar a una sesión.
Flujos de Excepción	Actor	Acciones del Sistema
	FE1. Autentificar sus datos del administrador del sistema para tener acceso al sistema.	**FE2.** Si el acceso a una sesión no esta disponible por alguna razón distinta al ingresar datos de autenticación incorrectos: - Despliega un mensaje pidiendo al administrador del sistema que intente de nuevo.

1.2. Diseño de la Base de Datos (2)

Una vez analizadas las entidades y atributos procedemos con el diseño de la Base de Datos, describiendo cada tabla, que es el mapeo del Diagrama Entidad Relación, en el se indican llaves primarias, foráneas y su tabla de referencia.

(Muestra de 2 tablas):

Tabla: SOPAS

Nombre atributo	nulo o no nulo		tabla de referencia
id_sopa	NOT NULL	PK	
timestamp	NOT NULL		
descripción	NOT NULL		
status_bl	NULL		

Tabla:PADRES_TUTORES

Nombre Atributo	Nulo o no nulo		tabla de referencia
curp	NOT NULL	PK	
timestamp	NOT NULL		
nombre	NOT NULL		
ap_paterno	NOT NULL		
calle_num_casa	NOT NULL		
colonia_casa	NOT NULL		
cp_casa	NOT NULL		
id_municipio_casa	NOT NULL	FK_1	MUNICIPIOS_DELEGACIONES
id_estado_casa	NOT NULL	FK_1	MUNICIPIOS_DELEGACIONES
calle_num_trab	NOT NULL		
colonia_trab	NOT NULL		
cp_trab	NOT NULL		

id_municipio_trabajo	NOT NULL	FK_2	MUNICIPIOS_DELEGACIONES
id_estado_trabajo	NOT NULL	FK_2	MUNICIPIOS_DELEGACIONES
id_parentesco	NOT NUL	FK_3	PARENTESCOS
id_dependencia	NOT NULL	FK_4	DEPENDENCIAS
nivel_academico	NOT NUL		
tel_trab	NOT NULL		
ap_mat	NULL		
tel_cel	NULL		
tel_casa	NULL		
id_pais	NULL	FK_5	PAISES
status_bl	NULL		

CALCULO DE TAMAÑO DE LA BASE DE DATOS

Ahora realizamos el cálculo de tamaño de la Base de Datos, tabla por tabla como se muestra a continuación:

Tamaño del bloque
Tamaño de bloque = 8 KB
= 8192 bytes

Bytes libres del bloque = Tamaño de bloque – 360 bytes (valor constante de header)
= 8192 - 360
= 7832 bytes libres del bloque

Cálculo del tamaño inicial (en bloques)
= (# registros * longitud del registro) / bytes libres del bloque

INITIAL (Tamaño inicial)

= (# registros * (SUMA LONGITUD DEL REGISTRO + SUMA BYTE DE SEPARACION + ROWDIR)) / BYTES LIBRES DEL BLOQUE
= bloques
= X * 8Kb

PCTFREE

= (100 * SUMA PROMEDIOS NULOS) / SUMA LONGITUD

PCTUSED

$$PCTUSED = 100 - PCTFREE - \frac{(SUMA\ LONGITUD\ DEL\ REGISTRO + SUMA\ BYTE\ DE\ SEPARACIÓN + ROWDIR)*100)}{BYTES\ LIBRES\ DEL\ BLOQUE}$$

MAXTRANS

= (BYTES LIBRES DEL BLOQUE) / (SUMA LONGITUD DEL REGISTRO + SUMA BYTE DE SEPARACION + ROWDIR)

PCTINCREASE

= 0 (RECOMENDADO para evitar que se descontrolen los tamaños de los segmentos y para posible desfragmentación se coloca en 1)

NEXT

100% del tamaño del parámetro INITIAL (Indica el tamaño que tendrá la próxima extensión que se cree cuando no quede más sitio en las extensiones que ya tiene asignadas el segmento)

CONSUMO_COMIDAS

Columna	Tipo dato	Longitud máxima (considerando promedio nulos)			Longitud + byte de separación		
		X	+ Prom. nulos	=	X	+ Byte	=
fecha	DATE (NOT NULL)	**7**		**7**	**7**	**1**	**8**
curp_infante	CHAR(18) (NOT NULL)	**18**		**18**	**18**	**1**	**19**
nivel	NUMBER(2) (NOT NULL)	**ceil(2/2+1)**		**2**	**2**	**1**	**3**
timestamp	DATE (NOT NULL)	**7**		**7**	**7**	**1**	**8**
status_jugo	CHAR(2) (NULL)	**2**		**2**	**2**	**1**	**3**
status_fruta	CHAR(2) (NULL)	**2**		**2**	**2**	**1**	**3**
status_cereal	CHAR(2) (NULL)	**2**		**2**	**2**	**1**	**3**
status_sopa	CHAR(2) (NULL)	**2**		**2**	**2**	**1**	**3**
status_alimento	CHAR(2) (NULL)	**2**		**2**	**2**	**1**	**3**
status_verdura_ensalada	CHAR(2) (NULL)	**2**		**2**	**2**	**1**	**3**
Id_jugo	SMALLINT (NULL)	**2**		**2**	**2**	**1**	**3**
Id_fruta	SMALLINT (NULL)	**2**		**2**	**2**	**1**	**3**
id_alimento	SMALLINT (NULL)	**2**		**2**	**2**	**1**	**3**
id_cereal	SMALLINT (NULL)	**2**		**2**	**2**	**1**	**3**
id_sopa	SMALLINT (NULL)	**2**		**2**	**2**	**1**	**3**
id_verdura_ensalada	SMALLINT (NULL)	**2**		**2**	**2**	**1**	**3**
Σ			**0**		**58**	**16**	**74**
Σ+ rowdir(5)							**79**

Tabla:		**Indice:**	
INITIAL	19370,1706	INITIAL	10543,2574
NEXT	100	NEXT	100
PCTINCREAS	0	PCTINCREAS	0
PCTFREE	0	PCTFREE	0
PCTUSED	98,9913177	PCTUSED	100
INITRANS	1	INITRANS	2
MAXTRANS	99,1392405	MAXTRANS	182,139535
MINEXTENTS	1	MINEXTENTS	1
MAXEXTENT	2	MAXEXTENT	2

Registros NOT NUL	
Prom. nulos	0
Longitud (Long)	34
Byte de separación (BS)	4
Long + BS + ROWDIR	43

CONSUMO_DESAYUNOS

Columna	Tipo dato	Longitud máxima (considerando promedio nulos)			Longitud + byte de separación		
		X	+ Prom. nulos	=	X	+ Byte	=
fecha	DATE (NOT NULL)	7		7	7	1	8
curp_infante	CHAR(18) (NOT NULL)	18		18	18	1	19
nivel	NUMBER(2) (NOT NULL)	ceil(2/2+1)		2	2	1	3
timestamp	DATE (NOT NULL)	7		7	7	1	8
status_jugo	CHAR(2) (NULL)	2		2	2	1	3
status_fruta	CHAR(2) (NULL)	2		2	2	1	3
status_cereal	CHAR(2) (NULL)	2		2	2	1	3
status_alimento	CHAR(2) (NULL)	2		2	2	1	3
status_leche	CHAR(2) (NULL)	2		2	2	1	3
id_jugo	SMALLINT (NULL)	2		2	2	1	3
id_fruta	SMALLINT (NULL)	2		2	2	1	3
id_alimento	SMALLINT (NULL)	2		2	2	1	3
id_cereal	SMALLINT (NULL)	2		2	2	1	3
id_leche	SMALLINT (NULL)	2		2	2	1	3
Σ			0		54	14	68
Σ+ rowdir(5)							73

Tabla:		Indice:	
INITIAL	17899,0184	INITIAL	124757,099
NEXT	100	NEXT	100
PCTINCREAS	0	PCTINCREAS	0
PCTFREE	0	PCTFREE	0
PCTUSED	100	PCTUSED	100
INITRANS	1	INITRANS	2
MAXTRANS	107,287671	MAXTRANS	182,139535
MINEXTENTS	1	MINEXTENTS	1
MAXEXTENT	2	MAXEXTENT	2

Registros NOT NUL	
Prom. nulos	0
Longitud (Long)	34
Byte de separación (BS)	4
Long + BS + ROWDIR	43

INFANTES

		X	+ Prom. nulos	=	X	+ Byte	=
curp_infante	CHAR(18) (NOT NULL)	**18**		**18**	**18**	**1**	**19**
timestamp	DATE (NOT NULL)	**7**		**7**	**7**	**1**	**8**
nombre	VARCHAR2(70) (NOT NULL)	**40**	**30**	**70**	**70**	**1**	**71**
ap_pat	VARCHAR2(70) (NOT NULL)	**15**	**55**	**70**	**70**	**1**	**71**
fecha_nacimiento	DATE (NOT NULL)	**7**		**7**	**7**	**1**	**8**
peso_nacimiento	NUMBER(2,1) (NOT NULL)	**ceil(2/2+1)**		**2**	**2**	**1**	**3**
estatura_nacimiento	NUMBER(2,2) (NOT NULL)	**ceil(2/2+1)**		**2**	**2**	**1**	**3**
calle_num	VARCHAR2(100) (NOT NULL)	**50**	**50**	**100**	**100**	**1**	**101**
colonia	VARCHAR2(120) (NOT NULL)	**60**	**60**	**120**	**120**	**1**	**121**
cp	NUMBER(5) (NOT NULL)	**ceil(5/2+1)**		**4**	**4**	**1**	**5**
id_municipio	NUMBER(3) (NOT NULL)	**ceil(3/2+1)**		**3**	**3**	**1**	**4**
id_estado	NUMBER(2) (NOT NULL)	**ceil(2/2+1)**		**2**	**2**	**1**	**3**
num_estancia	NUMBER(3) (NOT NULL)	**ceil(3/2+1)**		**3**	**3**	**1**	**4**
sexo	CHAR(1) (NOT NULL)	**1**		**1**	**1**	**1**	**2**
fecha_ingreso	DATE (NOT NULL)	**7**		**7**	**7**	**1**	**8**
leche_materna	CHAR(1) (NOT NULL)	**1**		**1**	**1**	**1**	**2**
id_pais_vive	NUMBER(3) (NOT NULL)	**ceil(3/2+1)**		**3**	**3**	**1**	**4**
ap_mat	VARCHAR2(70) (NULL)	**15**	**55**	**70**	**70**	**1**	**71**
fecha_egreso	DATE (NULL)	**7**		**7**	**7**	**1**	**8**
status_bl	CHAR(1) (NULL)	**1**		**1**	**1**	**1**	**2**
id_pais_nacionalidad	NUMBER(3) (NULL)	**ceil(3/2+1)**		**3**	**3**	**1**	**4**
curp_infante_hermano	CHAR(18) (NULL)	**18**		**18**	**18**	**1**	**19**
Σ			**250**		**519**	**22**	**541**
Σ+ rowdir(5)							**546**

Tabla:		**Indice:**	
INITIAL	1003,88151	INITIAL	812,665986
NEXT	100	NEXT	100
PCTINCREAS	0	PCTINCREAS	0
PCTFREE	48,1695568	PCTFREE	46,4285714
PCTUSED	44,8590438	PCTUSED	47,9279148
INITRANS	1	INITRANS	2
MAXTRANS	14,3443223	MAXTRANS	17,719457
MINEXTENTS	1	MINEXTENTS	1
MAXEXTENT	2	MAXEXTENT	2

Registros NOT NUL	
Prom. nulos	195
Longitud (Long)	420
Byte de separación (BS)	17
Long + BS + ROWDIR	442

INFANTES_NIVEL_PROFESORA

Columna	Tipo dato	Longitud máxima (considerando promedio nulos)			Longitud + byte de separación		
		X	+ Prom. nulos	=	X	+ Byte	=
curp_infante	CHAR(18) (NOT NULL)	18		18	18	1	19
nivel	NUMBER(2) (NOT NULL)	ceil(2/2+1)		2	2	1	3
curp_empleado	CHAR(18) (NOT NULL)	18		18	18	1	19
titular	CHAR(1) (NOT NULL)	1		1	1	1	2
timestamp	DATE (NOT	7		7	7	1	8
Σ			0		46	5	51
Σ+ rowdir(5)							56

Tabla:		Indice:	
INITIAL	102,962206	INITIAL	68,0286006
NEXT	100	NEXT	100
PCTINCREAS	0	PCTINCREAS	0
PCTFREE	0	PCTFREE	0
PCTUSED	99,2849847	PCTUSED	99,5275792
INITRANS	1	INITRANS	2
MAXTRANS	139,857143	MAXTRANS	211,675676
MINEXTENTS	1	MINEXTENTS	1
MAXEXTENT	2	MAXEXTENT	2

Registros NOT NUL	
Prom. nulos	0
Longitud (Long)	28
Byte de separación (BS)	4
Long + BS + ROWDIR	37

INFANTES_NIVEL

Columna	Tipo dato	Longitud máxima (considerando promedio nulos)			Longitud + byte de separación		
		X	+ Prom. nulos	=	X	+ Byte	=
curp_infante	CHAR(18) (NOT NULL)	18		18	18	1	19
nivel	NUMBER(2) (NOT NULL)	ceil(2/2+1)		2	2	1	3
timestamp	DATE (NOT NULL)	7		7	7	1	8
hr_salida	DATE (NOT NULL)	7		7	7	1	8
peso	NUMBER(3,1) (NOT NULL)	ceil(3/2+1)		3	3	1	4
estatura	NUMBER(3,1) (NOT NULL)	ceil(3/2+1)		3	3	1	4
comentario_final	VARCHAR2(100) (NULL)	50	50	100	100	1	101
Σ			50		140	7	147
Σ+ rowdir(5)							152

Tabla:		Indice:	
INITIAL	1391,28907	INITIAL	466,814096
NEXT	100	NEXT	100
PCTINCREAS	0	PCTINCREAS	0
PCTFREE	35,7142857	PCTFREE	0
PCTUSED	62,3449584	PCTUSED	99,3488253
INITRANS	1	INITRANS	2
MAXTRANS	51,5263158	MAXTRANS	153,568627
MINEXTENTS	1	MINEXTENTS	1
MAXEXTENT	2	MAXEXTENT	2

Registros NOT NUL	
Prom. nulos	0
Longitud (Long)	40
Byte de separación (BS)	6
Long + BS + ROWDIR	51

EXPEDIENTES_MEDICO_GUARDERIA

Columna	Tipo dato	Longitud máxima (considerando promedio nulos)			Longitud + byte de separación		
		X	+ Prom. nulos	=	X	+ Byte	=
curp_infante	CHAR(18) (NOT NULL)	18		18	18	1	19
num_secuencial_expediente	NUMBER(5 (NOT NULL)	ceil(5/2+1)		4	4	1	5
timestamp	DATE (NOT	7		7	7	1	8
curp_empleado	CHAR(18) (NOT NULL)	18		18	18	1	19
fecha	DATE (NOT	7		7	7	1	8
num_enfermedad	NUMBER(3) (NULL)	ceil(3/2+1)		3	3	1	4
comentario_constancia	VARCHAR2(200) (NULL)	100	100	200	200	1	201
dias_permiso_constancia	SMALLINT (NULL)	2		2	2	1	3
id_lugar_atencion	NUMBER(2) (NULL)	ceil(2/2+1)		2	2	1	3
Σ			100		261	9	270
Σ+ rowdir(5)							275

Tabla:		Indice:	
INITIAL	6601,40449	INITIAL	1536,32686
NEXT	100	NEXT	100
PCTINCREAS	0	PCTINCREAS	0
PCTFREE	38,3141762	PCTFREE	0
PCTUSED	58,1745878	PCTUSED	99,1828396
INITRANS	1	INITRANS	2
MAXTRANS	28,48	MAXTRANS	122,375
MINEXTENTS	1	MINEXTENTS	1
MAXEXTENT	2	MAXEXTENT	2

Registros NOT NUL	
Prom. nulos	0
Longitud (Long)	54
Byte de separación (BS)	5
Long + BS + ROWDIR	64

Tablas con pocos registros y poca actividad.

**MUNICIPIOS_DELEGACIONES		64	**0.064**
**TELEFONOS		64	**0.064**
**PAISES		64	**0.064**
**DELEGACIONES		64	**0.064**
**ESTADOS		64	**0.064**
**MEDICAMENTOS		64	**0.064**
**ESTANCIAS		64	**0.064**
INFANTES_NIVEL	<100	64	**0.064**
EMPLEADOS	<100	64	**0.064**
SOPAS	<100	64	**0.064**
VERDURAS_ENSALADAS	<100	64	**0.064**
CEREALES_LEGUMINOSAS	<100	64	**0.064**
VARIACIONES_LECHE	<100	64	**0.064**
JUGOS_AGUAS	<100	64	**0.064**
FRUTAS_YOGOURTH_POSTRES	<100	64	**0.064**
ALIMENTOS_FUERTES	<100	64	**0.064**
TIPOS_ANOMALIA	<100	64	**0.064**
ACTIVIDADES	<100	64	**0.064**
NIVELES	<100	64	**0.064**
HORARIOS_MEDICAMENTO	<100	64	**0.064**
TIPOS_NIVEL	<100	64	**0.064**
TIPOS_VACUNA	<100	64	**0.064**
ENFERMEDADES	<100	64	**0.064**
PARENTESCOS	<100	64	**0.064**
DEPENDENCIAS	<100	64	**0.064**
LUGASRES_ATENCION	<100	64	**0.064**
TIPO_EMPLEADO	<100	64	**0.064**
DOSIS	<100	64	**0.064**
TOTAL			1.792

Índices.

Tablas	# de registros	KB	MB
CONSUMO_COMIDAS	240043	10,543	**10.54325741**
CONSUMO_DESAYUNOS	240043	10,543	**10.54325741**
HORARIOS_SALIDA_INFANTE	2840400	147,968	**147.9677222**
HORARIOS_SALIDA_TUTOR	1800	94	**0.093769152**
PADRES_INFANTE	3600	188	**0.187538304**
VACUNAS	41400	1,818	**1.818386108**
ANTECEDENTES_ENFERMEDAD	21600	838	**0.838406537**
DETALLES_RECETA	23501	2,449	**2.44852094**
RECETAS	23501	1,272	**1.272270684**
EXPEDIENTES_MEDICO_GUARDERIA	23501	1,536	**1.536326864**
PADRES_TUTORES	5400	4,517	**4.517466803**
INFANTES	1800	813	**0.812665986**
INFANTES_NIVEL_PROFESORA	1800	68	**0.068028601**
ANOMALIAS	102355	5,855	**5.854831461**
INFANTES_NIVEL	89061	4,640	**4.639541369**
TOTAL			193.1419898

Tablespaces.

Tablas	MB	MB + 30%
CATALOGOS	1.792	2.3296
INDICES	193.1419898	251.0845867
VOLATILES	**274.7442654**	357.167545
TOTAL		610.5817318

Cálculo total de la Base de Datos

Tablas	# de registros	KB	MB
CONSUMO_COMIDAS	240,043	19,370	19.37017058
CONSUMO_DESAYUNOS	240,043	17,899	17.89901839
HORARIOS_SALIDA_INFANTE	2,840,400	147,968	147.9677222
HORARIOS_SALIDA_TUTOR	1,800	97	0.097446374
PADRES_INFANTE	3,600	188	0.187538304
VACUNAS	41,400	6,089	6.08947906
ANTECEDENTES_ENFERMEDAD	21,600	3,574	3.574259448
DETALLES_RECETA	23,501	2,449	2.44852094
RECETAS	23,501	1,272	1.272270684
EXPEDIENTES_MEDICO_GUARDERIA	23,501	6,601	6.601404494
PADRES_TUTORES	5,400	4,727	4.727068437
INFANTES	1,800	1,004	1.003881512
INFANTES_NIVEL_PROFESORA	1,800	103	0.102962206
ANOMALIAS	102,355	26,138	26.13764045
INFANTES_NIVEL	8,961	1,391	1.39128907
		DATOS	**211.3417426**
		*** Δ = 30%**	**63.40252278**
		TOTAL	**274.7442654**
Δ **ANUAL:** CONSUMO_COMIDAS	2,183	176	0.176156282
Δ **ANUAL:** CONSUMO_DESAYUNOS	2,183	163	0.162777324
Δ **ANUAL:** HORARIOS_SALIDA_INFANTE	94,680	7,060	7.059897855
Δ **ANUAL:** HORARIOS_SALIDA_TUTOR	1800	134	0.13421859
Δ **ANUAL:** PADRES_INFANTE	720	54	0.053687436
Δ **ANUAL:** VACUNAS	8280	617	0.617405516
Δ **ANUAL:** ANTECEDENTES_ENFERMEDAD	4320	322	0.322124617
Δ **ANUAL:** DETALLES_RECETA	20842	1,554	1.554102145
Δ **ANUAL:** RECETAS	20842	1,554	1.554102145
Δ **ANUAL:** EXPEDIENTES_MEDICO_GUARDERIA	20842	1,554	1.554102145
Δ **ANUAL:** PADRES_TUTORES	1080	81	0.080531154
Δ **ANUAL:** INFANTES	360	27	0.026843718
Δ **ANUAL:** INFANTES_NIVEL_PROFESORA	1800	134	0.13421859
Δ **ANUAL:** ANOMALIAS	13716	1,023	1.022745659
Δ **ANUAL:** INFANTES_NIVEL	1524	114	0.113638407
		Δ ANUAL	**14.56655158**
		Durante 5 años	**72.83275792**
		Sin problemas 5 años	**347.5770233**
		****LOG (Δ X .30)**	**104.273107**
Tablas con pocos registros y poca actividad			**1.792**
Índices			**196.1419173**
Tablespaces			**404.6449974**
		TOTAL GLOBAL	**1054.429045**

* La página utiliza espacios que identifican a quien le pertenece y también para relacionar en la cola doblemente ligada, por ello es 30% de más.

** LOG entre 15% y 30% de Δ (30% es altamente transaccional)

1.3 Construcción de la Base de Datos

Creación de la instancia:

A continuación describimos los pasos para crear una instancia en nuestro manejador de Base de Datos Oracle.

#(Abrir ventana MSDOS -> en ejecutar cmd)
#1 CREAR LOS SIGUIENTES DIRECTORIOS, LA BASE DE DATOS SE LLAMA ESTANCIA
#puede ser que algún directorio ya este creado, por ejemplo dbs
mkdir C:\oracle\product\10.2.0\admin\ESTANCIA\adump
mkdir C:\oracle\product\10.2.0\admin\ESTANCIA\bdump
mkdir C:\oracle\product\10.2.0\admin\ESTANCIA\cdump
mkdir C:\oracle\product\10.2.0\admin\ESTANCIA\dpdump
mkdir C:\oracle\product\10.2.0\admin\ESTANCIA\pfile
mkdir C:\oracle\product\10.2.0\admin\ESTANCIA\udump
mkdir C:\oracle\product\10.2.0\db_1\cfgtoollogs\dbca\ESTANCIA
mkdir C:\oracle\product\10.2.0\db_1\dbs ...ya existe
mkdir C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA
mkdir C:\oracle\product\10.2.0\flash_recovery_area

#La siguiente línea solo sera necesario cuando vuelven a repetir el proceso
C:\oracle\product\10.2.0\db_1\bin\oradim.exe -delete -sid ESTANCIA

#2 Crear initESTANCIA.ora en C:\oracle\product\10.2.0\db_1\dbs\initESTANCIA.ora
#(recuerden que es una copia de init.ora (default),
#Pero consideren el codigo siguiente, para crear dicho archivo en bloc de notas
#es importante que la extensión del archivo sea .ora

```
###############################################################################
# Copyright (c) 1991, 2001, 2005 by Oracle Corporation
###############################################################################

###########################################
# Archive
###########################################
log_archive_format=ARC%S_%R.%T
log_archive_dest=C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA

###########################################
# Cache and I\O
###########################################
db_block_size=8192
db_file_multiblock_read_count=16

###########################################
# Job Queues
###########################################
job_queue_processes=10

###########################################
# File Configuration
###########################################
control_files=("C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\control01.ctl",
"C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\control02.ctl",
"C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\control03.ctl")
db_recovery_file_dest=C:\oracle\product\10.2.0\flash_recovery_area
db_recovery_file_dest_size=134217728

###########################################
# Cursors and Library Cache
###########################################
open_cursors=300

###########################################
# Diagnostics and Statistics
###########################################
background_dump_dest=C:\oracle\product\10.2.0\admin\ESTANCIA\bdump
core_dump_dest=C:\oracle\product\10.2.0\admin\ESTANCIA\cdump
```

```
user_dump_dest=C:\oracle\product\10.2.0\admin\ESTANCIA\udump
 ##############################################
# Database Identification
##############################################
db_domain=""
db_name=ESTANCIA
 ##############################################
# SGA Memory
##############################################
sga_target=167772160
 ##############################################
# Processes and Sessions
##############################################
processes=150
 ##############################################
# System Managed Undo and Rollback Segments
##############################################
undo_management=AUTO
undo_tablespace=UNDO
 ##############################################
# Security and Auditing
##############################################
audit_file_dest=C:\oracle\product\10.2.0\admin\ESTANCIA\adump
remote_login_passwordfile=EXCLUSIVE
 ##############################################
# Shared Server
##############################################
dispatchers="(PROTOCOL=TCP) (SERVICE=ESTANCIAXDB)"
 ##############################################
# Miscellaneous
##############################################
compatible=10.2.0.1.0
 ##############################################
# Sort, Hash Joins, Bitmap Indexes
##############################################
pga_aggregate_target=16777216
```

#3 inicia el proceso del servidor de oracle (MSDOS) (aparecerá un mensaje de instancia creada)

```
C:\oracle\product\10.2.0\db_1\bin\oradim.exe -new  -sid ESTANCIA -startmode manual -pfile
C:\oracle\product\10.2.0\db_1\dbs\initESTANCIA.ora
```

#4 para reconocer el usuario system y password de system (MSDOS)

```
C:\oracle\product\10.2.0\db_1\bin\oradim.exe -edit -sid ESTANCIA -startmode manual -srvcstart system -
syspwd estancia
```

#5 crear el archivo de autenticación (MSDOS) (revisen la ruta con el nuevo archivo PWDESTANCIA.ora)

```
orapwd file=C:\oracle\product\10.2.0\db_1\database\PWDESTANCIA.ora password=estancia force=y
entries=5
```

#6 localizar el siguiente archivo (C:\oracle\product\10.2.0\db_1\NETWORK\ADMIN\SAMPLE)

```
-- Editar listener.ora
LISTENER =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    )
  )

SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = ESTANCIA)
      (SID_NAME      = ESTANCIA)
    ) )
```

#7 localizar el siguiente archivo (C:\oracle\product\10.2.0\db_1\NETWORK\ADMIN\SAMPLE)

```
-- Editar tnsnames.ora
```

```
ESTANCIA =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = ESTANCIA)
    )
  )
```

#8 importante establecer esta variable de la BD (MSDOS)

```
set ORACLE_SID=ESTANCIA
```

Creación de la Base de Datos

Una vez creada la instancia procedemos a crear nuestra Base de Datos.

#9 conectarse como dba (MSDOS), aparecerá un mensaje "conectado a una instancia inactiva"

```
sqlplus "/ as sysdba"
```

#10 levantar la instancia en el prompt de sql> (aparecera mensaje de instancia oracle iniciada mostrando tamaños SGA,etc)

```
startup nomount pfile="C:\oracle\product\10.2.0\db_1\dbs\initESTANCIA.ora";
```

#11 crear la bd, puede tardar varios minutos depende su máquina, hasta que aparezca Base de Datos creada

```
create database ESTANCIA
maxlogfiles   32
MAXLOGMEMBERS 5
maxdatafiles 400
maxinstances   8
ARCHIVELOG
CHARACTER SET WE8ISO8859P15
NATIONAL CHARACTER SET AL16UTF16
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\sysaux_01.dbf' SIZE 200M
REUSE
DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\system_01.dbf' SIZE 500M
logfile  GROUP 1 ('C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\redo01a.log') SIZE 5M,
       GROUP 2 ('C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\redo02a.log') SIZE 5M
DEFAULT TEMPORARY TABLESPACE TEMP
      TEMPFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\temp_01.dbf' SIZE 50M
      REUSE AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
UNDO    TABLESPACE UNDO
      DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\undo_01.dbf' SIZE 610M
      REUSE AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
;
```

Para Generar el scrip de creación de objetos para nuestra Base de Datos, desde Erwin, procedemos de la siguiente manera:

Una vez abierto nuestro entorno con nuestro diagrama entidad relación, hacemos click en la barra de herramientas en **Tools** y después elegimos **Forward Engineer/..**, como se ve acontinuación:

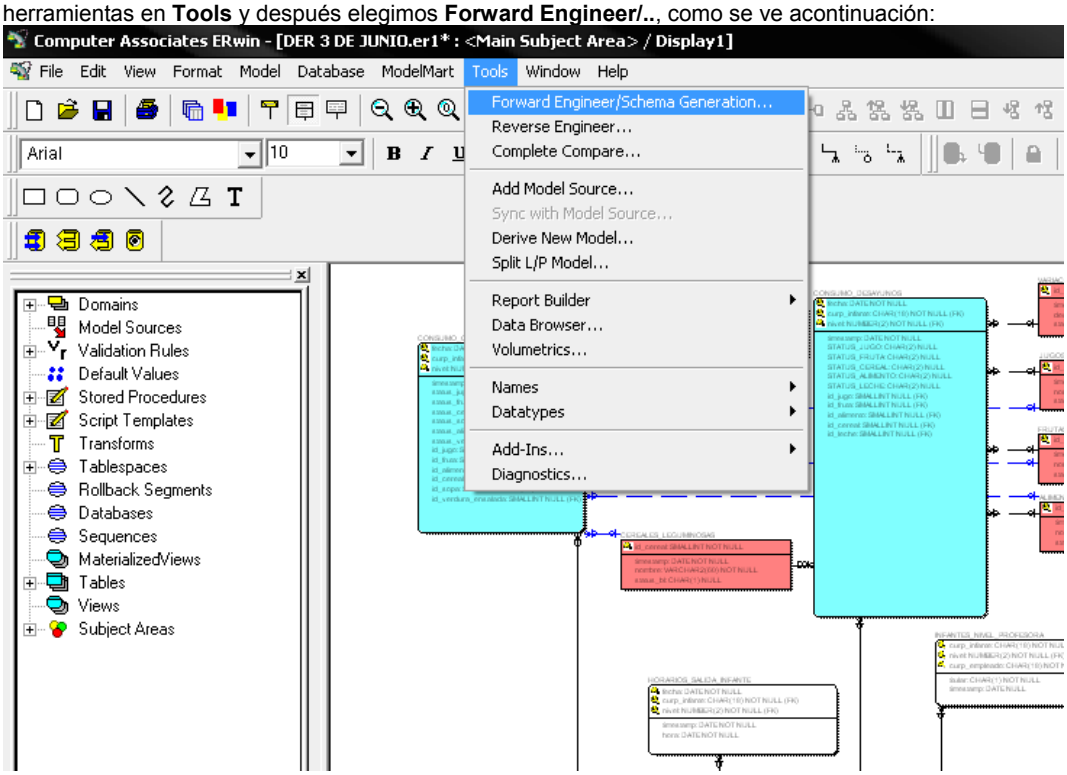


Fig.1.31. Ejemplo de generación de Scrip desde Erwin.

Ahora habilitamos las siguientes opciones, como se muestra en las siguientes imágenes
Para *schema:*

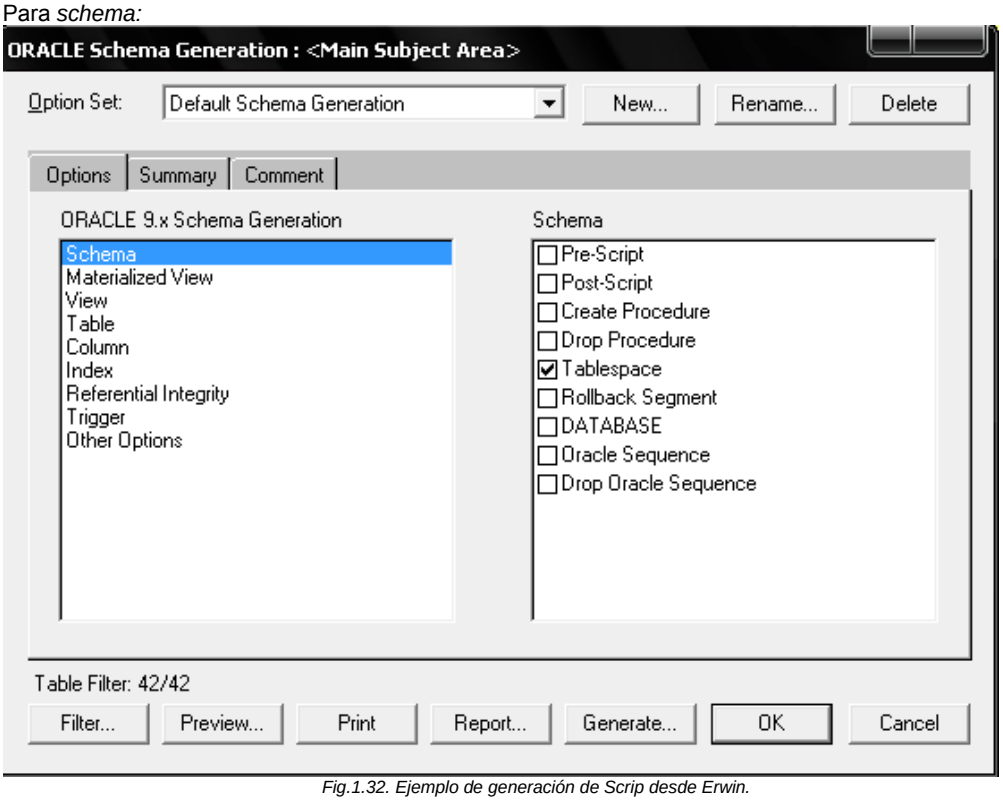
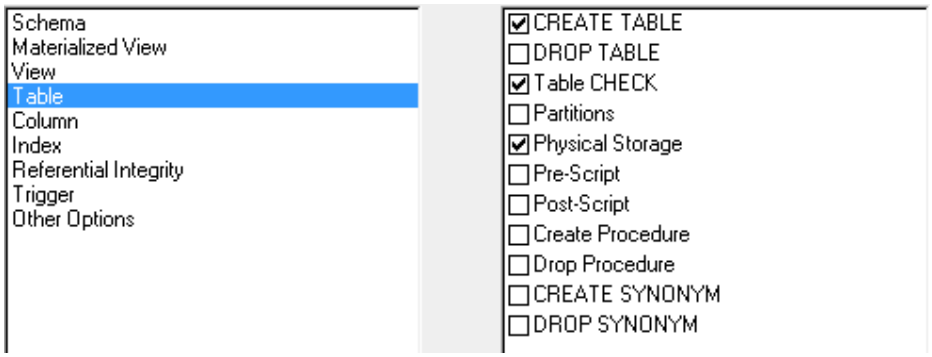
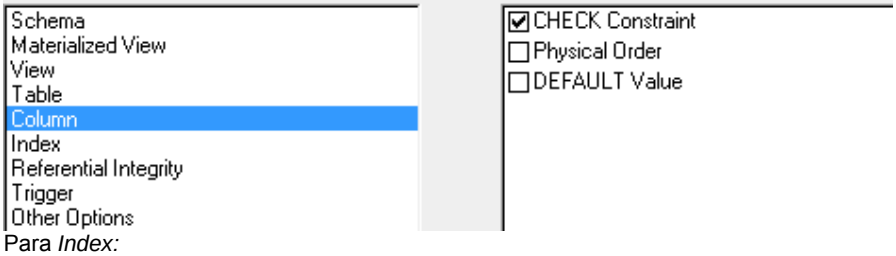


Fig.1.32. Ejemplo de generación de Scrip desde Erwin.

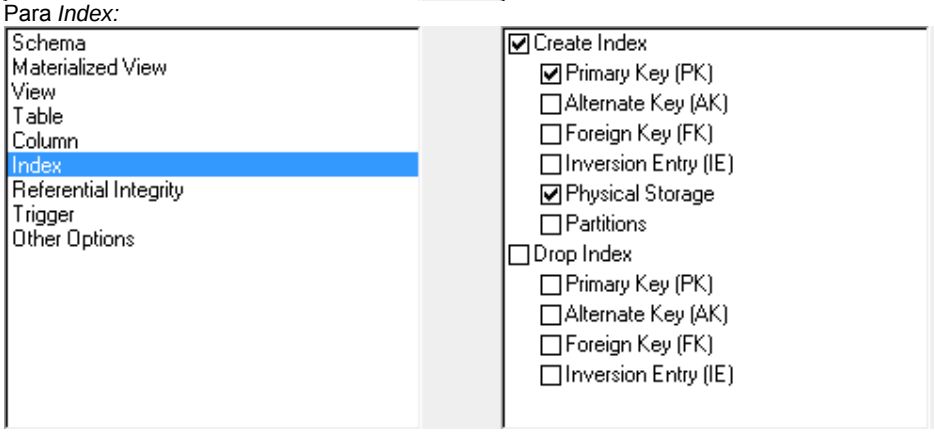
Habilitamos las siguientes opciones para *Table:*



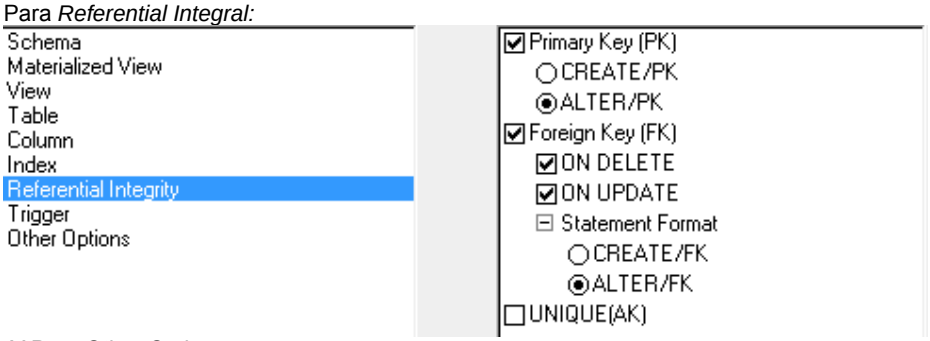
Para *column:*



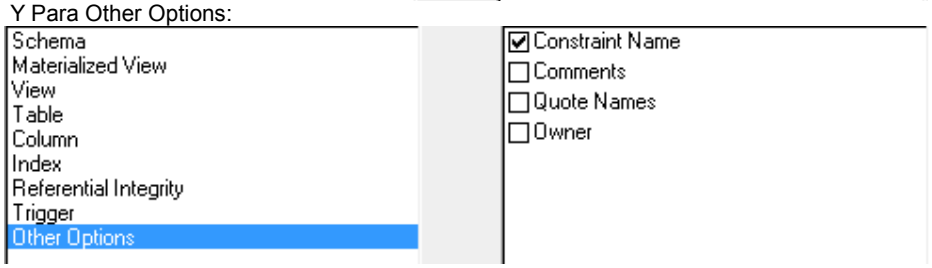
Para *Index:*



Para *Referential Integral:*



Y Para Other Options:



Y como vamos a modificar los tamaños de acuerdo a nuestros cálculos, hacemos clic en *Preview* y guardamos el scrip para modificarlo, como veremos ahora:

Creación de objetos (scrip):

```
CREATE TABLESPACE TS_CATALOGOS
		DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\ts_catalogos.dbf' SIZE 10M AUTOEXTEND on
		LOGGING
		DEFAULT STORAGE  (
				INITIAL 2330
				NEXT 2330
				MINEXTENTS 1
				MAXEXTENTS 3
				PCTINCREASE 1
		)
		ONLINE
		PERMANENT
;
CREATE TABLESPACE TS_INDICES
		DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\ts_indices.dbf' SIZE 10M AUTOEXTEND on
		LOGGING
		DEFAULT STORAGE  (
				INITIAL 251084587
				NEXT 251084587
				MINEXTENTS 1
				MAXEXTENTS 3
				PCTINCREASE 1
		)
		ONLINE
		PERMANENT
;
CREATE TABLESPACE TS_VOLATILES
		DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\ts_volatiles.dbf' SIZE 10M AUTOEXTEND on
		LOGGING
		DEFAULT STORAGE  (
				INITIAL 357167545
				NEXT 357167545
				MINEXTENTS 1
				MAXEXTENTS 3
				PCTINCREASE 1
		)
		ONLINE
		PERMANENT
;
```

Ejemplo de creación de algunas tablas, con sus respectivos constraints e índices:

```
CREATE TABLE CONSUMO_COMIDAS (
    fecha              DATE NOT NULL,
    curp_infante       CHAR(18) NOT NULL,
    nivel              NUMBER(2) NOT NULL,
    timestamp          DATE NOT NULL,
    status_jugo        CHAR(2) NULL CONSTRAINT STATUS216 CHECK (status_jugo IN ('PR', 'CT', 'NC', 'CM')),
    status_fruta       CHAR(2) NULL CONSTRAINT STATUS217 CHECK (status_fruta IN ('PR', 'CT', 'NC', 'CM')),
    status_cereal      CHAR(2) NULL CONSTRAINT STATUS218 CHECK (status_cereal IN ('PR', 'CT', 'NC', 'CM')),
    status_sopa        CHAR(2) NULL CONSTRAINT STATUS219 CHECK (status_sopa IN ('PR', 'CT', 'NC', 'CM')),
    status_alimento    CHAR(2) NULL CONSTRAINT STATUS220 CHECK (status_alimento IN ('PR', 'CT', 'NC', 'CM')),
    status_verdura_ensalada CHAR(2) NULL CONSTRAINT STATUS221 CHECK (status_verdura_ensalada IN ('PR', 'CT', 'NC', 'CM')),
    id_jugo            SMALLINT NULL,
    id_fruta           SMALLINT NULL,
    id_alimento        SMALLINT NULL,
    id_cereal          SMALLINT NULL,
    id_sopa            SMALLINT NULL,
    id_verdura_ensalada  SMALLINT NULL
)
		PCTFREE 0
		PCTUSED 99
		INITRANS 1
		MAXTRANS 99
		TABLESPACE TS_VOLATILES
		STORAGE  (
				INITIAL 19371
				NEXT 100
				MINEXTENTS 1
				MAXEXTENTS 2
				PCTINCREASE 0
		);

CREATE UNIQUE INDEX PK_CONSUMO_COMIDAS ON CONSUMO_COMIDAS
(
    fecha                  ASC,
    curp_infante             ASC,
    nivel                  ASC
)
		PCTFREE 0
		INITRANS 2
		MAXTRANS 182
		TABLESPACE TS_INDICES
		STORAGE  (
```

```
                    INITIAL 10544
                    NEXT 100
                    MINEXTENTS 1
                    MAXEXTENTS 2
                    PCTINCREASE 0
            )
;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT PK_CONSUMO_COMIDAS PRIMARY KEY (fecha,
        curp_infante, nivel)
    USING INDEX
            PCTFREE 0
            INITRANS 2
            MAXTRANS 182
            TABLESPACE TS_INDICES
            STORAGE  (
                    INITIAL 10544
                    NEXT 100
                    MINEXTENTS 1
                    MAXEXTENTS 2
                    PCTINCREASE 0
            ) ) ;


CREATE TABLE EXPEDIENTES_MEDICO_GUARDERIA (
    curp_infante        CHAR(18) NOT NULL,
    num_secuencial_expediente NUMBER(5,0) NOT NULL,
    timestamp           DATE NOT NULL,
    curp_empleado       CHAR(18) NOT NULL,
    fecha               DATE NOT NULL,
    num_enfermedad      NUMBER(3) NULL,
    comentario_constancia VARCHAR2(200) NULL,
    dias_permiso_constancia SMALLINT NULL,
    id_lugar_atencion   NUMBER(2) NULL
)
            PCTFREE 38
            PCTUSED 58
            INITRANS 1
            MAXTRANS 28
            TABLESPACE TS_VOLATILES
            STORAGE  (
                    INITIAL 6601
                    NEXT 100
                    MINEXTENTS 1
                    MAXEXTENTS 2
                    PCTINCREASE 0
            )
;

CREATE UNIQUE INDEX PK_EXPEDIENTES_MEDICO_GUARDERI ON EXPEDIENTES_MEDICO_GUARDERIA
(
    curp_infante                ASC,
    num_secuencial_expediente   ASC
)
            PCTFREE 0
            INITRANS 2
            MAXTRANS 122
            TABLESPACE TS_INDICES
            STORAGE  (
                    INITIAL 1536
                    NEXT 100
                    MINEXTENTS 1
                    MAXEXTENTS 2
                    PCTINCREASE 0
            );
ALTER TABLE EXPEDIENTES_MEDICO_GUARDERIA
    ADD  ( CONSTRAINT PK_EXPEDIENTES_MEDICO_GUARDERI PRIMARY KEY (
        curp_infante, num_secuencial_expediente)
    USING INDEX
            PCTFREE 0
            INITRANS 2
            MAXTRANS 122
            TABLESPACE TS_INDICES
            STORAGE  (
                    INITIAL 1536
                    NEXT 100
                    MINEXTENTS 1
                    MAXEXTENTS 2
                    PCTINCREASE 0
            ) ) ;

CREATE TABLE INFANTES (
    curp_infante        CHAR(18) NOT NULL,
    timestamp           DATE NOT NULL,
```

```
    nombre              VARCHAR2(70) NOT NULL,
    ap_pat              VARCHAR2(70) NOT NULL,
    fecha_nacimiento    DATE NOT NULL,
    peso_nacimiento     NUMBER(2,1) NOT NULL,
    estatura_nacimiento NUMBER(2,2) NOT NULL,
    calle_num           VARCHAR2(100) NOT NULL,
    colonia             VARCHAR2(120) NOT NULL,
    cp                  NUMBER(5,0) NOT NULL,
    id_municipio        NUMBER(3) NOT NULL,
    id_estado           NUMBER(2) NOT NULL,
    num_estancia        NUMBER(3) NOT NULL,
    sexo                CHAR(1) NOT NULL,
    fecha_ingreso       VARCHAR2(10) NOT NULL,
    leche_materna       CHAR(1) NOT NULL,
    fecha_egreso        DATE NULL,
    id_pais_vive        NUMBER(3) NOT NULL,
    ap_mat              VARCHAR2(70) NULL,
    id_pais_nacionalidad NUMBER(3) NULL,
    curp_infante_hermano CHAR(18) NULL,
    status_bl           CHAR(1) NULL
                        CONSTRAINT check_status_baja_logica257
                            CHECK (status_bl IN ('D'))
)
        PCTFREE 48
        PCTUSED 45
        INITRANS 1
        MAXTRANS 14
        TABLESPACE TS_VOLATILES
        STORAGE  (
                INITIAL 1004
                NEXT 100
                MINEXTENTS 1
                MAXEXTENTS 2
                PCTINCREASE 0
        )
        CACHE
;
CREATE UNIQUE INDEX PK_INFANTES ON INFANTES
(
    curp_infante                ASC
)
        PCTFREE 46
        INITRANS 2
        MAXTRANS 17
        TABLESPACE TS_INDICES
        STORAGE  (
                INITIAL 813
                NEXT 100
                MINEXTENTS 1
                MAXEXTENTS 2
                PCTINCREASE 0
        )
;
ALTER TABLE INFANTES
    ADD  ( CONSTRAINT PK_INFANTES PRIMARY KEY (curp_infante)
    USING INDEX
        PCTFREE 46
        INITRANS 2
        MAXTRANS 17
        TABLESPACE TS_INDICES
        STORAGE  (
                INITIAL 813
                NEXT 100
                MINEXTENTS 1
                MAXEXTENTS 2
                PCTINCREASE 0
        ) ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_VERDURAS_CONSUMO_COMIDA
        FOREIGN KEY (id_verdura_ensalada)
                REFERENCES VERDURAS_ENSALADAS
                ON DELETE SET NULL ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_SOPA_CONS_COM
        FOREIGN KEY (id_sopa)
                REFERENCES SOPAS
                ON DELETE SET NULL ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_CEREAL_LEGUM_CONS_COM
```

```
        FOREIGN KEY (id_cereal)
                REFERENCES CEREALES_LEGUMINOSAS
                ON DELETE SET NULL ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_ALIMENTO_FUERTE_CONS_COM
        FOREIGN KEY (id_alimento)
                REFERENCES ALIMENTOS_FUERTES
                ON DELETE SET NULL ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_FRUTA_YOG_POSTRE_CONS_COM
        FOREIGN KEY (id_fruta)
                REFERENCES FRUTAS_YOGOURTH_POSTRES
                ON DELETE SET NULL ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_JUGO_AGUA_CONS_COM
        FOREIGN KEY (id_jugo)
                REFERENCES JUGOS_AGUAS
                ON DELETE SET NULL ) ;


ALTER TABLE CONSUMO_COMIDAS
    ADD  ( CONSTRAINT FK_INFANTE_NIVEL_CONSUMO_COM
        FOREIGN KEY (curp_infante, nivel)
                REFERENCES INFANTES_NIVEL ) ;



ALTER TABLE EXPEDIENTES_MEDICO_GUARDERIA
    ADD  ( CONSTRAINT FK_LUGAR_ATENCION_EXPEDIENTE
        FOREIGN KEY (id_lugar_atencion)
                REFERENCES LUGARES_ATENCION
                ON DELETE SET NULL ) ;


ALTER TABLE EXPEDIENTES_MEDICO_GUARDERIA
    ADD  ( CONSTRAINT FK_ENFERMEDAD_EXPEDIENTE
        FOREIGN KEY (num_enfermedad)
                REFERENCES ENFERMEDADES
                ON DELETE SET NULL ) ;


ALTER TABLE EXPEDIENTES_MEDICO_GUARDERIA
    ADD  ( CONSTRAINT FK_INFANTE_EXPEDIENTE
        FOREIGN KEY (curp_infante)
                REFERENCES INFANTES ) ;


ALTER TABLE EXPEDIENTES_MEDICO_GUARDERIA
    ADD  ( CONSTRAINT FK_EMPLEADO_EXPEDIENTE
        FOREIGN KEY (curp_empleado)
                REFERENCES EMPLEADOS
                ON DELETE SET NULL ) ;


ALTER TABLE INFANTES
    ADD  ( CONSTRAINT FK_PAISES_INFANTES
        FOREIGN KEY (id_pais_vive)
                REFERENCES PAISES ) ;


ALTER TABLE INFANTES
    ADD  ( CONSTRAINT FK_PAIS_NACIMIENTO_INFANTE
        FOREIGN KEY (id_pais_nacionalidad)
                REFERENCES PAISES
                ON DELETE SET NULL ) ;

ALTER TABLE INFANTES
    ADD  ( CONSTRAINT FK_MUN_DEL_INFANTE
        FOREIGN KEY (id_municipio, id_estado, id_pais_vive)
                REFERENCES MUNICIPIOS_DELEGACIONES
                ON DELETE SET NULL ) ;

ALTER TABLE INFANTES
    ADD  ( CONSTRAINT FK_ESTANCIAS_INFANTE
        FOREIGN KEY (num_estancia)
                REFERENCES ESTANCIAS
                ON DELETE SET NULL ) ;
```

…

Reportes en ERwin para el Diccionario de Datos Empresarial.

ERwin ofrece una variedad de reportes de entidades, atributos y relaciones. Los reportes pueden ser definidos por el usuario, seleccionando distintos criterios con sólo marcarlos con el mouse. Los reportes resultan de suma utilidad para el análisis y documentación de los modelos.

Pasos a seguir:

Para crear nuevos reportes:

1. Presione el botón **Data Browser** en la barra de herramientas del menú principal.

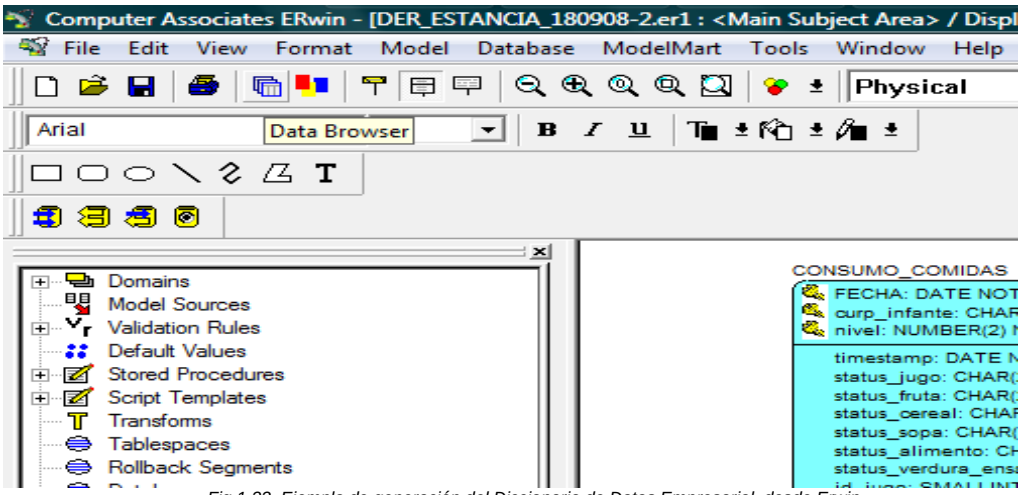


Fig.1.33. Ejemplo de generación del Diccionario de Datos Empresarial desde Erwin.

2. Al aparecer la ventana del **Data Browser,** presione el botón **New report or folder** y escoja la opción **Erwin Report**

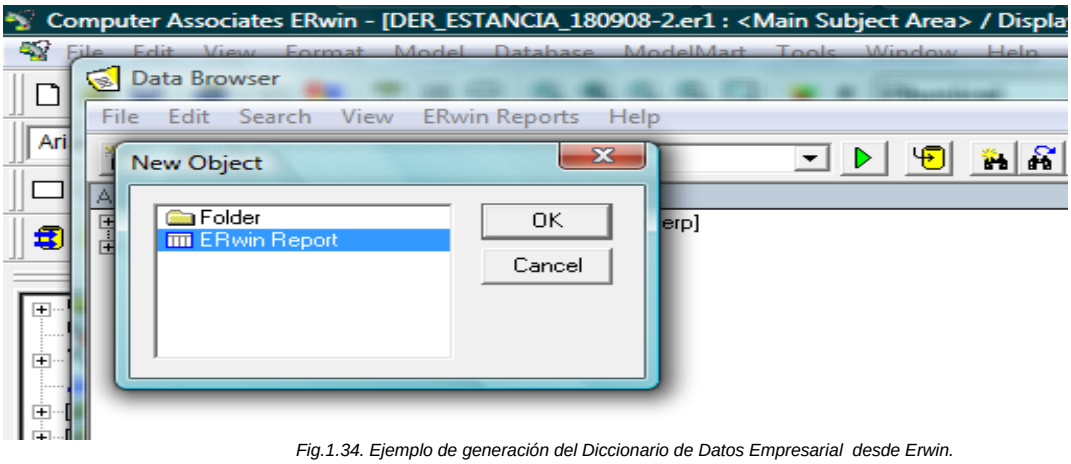


Fig.1.34. Ejemplo de generación del Diccionario de Datos Empresarial desde Erwin.

3. Introduzca un nombre para su informe en **Name.** Escoja una de las siguientes opciones, **Logical** o **Physical,** si el reporte es sobre el modelo lógico o el modelo físico, respectivamente. Seleccione el tipo de informe que usted quiere de la lista de Categorías (**Category**).

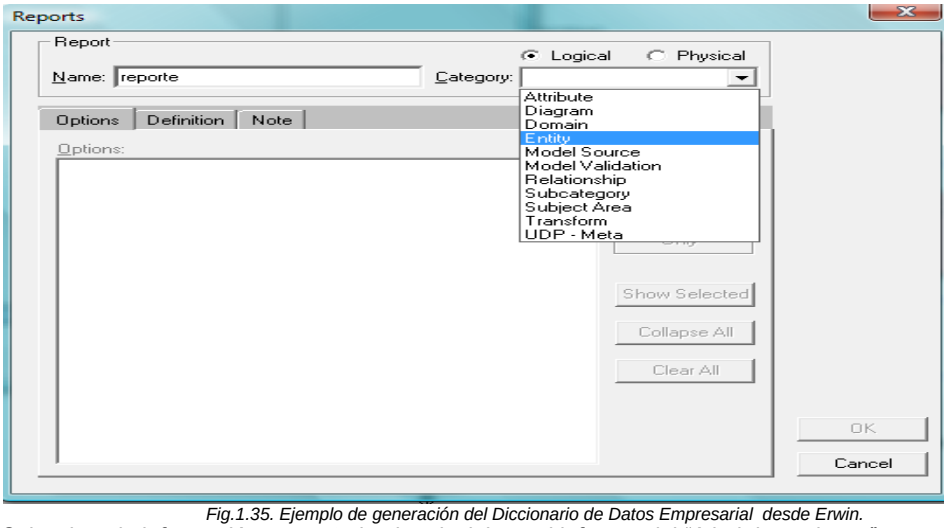


Fig.1.35. Ejemplo de generación del Diccionario de Datos Empresarial desde Erwin.

4. Seleccione la información que usted quiere incluir en el informe, del "árbol de opciones"

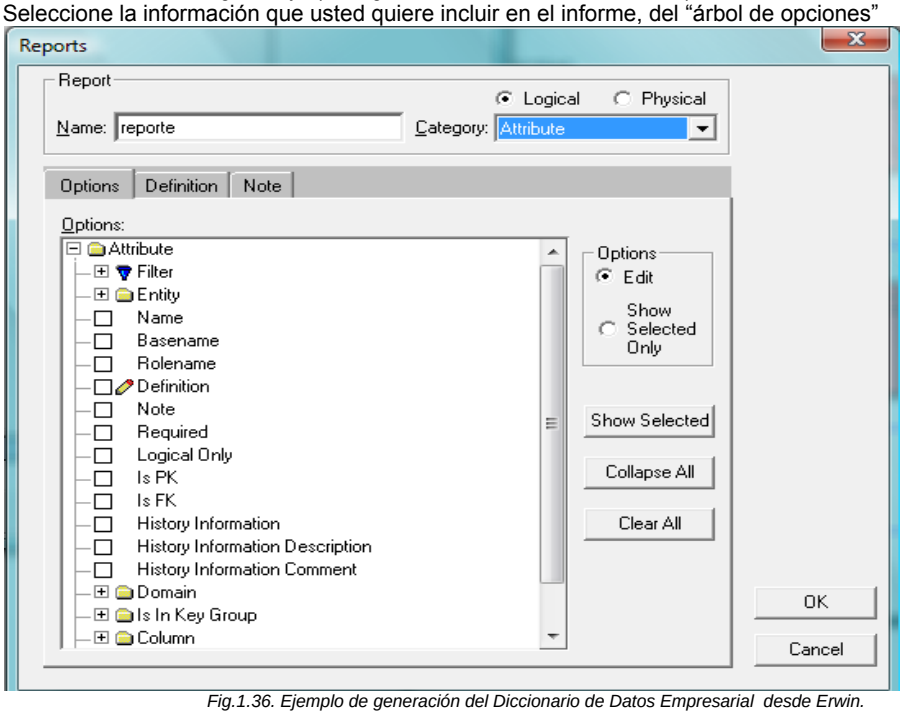


Fig.1.36. Ejemplo de generación del Diccionario de Datos Empresarial desde Erwin.

5. Presione **OK.** El **Data Browser** agrega su informe al árbol de control en la ventana **All reports** del **Data Browser.** Por ultimo Haga doble click en el nombre del reporte para ejecutarlo

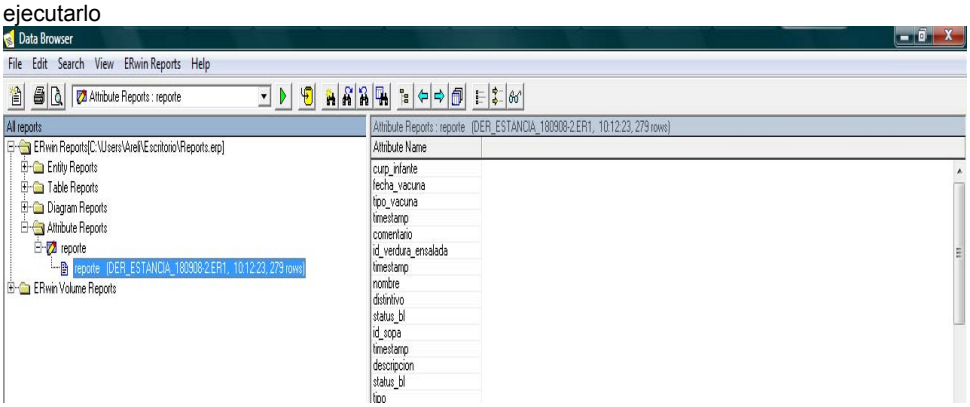


Fig.1.37. Ejemplo de generación del Diccionario de Datos Empresarial desde Erwin.

Salida de un Reporte Empresarial:
Seleccionamos Column Reports: **Reporte_infantes** desde el filtro
Y nos muestra el siguiente reporte:

Column Table Name	Column Table Comment	Column Name	Column Datatype	Column Null Option	Column Comment
INFANTES	Tabla donde se almacenan los datos generales de los niños de la guardería	curp_infante	CHAR(18)	NOT NULL	CURP del niño
		timestamp	DATE	NOT NULL	Fechas cuando se realizan cargas a la tabla
		nombre	VARCHAR2(70)	NOT NULL	Nombre del infante
		ap_pat	VARCHAR2(70)	NOT NULL	apellido paterno del infante
		fecha_nacimiento	DATE	NOT NULL	fecha de nacimiento del infante
		peso_nacimiento	NUMBER(2,1)	NOT NULL	Peso de nacimiento del infante
		estatura_nacimiento	NUMBER(2,2)	NOT NULL	Estatura de nacimiento del infante
		calle_num	VARCHAR2(100)	NOT NULL	Calle y numero de la direccion del infante
		colonia	VARCHAR2(120)	NOT NULL	Colonia donde vive el infante
		cp	NUMBER(5)	NOT NULL	Código postal del infante
		id_municipio	NUMBER(3)	NOT NULL	Identificador del municipio donde vive el infante
		id_estado	NUMBER(2)	NOT NULL	Identificador del estado donde vive el infante
		num_estancia	NUMBER(3)	NOT NULL	Numero de estancia donde esta inscrito el infante
		sexo	CHAR(1)	NOT NULL	Sexo del infante
		fecha_ingreso	DATE	NOT NULL	Fecha de ingreso a la estancia del infante
		leche_materna	CHAR(1)	NOT NULL	Campo que indica si el infante tomo lecha materna al nacer
		fecha_egreso	DATE	NULL	Fecha en la que si es el caso, el infante egreso de la guardería
		id_pais_vive	NUMBER(3)	NOT NULL	Identificador del país donde vive el infante
		ap_mat	VARCHAR2(70)	NULL	apellido materno del infante
		id_pais_nacionalidad	NUMBER(3)	NULL	Identificador del país donde nació el infante
		curp_infante_hermano	CHAR(18)	NULL	CURP del hermano mayor que haya ingresado a la guardería
		status_bl	CHAR(1)	NULL	Indica si esta activa o no la tabla

Ejemplos de Carga de Datos en Oracle, PostgreSQL y SQLServer

Parte importante de nuestro trabajo es la carga inicial de datos en nuestra Base de Datos, es por eso que consideramos necesario explicar brevemente algunas de las formas más comunes de hacerlo es los distintos manejadores que empleamos (Oracle, PostgreSQL, y SQL Server 2005).

Las cargas que explicamos se realizan desde archivos *.txt, .xls, o .cvs* a la Base de Datos en el manejador correspondiente pues es la manera más común de obtener los datos.

Para ejemplificar las diferentes cargas emplearemos la tabla "INFANTES" que es la tabla más representativa de nuestra base, además de tener una amplia variedad de tipos de datos.

INFANTES

curp_infante: CHAR(18) NOT NULL
timestamp: DATE NOT NULL
nombre: VARCHAR2(70) NOT NULL
ap_pat: VARCHAR2(70) NOT NULL
fecha_nacimiento: DATE NOT NULL
peso_nacimiento: NUMBER(2,1) NOT NULL
estatura_nacimiento: NUMBER(2,2) NOT NULL
calle_num: VARCHAR2(100) NOT NULL
colonia: VARCHAR2(120) NOT NULL
cp: CHAR(5) NOT NULL
id_municipio: NUMBER(3) NOT NULL (FK)
id_estado: NUMBER(2) NOT NULL (FK)
num_estancia: NUMBER(3) NOT NULL (FK)
sexo: CHAR(1) NOT NULL
fecha_ingreso: DATE NOT NULL
leche_materna: CHAR(1) NOT NULL
fecha_egreso: DATE NULL
id_pais_vive: NUMBER(3) NOT NULL (FK)
ap_mat: VARCHAR2(70) NULL
id_pais_nacionalidad: NUMBER(3) NULL (FK)
curp_infante_hermano: CHAR(18) NULL (FK)
status_bl: CHAR(1) NULL

Fig.1.38. Tabla Infantes.

Notas antes de comenzar:

Para evitar cometer algunos de los errores más comunes al intentar realizar una carga de datos debemos:

- Debemos tener cuidado en que el formato de los datos a cargar corresponda al formato de los "data types" de la tabla receptora.

- Para las diferentes formas de cargar datos debemos tener cuidado al emplear el carácter coma (,) para separar las columnas , ya que podemos presentar problemas con algunos tipos de datos de punto flotante.

- Los formatos de fechas que emplean los distintos manejadores suelen ser diferentes. Es muy importante tener nuestros datos con el formato de fecha correcto.

- Los manejadores suelen usar "data types" similares pero difícilmente iguales, es por eso que debemos tener en cuenta este punto para evitar posibles errores al migrar nuestros datos de un manejador a otro.

- Los datos a cargar se encuentran en formato “csv” de Excel y el archivo lleva por nombre “muestra_datosCSV.csv”. - El script de creación de la tabla infantes en Oracle es el siguiente:	CREATE TABLE INFANTES (curp_infante CHAR(18) NOT NULL, timestamp DATE NOT NULL, nombre VARCHAR2(70) NOT NULL, ap_pat VARCHAR2(70) NOT NULL, fecha_nacimiento DATE NOT NULL, peso_nacimiento NUMBER(2,1) NOT NULL, estatura_nacimiento NUMBER(2,2) NOT NULL, calle_num VARCHAR2(100) NOT NULL, colonia VARCHAR2(120) NOT NULL, cp CHAR(5) NOT NULL, id_municipio NUMBER(3) NOT NULL, id_estado NUMBER(2) NOT NULL, num_estancia NUMBER(3) NOT NULL, sexo CHAR(1) NOT NULL, fecha_ingreso DATE NOT NULL, leche_materna CHAR(1) NOT NULL fecha_egreso DATE NULL, id_pais_vive NUMBER(3) NOT NULL, ap_mat VARCHAR2(70) NULL, id_pais_nacionalidad NUMBER(3) NULL, curp_infante_hermano CHAR(18) NULL, status_bl CHAR(1) NULL,);

Carga empleando el Wizard de la versión “Express Edition”

1. Desde el menú inicio debemos ir a “Ir a Página Inicial de Base de Datos”, o a la dirección que utilice oracle como “localhost”, en este caso: http://127.0.0.1:8080/apex/

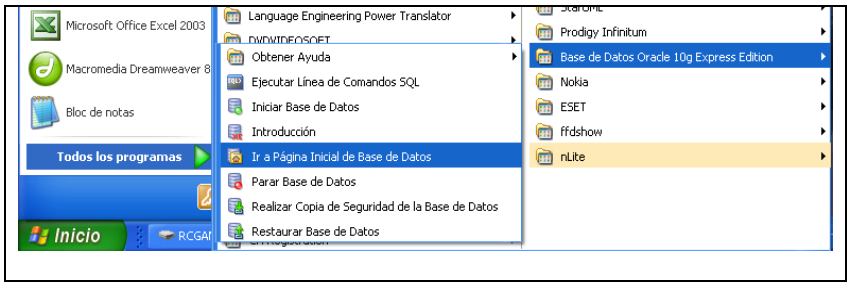


Fig.1.39. Ejemplo Carga de datos desde Wizard

2. Posteriormente Oracle nos pedirá que hagamos “login” para poder acceder:

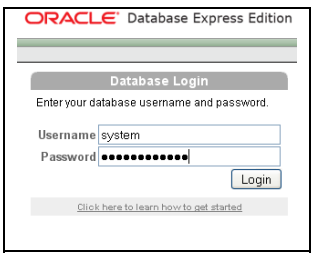


Fig.1.40. Ejemplo Carga de datos desde Wizard

3. Una vez dentro, iremos a: - Utilities

- Data Load/Unload
- Load

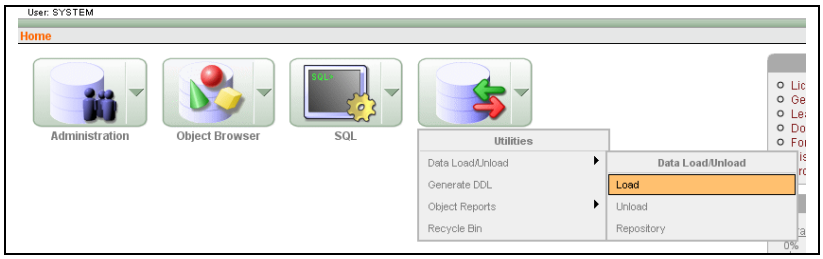


Fig.1.41. Ejemplo Carga de datos desde Wizard

4. Damos click en Load Text Data:

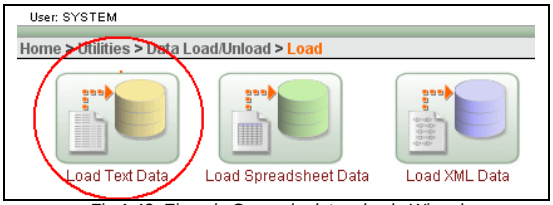


Fig.1.42. Ejemplo Carga de datos desde Wizard

5. Como la tabla a la que le cargaremos los datos ya existe en nuestra base, escogemos la opción: “Load to: Existing table”

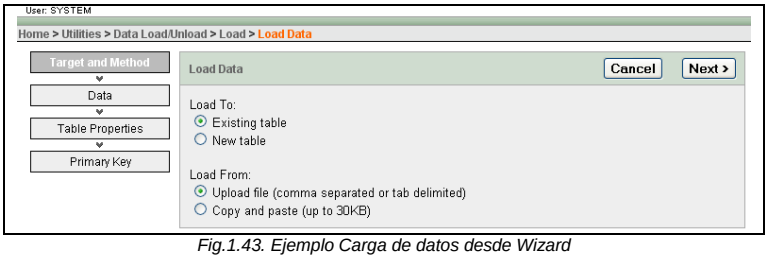


Fig.1.43. Ejemplo Carga de datos desde Wizard

6. Escogemos el “schema” en que esta nuestra tabla:

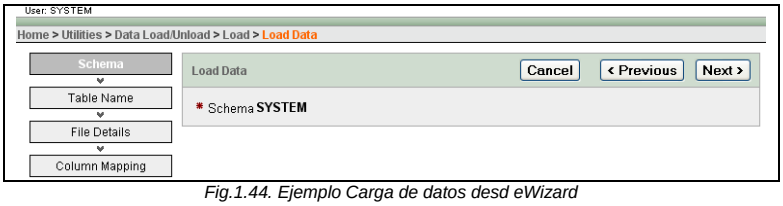


Fig.1.44. Ejemplo Carga de datos desd eWizard

7. Seleccionamos la tabla a la que le cargaremos lo datos:

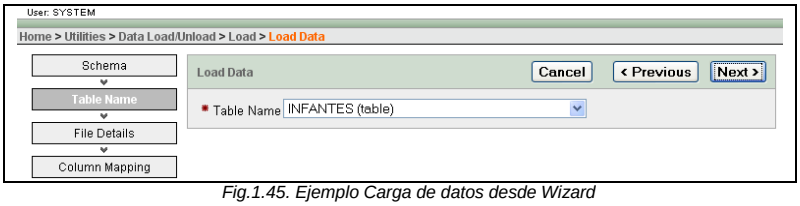


Fig.1.45. Ejemplo Carga de datos desde Wizard

8. Seleccionamos el archivo CSV que contiene los datos a cargar:

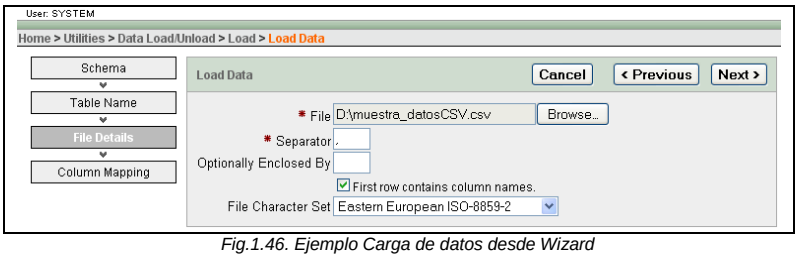


Fig.1.46. Ejemplo Carga de datos desde Wizard

9. Oracle nos pedirá que verifiquemos el lugar en que se cargaran los datos y los datos a cargar:

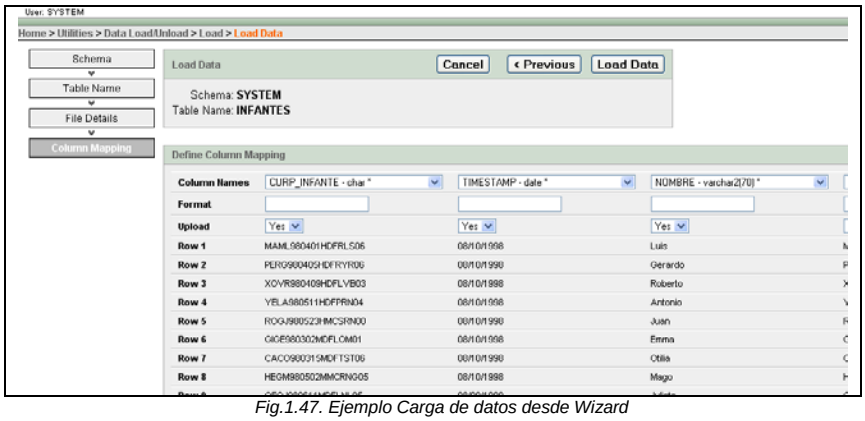


Fig.1.47. Ejemplo Carga de datos desde Wizard

10. Y finalmente Oracle nos mostrara el número de renglones cargados y el número de renglones que no se pudieron cargar.

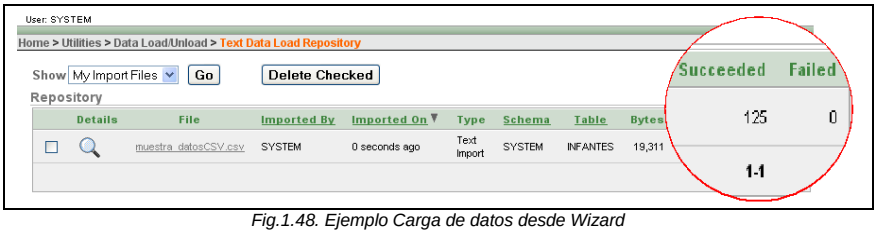


Fig.1.48. Ejemplo Carga de datos desde Wizard

Empleando sqlldr.exe

1. Buscar desde Windows el archivo sqlldr.exe y en esa ruta copiar los archivos “infantes.txt” y el archivo “muestra_datosCSV2.txt”. Este último es el archivo que contiene los datos a cargar y es una pequeña variante del formato “csv”, ya que requerimos cambiar el formato de la fecha y el símbolo “,” que utiliza csv para separar columnas fue cambiado por el símbolo “*”.

infantes.txt

```
load data
infile 'muestra_datosCSV2.txt'
append into table infantes
fields terminated by '*'
(curp_infante,timestamp date(10) 'MM/DD/YYYY',nombre,ap_pat,fecha_nacimiento date(10)
'MM/DD/YYYY',peso_nacimiento,estatura_nacimiento,calle_num,colonia,cp
char(5),id_municipio,id_estado,num_estancia,sexo,fecha_ingreso date(10)
'MM/DD/YYYY',leche_materna,fecha_egreso date(10)
'MM/DD/YYYY',id_pais_vive,ap_mat,id_pais_nacionalidad,curp_infante_hermano,status_bl)
```

2. Después tenemos que abrir una ventana de “símbolo del sistema” y ubicarnos en la ruta donde copiamos los archivos y teclear el comando:

```
sqlld userid= system control = infantes.txt
```

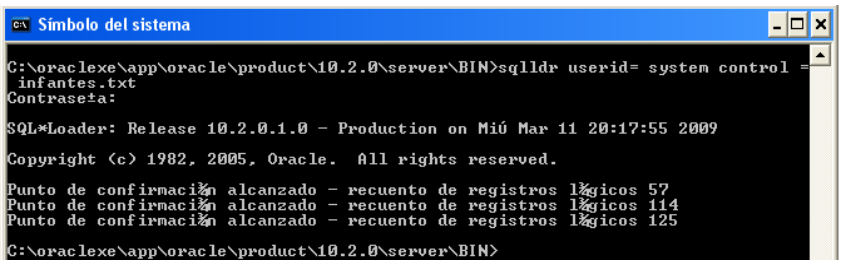


Fig.1.49 Ejemplo Carga de datos empleando sqlldr.exe

3. Y finalmente podemos comprobar la carga de los datos y los posibles errores en el archivo "infantes.log" que crea Oracle después de efectuar una carga.

PostgreSQL

- Los datos a cargar se encuentran en formato "csv" de Excel y el archivo lleva por nombre "muestra_datosCSV.csv". - El script de creación de la tabla infantes en PostgreSQL es el siguiente:	CREATE TABLE INFANTES (curp_infante CHAR(18), timestamp DATE, nombre VARCHAR(70), ap_pat VARCHAR(70), fecha_nacimiento date, peso_nacimiento float(2), estatura_nacimiento float(2), calle_num VARCHAR(100), colonia VARCHAR(120), cp CHAR(5), id_municipio numeric(3), id_estado numeric(2), num_estancia numeric(3), sexo CHAR(1), fecha_ingreso date, leche_materna CHAR(1), fecha_egreso DATE, id_pais_vive numeric(3), ap_mat VARCHAR(70), id_pais_nacionalidad numeric(3), curp_infante_hermano CHAR(18), status_bl CHAR(1));

Empleando psql (comando COPY)

1. Primero ingresamos a una pantalla de comando de psql y nos logueamos emplendo el comando:

```
psql –U postgres –d tesis
```

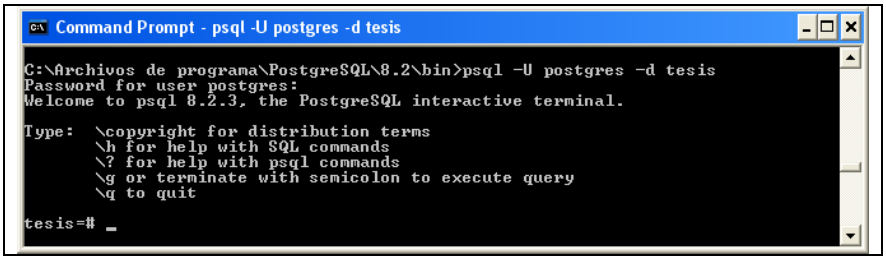


Fig.1.50. Ejemplo Carga de datos empleando COPY en Postgres

2. Posteriormente empleamos el comando COPY indicandole en donde esta el archivo CSV a cargar:

```
COPY infantes from 'd:muestra_datosCSV.csv' DELIMITERS ',' CSV HEADER;
```

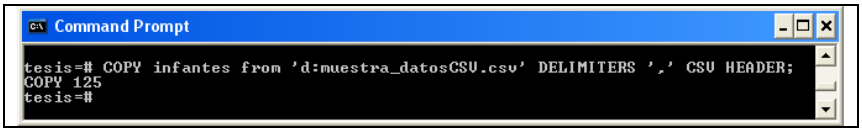


Fig.1.51. Ejemplo Carga de datos empleando COPY en Postgres

Microsoft SQLServer

Empleando el Wizard de SQL Server 2005

- Los datos a cargar se encuentran en formato “csv” de Excel y el archivo lleva por nombre “muestra_datos.xls”.

- El script de creación de la tabla infantes en SQL Server 2005 es el siguiente:

```
CREATE TABLE INFANTES (
curp_infante            CHAR(18) NOT NULL,
timestamp               datetime NOT NULL,
nombre                  VARCHAR(70) NOT NULL,
ap_pat                  VARCHAR(70) NOT NULL,
fecha_nacimiento        datetime NOT NULL,
peso_nacimiento         float(2) NOT NULL,
estatura_nacimiento     float(2) NOT NULL,
calle_num               VARCHAR(100) NOT NULL,
colonia                 VARCHAR(120) NOT NULL,
cp                      CHAR(5) NOT NULL,
id_municipio            decimal(3) NOT NULL,
id_estado               decimal(2) NOT NULL,
num_estancia            decimal(3) NOT NULL,
sexo                    CHAR(1) NOT NULL,
fecha_ingreso           datetime NOT NULL,
leche_materna           CHAR(1) NOT NULL,
fecha_egreso            datetime NULL,
id_pais_vive            decimal(3) NOT NULL,
ap_mat                  VARCHAR(70) NULL,
id_pais_nacionalidad    decimal(3) NULL,
curp_infante_hermano    CHAR(18) NULL,
status_bl               CHAR(1) NULL
);
```

1. En el menú inicio abrimos “SQL Server Management Studio”:

Fig.1.52. Ejemplo Carga de datos empleando Wizard en SQLServer

2. Una vez dentro, damos clic derecho en la Base de Datos que contiene la tabla a la que le queremos insertar datos y seleccionamos, del menú contextual desplegado, la opción:

- Tasks
  - Import Data

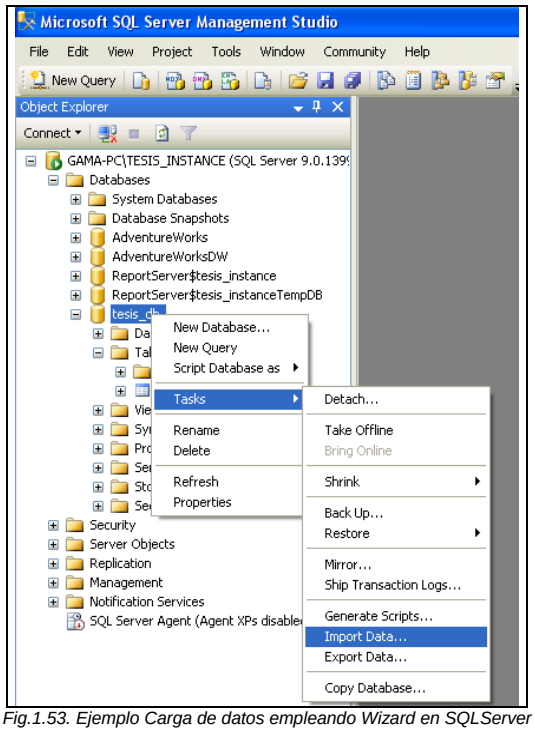


Fig.1.53. Ejemplo Carga de datos empleando Wizard en SQLServer

3. Se abrirá la primera pantalla de presentación del “wizard”:

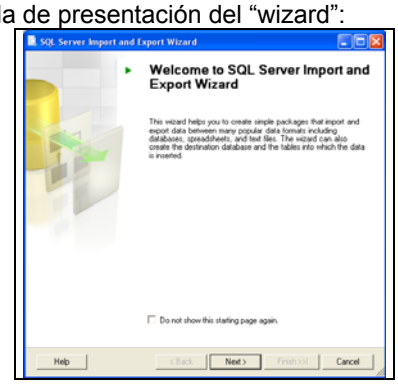


Fig.1.54. Ejemplo Carga de datos empleando Wizard en SQLServer

4. En la siguiente pantalla elegiremos el tipo de formato que tienen nuestros datos que

queremos cargar, ya que SQL Server tiene gran compatibilidad con Excel, escogeremos esta opción, además de señalar la ubicación y nombre del archivo que contienen nuestros datos:

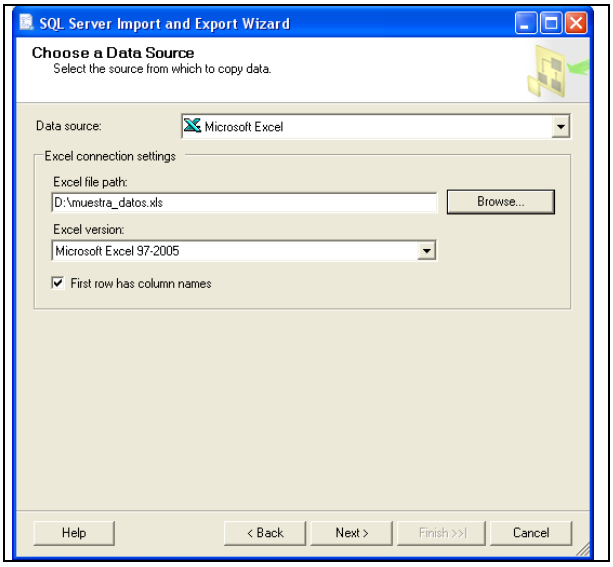


Fig.1.55. Ejemplo Carga de datos empleando Wizard en SQLServer

5. En la siguiente pantalla elegimos el destino de nuestro datos, en este caso SQL Native Client, el servidor en que estemos trabajando y el nombre de nuestra Base de Datos:

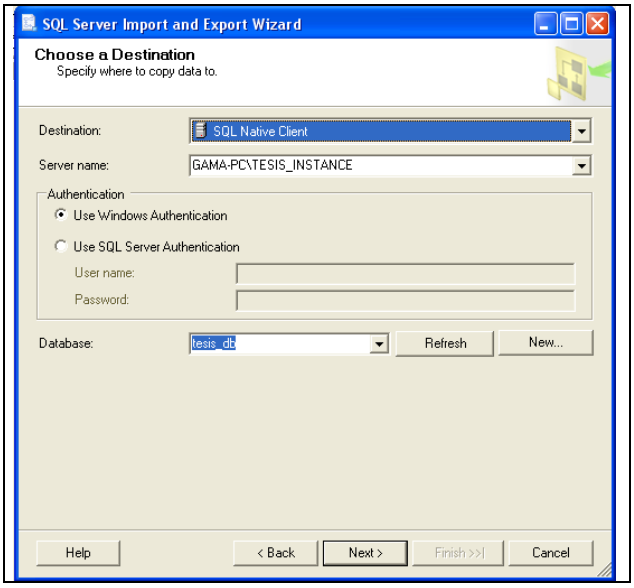


Fig.1.56. Ejemplo Carga de datos empleando Wizard en SQLServer

6. En la siguiente pantalla elegimos la opción “Copy data from one or more tables or views”:

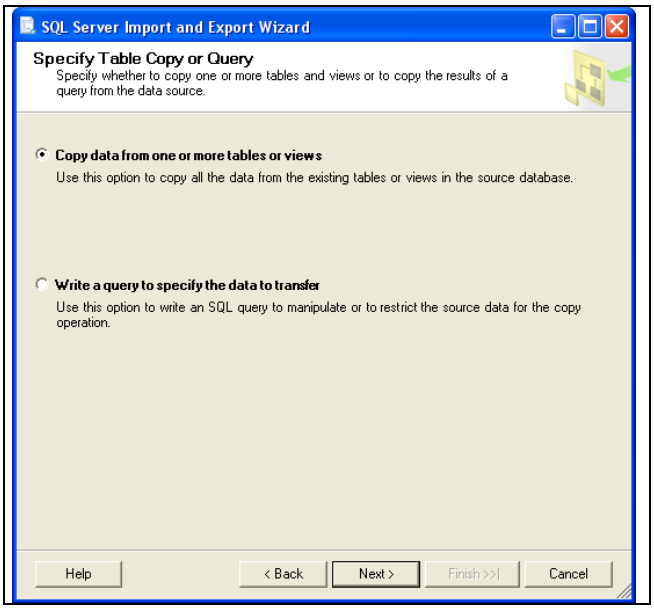


Fig.1.57. Ejemplo Carga de datos empleando Wizard en SQLServer

7. En la siguiente pantalla elegimos la tabla a la que se le cargaran los datos, en este caso “infantes”:

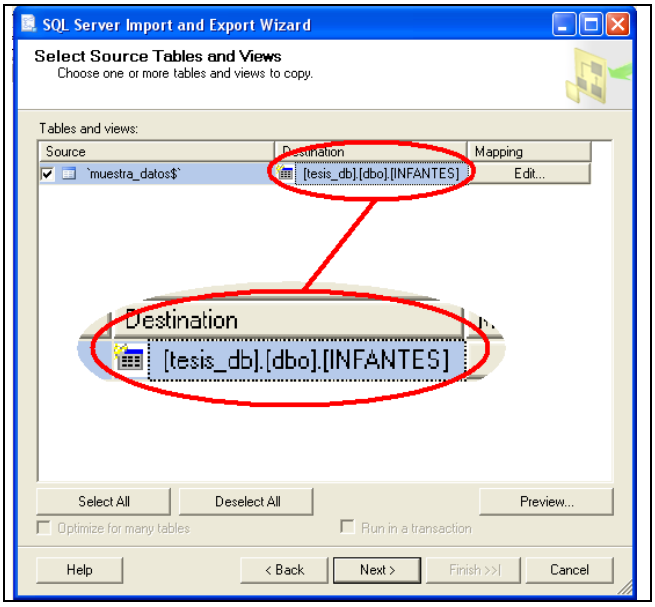


Fig.1.58. Ejemplo Carga de datos empleando Wizard en SQLServer

8. En la siguiente pantalla seleccionamos la opción “execute immediately”:

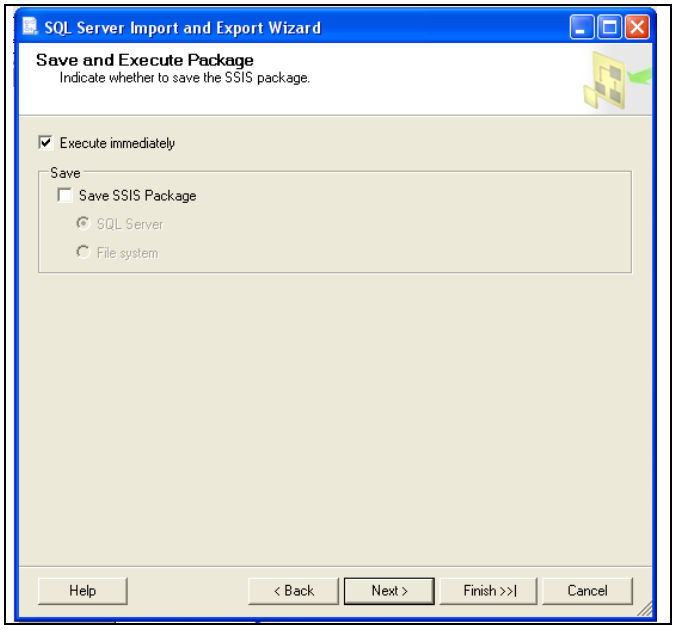


Fig.1.59. Ejemplo Carga de datos empleando Wizard en SQLServer

9. En la siguiente pantalla se nos pide verificar lo que se realizara:

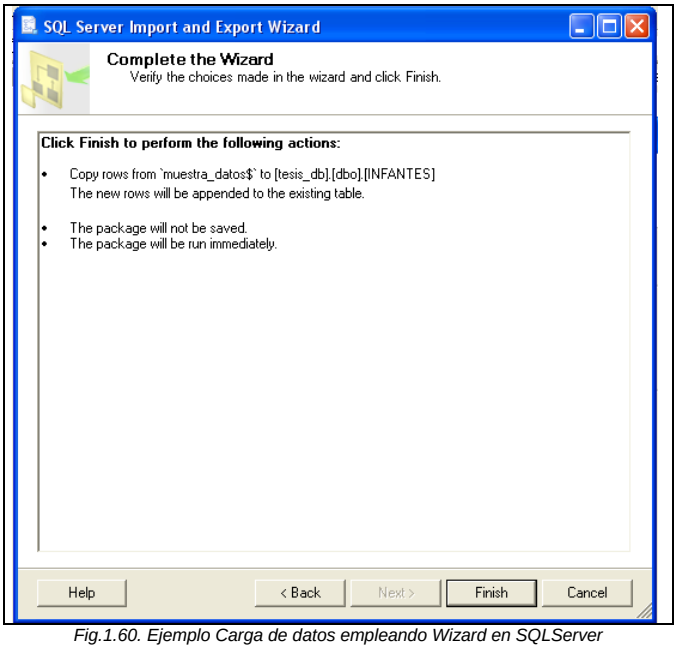


Fig.1.60. Ejemplo Carga de datos empleando Wizard en SQLServer

10. Y en la ultima ventana se nos dice que la carga fue satisfactoria o en caso contrario se nos muestran los errores ocurridos:

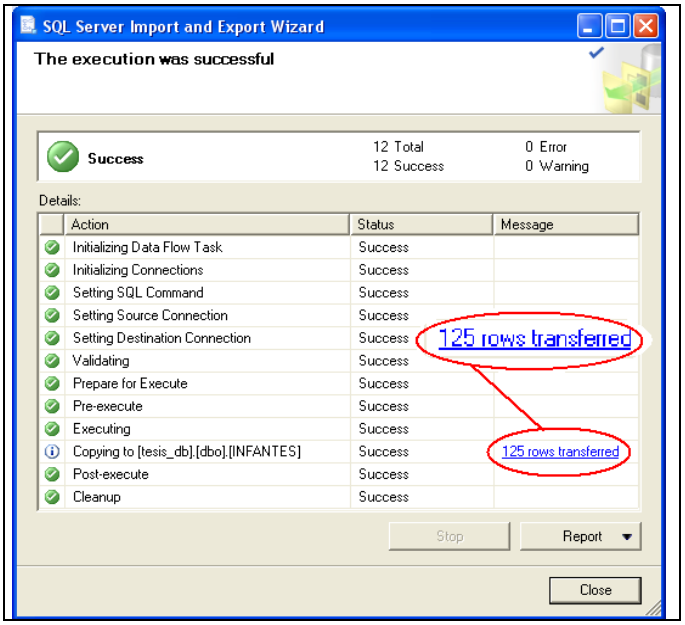


Fig.1.61. Ejemplo Carga de datos empleando Wizard en SQLServer

Empleando el comando BULK INSERT

1. Dentro de “SQL Server Management Studio” abrimos una ventana “New Query” y ejecutamos el comando:

```
BULK INSERT tesis_db.dbo.infantes
  FROM 'd:muestra_datos.csv'
  UIT
   (
     FIELDTERMINATOR =',',
     ROWTERMINATOR = '\n',
     FIRSTROW = 2,
     LASTROW = 126
    );
```

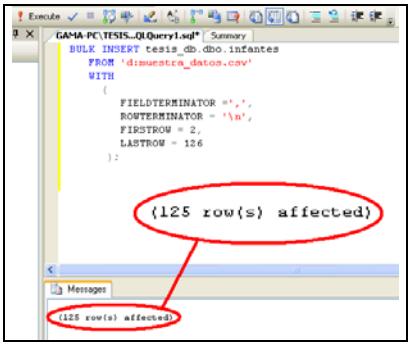


Fig.1.62. Ejemplo Carga de datos empleando Bulk en SQLServer

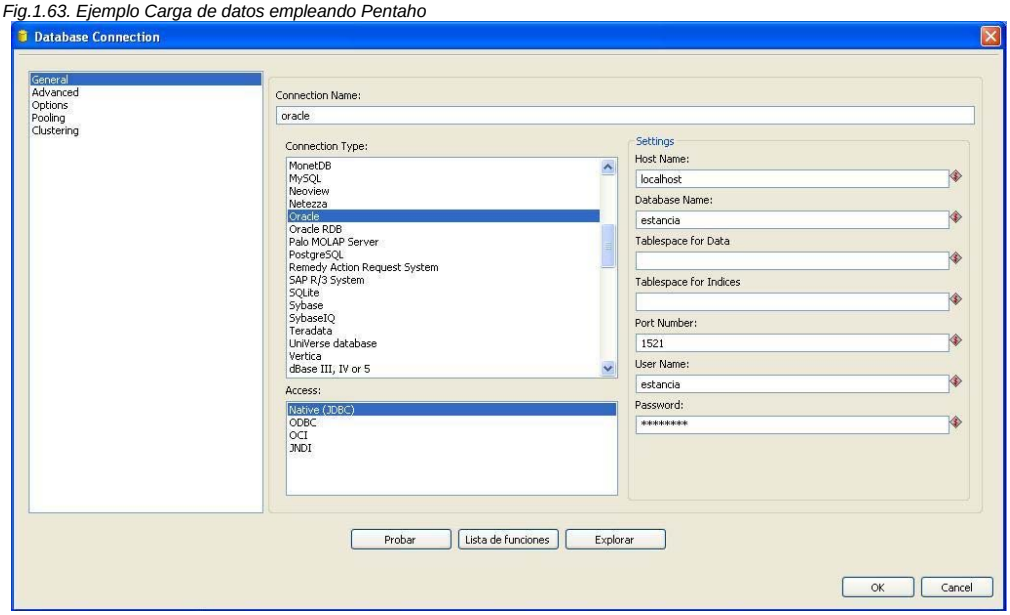
NOTA: Dichas cargas se tratan de ejemplos pues la carga definitiva se hizo desde archivos *.xls* y *.cvs* a nuestra Base de Datos en *Oracle* con ayuda de *Pentaho*, herramienta que explicaremos a continuación.

Carga de Datos Completa con Pentaho.

Una vez montada y abierta nuestra Base de Datos Estancia, nos conectados con nuestro usuario predeterminado (us: estancia, pw: estancia).

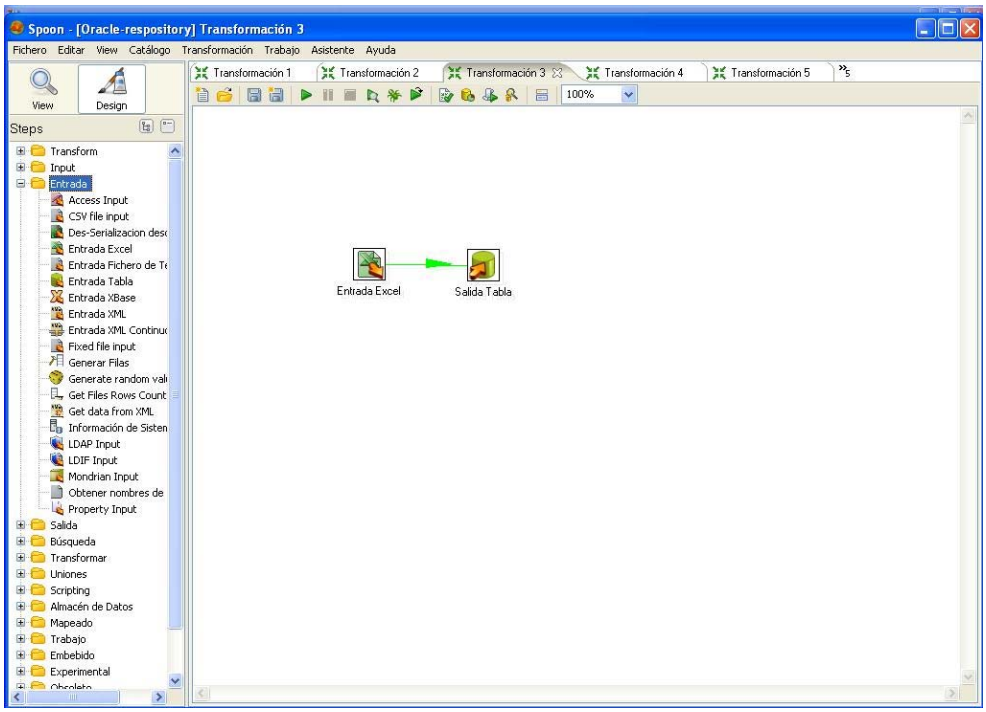
Lo primero que hacemos es establecer la conexión con nuestra Base de Datos, como se muestra a continuación:

Fig.1.63. Ejemplo Carga de datos empleando Pentaho



Ahora definimos una nueva transformación, en la que la entrada será un archivo de Excel (.xls) y la salida será una tabla de nuestra Base de Datos en ORACLE.

Fig.1.64. Ejemplo Carga de datos empleando Pentaho



Ahora hacemos doble clic en la entrada para especificar el archivo del que obtendremos los datos, en examinar elegimos el archivo y una vez ubicado, hacemos clic en añadir.

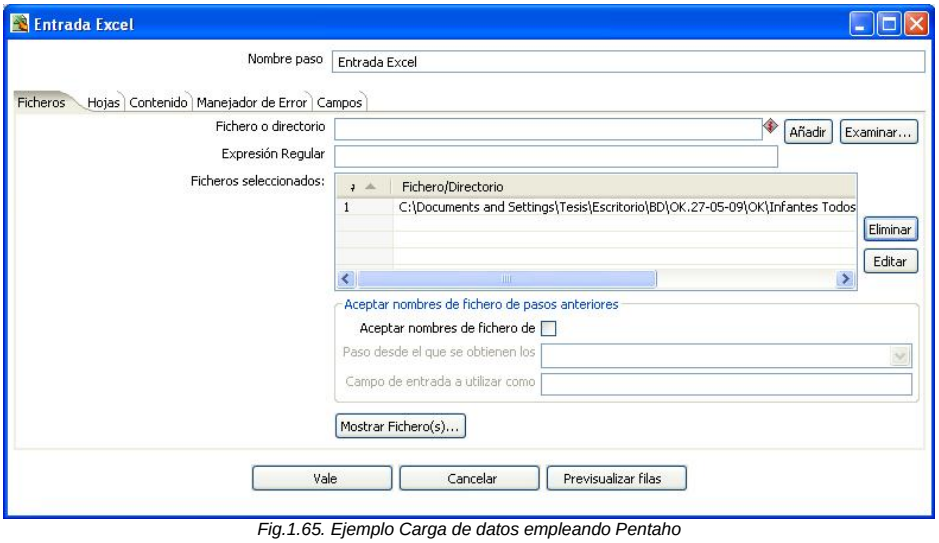


Fig.1.65. Ejemplo Carga de datos empleando Pentaho

En la pestaña Hoja elegimos la hoja del archivo de Excel del que provienen los datos.

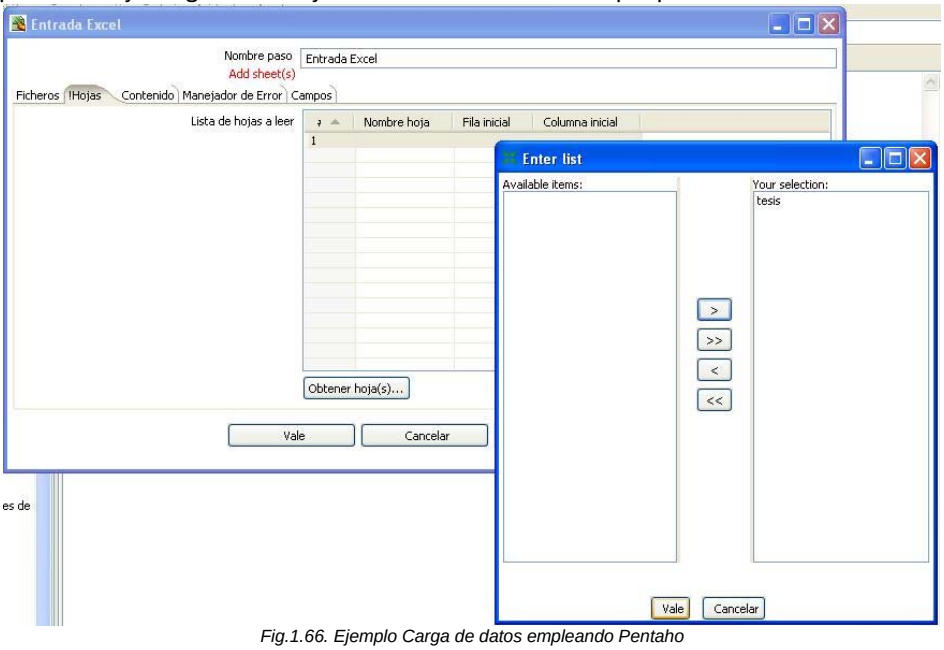


Fig.1.66. Ejemplo Carga de datos empleando Pentaho

Una vez elegida la hoja, abrimos la pestaña Campos en la que especificaremos el formato de nuestras columnas, es importante mencionar que las columnas se deben llamar igual que en nuestra tabla de la Base de Datos.

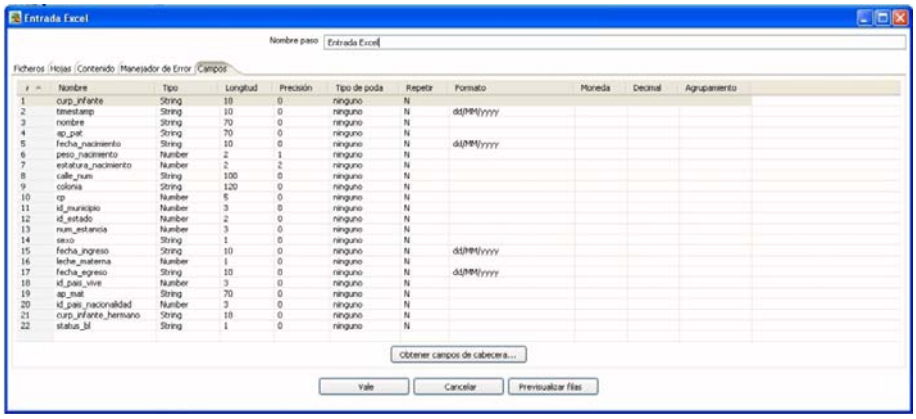


Fig.1.67. Ejemplo Carga de datos empleando Pentaho

Una vez definida la entrada, ahora procedemos a definir la salida, en este caso una tabla de la Base de Datos, hacemos doble clic y elegimos la Conexión llamada **Oracle**, el esquema destino llamado **estancia**, y la tabla destino llamada **INFANTES.**

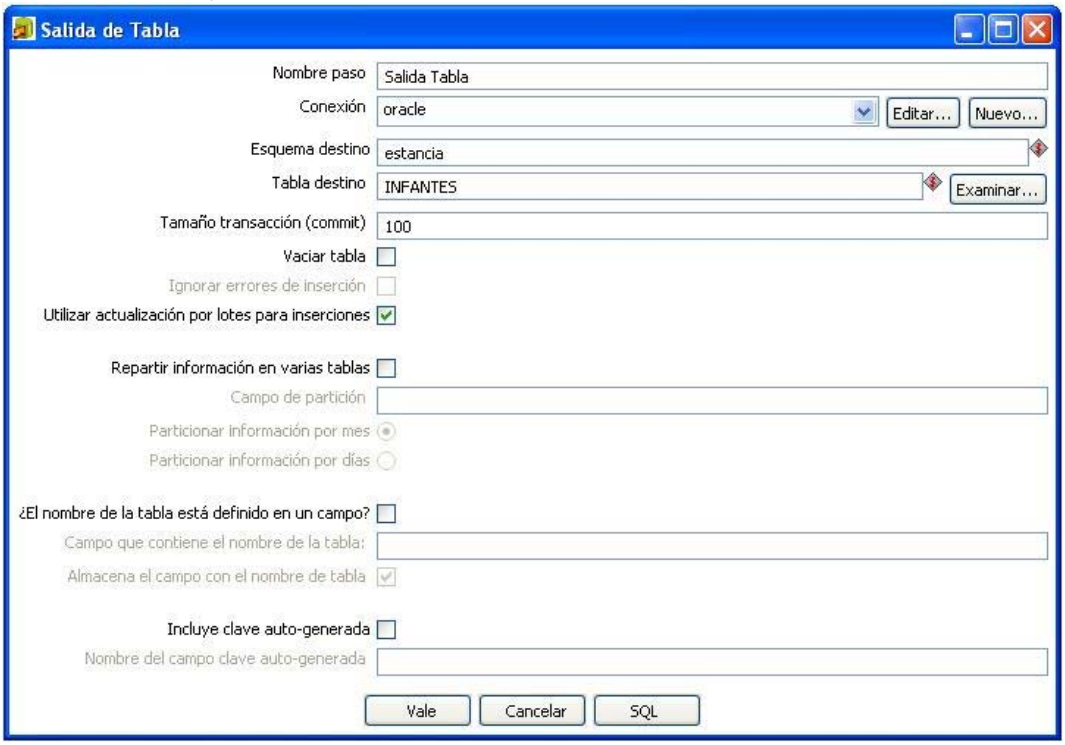


Fig.1.68. Ejemplo Carga de datos empleando Pentaho

Teniendo definidas la entrada y salida de nuestra transformación hacemos clic en Play

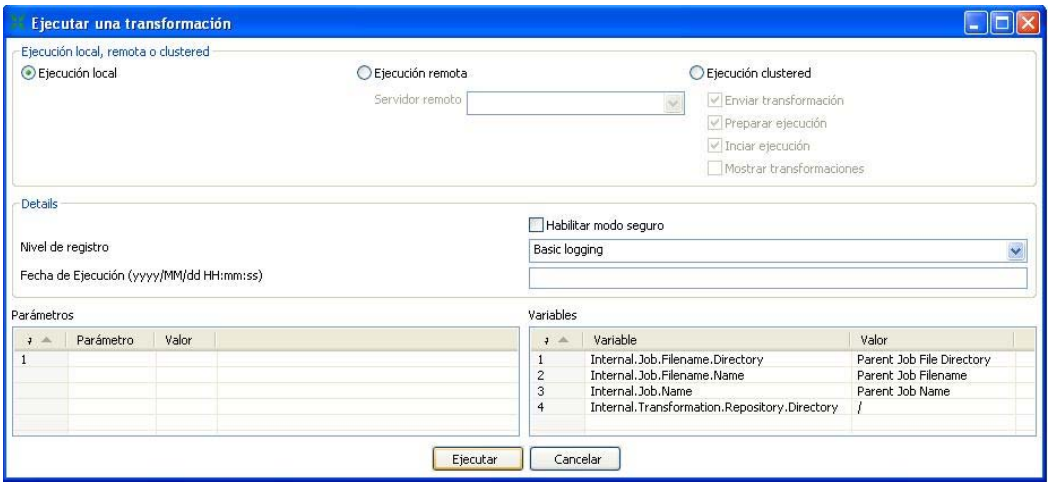


Fig.1.69. Ejemplo Carga de datos empleando Pentaho

Y Luego en ejecutar, el Log nos indicará cuando haya finalizado, y para comprobar podemos hacer un querie seleccionando los datos de la tabla Infantes de nuestra Base de Datos para verificar la carga de datos.

```
SQL> select count(*) from infantes;

  COUNT(*)
----------
      2167
```

Fig.1.70. Comprobación de Carga de datos empleando Pentaho

De la misma manera procedemos para cargar los datos de todas las tablas de nuestra Base de Datos Estancia.

ALGEBRA RELACIONAL

Es un lenguaje de consulta que sirve para obtener la información y familiarizarse con la teoría de conjuntos en las Bases de Datos.

Consta de 5 operaciones fundamentales:

- Elegir σ
- Proyectar Π
- Producto Cartesiano X
- Unión U
- Diferencia de conjuntos -

Y 2 operaciones adicionales:

- Intersección ∩
- Producto Natural (X)

Se permiten operadores

(diferente) <> = ≥ ≤ < > Λ(AND) V(OR)

Producto Cartesiano X

Una expresión en Álgebra Relacional se puede construir con subexpresiones.
Sean E_1 y E_2 expresiones de Álgebra Relacional

E_1 U E_2

E_1 - E_2

E_1 X E_2

$\sigma p(E_1)$ donde p = Predicado con atributos de E_1

$\Pi s (E_1)$ donde s Lista de atributos que aparecen de E_1

OPERADORES ADICIONALES.∩ y (X)

Ejemplo:

Obtener a todos los padres o tutores e infantes donde el nivel sea preescolar 1 y 2.

```
Π PADRES_TUTORES.nombre, INFANTES.nombre
 σ PADRES_TUTORES.curp            = PADRES_INFANTE.curp_padre
 ^ PADRES_INFANTE.curp_infante    = INFANTES.curp_infante
 ^ INFANTES_NIVEL.nivel           = NIVELES.nivel
 ^ NIVELES.nivel                  = 1
            (PADRES_TUTORES X INFANTES X INFANTES_NIVEL)
∪

 Π PADRES_TUTORES.nombre, INFANTES.nombre
 σ PADRES_TUTORES.curp            = PADRES_INFANTE.curp_padre
 ^ PADRES_INFANTE.curp_infante    = INFANTES.curp_infante
 ^ INFANTES_NIVEL.nivel           = NIVELES.nivel
 ^ NIVELES.nivel = 2
            (PADRES_TUTORES X INFANTES X INFANTES_NIVEL)
```

A continuación, procedemos hacer algunas consultas o Queries para practicar:

Uniones Desiguales.

Una condición de unión puede especificar cualquier relación entre columnas, no exactamente igual (=). Una condición de unión que especifica una relación diferente a igual es llamada unión desigual. Por ejemplo, se puede escribir una condición de unión como esta:

X.PESO_NACIMIENTO > **Y.PESO_NACIMIENTO**

La cual une cada renglón en Y a los renglones en X cuyo PESO sea mayor.

Suponga que se quiere encontrar todos los infantes que pesen más que el infante cuyo Curp es: AUPV080405HDFGDC09:

```
SQL> SELECT X.CURP_INFANTE, X.PESO_NACIMIENTO, Y.CURP_INFANTE,
Y.PESO_NACIMIENTO
FROM INFANTES X, INFANTES Y
WHERE X.PESO_NACIMIENTO > Y.PESO_NACIMIENTO
AND Y.CURP_INFANTE = 'AUPV080405HDFGDC09' AND X.NUM_ESTANCIA=44;
```

CURP_INFANTE	PESO_NACIMIENTO	CURP_INFANTE	PESO_NACIMIENTO
TOWF620711HMCRSB05	3,9	AUPV080405HDFGDC09	3,4
FUMP620413HDFNNZ08	3,8	AUPV080405HDFGDC09	3,4
COMZ620324MDFSNR00	3,8	AUPV080405HDFGDC09	3,4
HEVA620407MMCVLL08	3,6	AUPV080405HDFGDC09	3,4
EOCZ620404MMCLRR08	3,6	AUPV080405HDFGDC09	3,4
HEAD600802HMCRYR05	3,9	AUPV080405HDFGDC09	3,4
HEVS601004HMCRCL04	3,5	AUPV080405HDFGDC09	3,4
UACB600826HMCGLR09	3,6	AUPV080405HDFGDC09	3,4
FIMY590831HMCGRR05	3,5	AUPV080405HDFGDC09	3,4
UICV590511HDFRLL09	3,5	AUPV080405HDFGDC09	3,4
HEMB590420HDFRDL06	3,6	AUPV080405HDFGDC09	3,4

...191 filas seleccionadas.

En este ejercicio la tabla **INFANTE es unida** así misma **usando el operador de comparación (> mayor que) (.PESO_NACIMIENTO** > **Y.PESO_NACIMIENTO).**

Uniones Externas (+). (Outer join)

Si un renglón en una de las tablas no satisface la condición de unión, el renglón ordinariamente no aparecerá en el resultado de la consulta. Para eso, utilice el operador de unión exterior (OUTER), un signo más encerrado en paréntesis **(+)**.

Para unir EMPLEADOS y ESTANCIA y la lista de estancias 39 y 44, con ó sin **empleados con puesto de educadoras (tipo_emp =9), teclee:**

```
SQL> SELECT E.num_estancia, E.nombre, EMP.curp_empleado
     FROM ESTANCIAS E, EMPLEADOS EMP
     WHERE E.num_estancia = EMP.num_estancia(+)
     AND   E.num_estancia IN (39, 44) AND EMP.tipo_emp=9
     ORDER BY E.num_estancia;
```

NUM_ESTANCIA	NOMBRE	CURP_EMPLEADO
39	39	HITF441008MDFDRR06
39	39	GUPO610908HDFTLS05
39	39	GIGM790905MDFMMY06
39	39	JIGM790905MDFMMY03
39	39	GAOJ790617MDFRLN07
44	44	MACS850606MDFLSN03
44	44	EIVM900508MDFLLR04
44	44	DEEL891217MDFLCC06
44	44	LERJ880425MDFYYZ03

...21 filas

Como vemos en la estancia 39 sólo hay 5 educadoras a diferencia de la estancia 44 en la que hay 16.
Así, el símbolo de unión exterior es usado después de **EMP. num_estancia** en la cláusula **WHERE.**
SQL*Plus une un renglón nulo de **EMP** a cualquier renglón de **ESTANCIAS** que no pueda ser unido a un renglón real de **EMP.**
Si se omiten los paréntesis, la multiplicación será realizada antes de la suma.

FUNCIONES ARITMÉTICAS.

FUNCIÓN	EJEMPLO	RESULTADO
ABS	**ABS(SAL)**	**Valor absoluto de Salario**
GREATEST	**GREATEST(SAL,COMM)**	**Cuál es mayor entre SAL y COMM**
LEAST	**LEAST(SAL,COMM)**	**Cuál es más pequeño entre SAL y COMM**
ROUND	**ROUND(SAL,2)**	**Redondea SAL a 2 dígitos después del punto decimal**
TO_NUMBER	**TO_NUMBER(GRADE)**	**Convierte CHAR a NUMBER**
TRUNC	**TRUNC(SAL,2)**	**Trunca SAL a 2 dígitos**

Ejemplos de Funciones Aritméticas.

Para calcular el salario diario de empleados en la estancia 44

```
SQL> SELECT EMP.curp_empleado, ROUND (T.sueldo/30,2)
        FROM EMPLEADOS EMP, TIPO_EMPLEADO T
        WHERE EMP.num_estancia=44
         AND EMP.tipo_emp = T. tipo_emp;
```

```
CURP_EMPLEADO          ROUND(T.SUELDO/30,2)
------------------     --------------------
LOLV660430MDFPMR02      100
CAOA600716MDFRRN12      233,33
MERR550524MDFNBT05      133,33
ORSM621031MDFRNR05      133,33
VIPG580908HDFLLL03      100
SAMM710203MDFNYN03      116,67
MOPL800811MDFNRL08      116,67
SAQL600515MDFNNS01      133,33
PUVG820927MDFLLL03      133,33
PEZE591103MDFRTL02      166,67
MASB631031MDFRRL05      166,67
```

…33 filas seleccionadas.

Para calcular el total de niñas y niños en la estancia en la estancia numero 44:

```
SQL> SELECT count(I.curp_infante) as TotalNinas
        FROM infantes I
        WHERE I.sexo='m'
        AND num_estancia=44;
TOTALNINAS
----------
     403

SQL> SELECT count(I.curp_infante) as TotalNinos
        FROM infantes I
        WHERE I.sexo='h'
        AND num_estancia=44;
TOTALNINOS
----------
     319
```

Truncación de un Número.
La función **TRUNC** trunca el número en su parte decimal.
Constantes **CHAR.**

Para sumar constantes Char, teclee:

```
SQL> COLUMN Y HEADING ' ';
SQL> COLUMN Z HEADING ' ';
SQL> SELECT 'Empleado' Y, curp_empleado, 'reporta a' Z, curp_jerarquia_empleado
       FROM EMPLEADOS;
```

	CURP_EMPLEADO		CURP_JERARQUIA_EMP
Empleado	ESLV780523HDFSPC04	reporta a	NISJ830418HDFTNR02
Empleado	GUPI680928MDFTLL03	reporta a	MERO690518MDFSSL02
Empleado	LECC820512MDFDBR02	reporta a	GUPI680928MDFTLL03
Empleado	SEJE730103MDFRRL01	reporta a	MERO690518MDFSSL02
Empleado	MOSA760730HDFNNB01	reporta a	NISJ830418HDFTNR02
Empleado	FEFM821030MDFRN01	reporta a	MERO690518MDFSSL02
Empleado	PACJ840515HDFLRN01	reporta a	FEFM821030MDFRN01
Empleado	LIOL790716MDFVRS06	reporta a	SEJE730103MDFRRL01
Empleado	ZUMM720418MDFBRN01	reporta a	SEJE730103MDFRRL01
Empleado	REOA680519HDFYCN02	reporta a	NISJ830418HDFTNR02
Empleado	ESGH760915MDFYSR03	reporta a	REOA680519HDFYCN02

…99 filas seleccionadas.

Note el uso del comando **COLUMN** para eliminar cualquier encabezado en las columnas constantes.
Formatos de fechas.

Los valores date son normalmente desplegados en un formato estándar: 12-JAN-82
Este formato es escrito 'DD-MON-YY'.

Despliegue de fechas de nacimiento de los niños de la estancia no. 44:

```
SQL> SELECT fecha_nacimiento,curp_infante
       FROM INFANTES
       WHERE num_estancia = 44
        AND fecha_ingreso > '06/08/2007';
```

FECHA_NA	CURP_INFANTE
07/08/07	AICE070807HDFVMD06
12/12/07	VASS071212HDFLLL03
03/12/07	PAVM071203MDFDLL09
05/10/07	ZERA071005MMCTZR08
15/12/07	IIMS071215MMCLNR00
04/11/07	GUCY071104MDFRBN08
15/10/07	LASI071015MDFRRV05
08/12/07	VASS071208MMCLNF04

…30 filas seleccionadas.

Liste en orden descendente:

```
SQL> SELECT fecha_nacimiento, curp_infante
        FROM INFANTES
        WHERE num_estancia = 44
        AND fecha_ingreso > '06/08/2007'
        ORDER BY fecha_nacimiento DESC;
```

FECHA_NA	CURP_INFANTE
15/10/07	LASI071015MDFRRV05
05/10/07	ZERA071005MMCTZR08
22/08/07	DEHJ070822HMCLRS08
11/08/07	MARF070811HDFRCR04
07/08/07	AICE070807HDFVMD06
01/08/07	OOIJ070801HMCRNN01
07/07/07	RIGR070707HMCSLB09
04/07/07	MACE070704HMCRHN09

…30 filas seleccionadas.

"Las entradas nulas aparecen primero."

Valores Nulos en Expresiones y Funciones.

Cuando una expresión o función matemática individual se refiere a una columna que contiene un valor nulo, el resultado también es nulo.

Para encontrar todos los Directores, teclee:

```
SQL> SELECT E.curp_empleado, T.nombre_tipo_emp
        FROM EMPLEADOS E, TIPO_EMPLEADO T
        WHERE E.Tipo_emp=T.tipo_emp
         AND T.nombre_tipo_emp= 'director';
```

CURP_EMPLEADO	NOMBRE_TIPO_EMP
RODP550814MDFDRR00	director
RISS751227HDFVNR00	director
MERO690518MDFSSL02	director

3 filas seleccionadas.

Puede obtener el mismo resultado con una consulta usando una subconsulta en la cláusula WHERE para encontrar el empleo de JONES.

Siempre se encierran las subconsultas entre paréntesis.

Una subconsulta puede contener otra subconsulta**.**

Para listar los infantes en la estancia 39 con la misma fecha de ingreso que la estancia 44

```
SQL> SELECT curp_infante, fecha_ingreso
        FROM INFANTES
        WHERE num_estancia = 44
        AND fecha_ingreso IN
            (SELECT fecha_ingreso
             FROM INFANTES
             WHERE num_estancia =
                        (SELECT num_estancia
                         FROM ESTANCIAS
                        WHERE nombre = '39'));
```

CURP_INFANTE	FECHA_INGR
HEGM980502MMCRNG05	10/08/1998
OOVE980114MDFRRL04	10/08/1998
IAHG980605MDFBRR05	10/08/1998
TORP080405HDFRYD08	10/08/1998
IALG990111MDFSPD02	10/08/1998
GOJH970724HDFNRR00	10/08/1998
GATG970730HDFLRD05	10/08/1998
MOGJ970606HMCNSR04	10/08/1998
GUFA970718HDFTLX06	10/08/1998
TIMI970727HDFRXV06	10/08/1998
PAPA970621HDFTNG07	10/08/1998

…**150 filas seleccionadas.**

Empleados y sus dependencias en forma de árbol.

```
SQL> SELECT A.nombre_emp, A.curp_empleado,
        B.nombre_tipo_emp,A.curp_jerarquia_empleado
        FROM empleados A, tipo_empleado B
        WHERE A.tipo_emp=B.tipo_emp
        AND A.num_estancia in(39,44)
        CONNECT BY PRIOR curp_empleado=curp_jerarquia_empleado
        START WITH B.nombre_tipo_emp ='director';
```

NOMBRE_EMP	CURP_EMPLEADO
NOMBRE_TIPO_EMP	CURP_JERARQUIA_EMP
Sergio	RISS751227HDFVNR00
director	
Mayte	JIGM790905MDFMMY03
educador	RISS751227HDFVNR00
Mayte	GIGM790905MDFMMY06
educador	RISS751227HDFVNR00
Oscar	GUPO610908HDFTLS05
educador	RISS751227HDFVNR00
Fernanda	HITF441008MDFDRR06
educador	RISS751227HDFVNR00
Martha	AAVM500407MDFRLR00
educador	RISS751227HDFVNR00

…66 filas seleccionadas.

LPAD(cad1,n,cad2) Devuelve cad1 con longitud n, y ajustada a la derecha, rellenando por la izquierda con cad2.
Operadores de Cadenas de Caracteres: || que representa concatenación.

```
SQL> Set lines 1000
SQL> Set pages 200
SQL> SELECT LPAD('  ',2*LEVEL)||A.nombre_emp 'EMPLEADO', B.nombre_tipo_emp  'NOMBRE_TIPO_EMP'
  FROM   empleados  A, tipo_empleado B
  WHERE A.tipo_emp=B.tipo_emp
  AND A.num_estancia =44
  CONNECT BY PRIOR curp_empleado=curp_jerarquia_empleado
  START WITH A.nombre_emp ='Perla';
```

```
--------------------------------------------------------------------------------
EMPLEADO NOMBRE_TIPO_EMP
---------- --------------------------------------------------------------
  Perla              EMPLEADO director
   Margarita         EMPLEADO educador
  Graciela           EMPLEADO educador
   Margarita         EMPLEADO educador
   Beatriz           EMPLEADO educador
   Susana            EMPLEADO educador
    Aurora           EMPLEADO educador
    Claudia          EMPLEADO educador
    Marisol          EMPLEADO educador
    Cinthia          EMPLEADO educador
    Susana           EMPLEADO educador
    Leticia          EMPLEADO educador
    Jennifer         EMPLEADO educador
    Jazmín           EMPLEADO educador
    Lucila           EMPLEADO educador
    Maru             EMPLEADO educador
    Sandra           EMPLEADO educador
    Leticia          EMPLEADO pedagogo
     Socorro         EMPLEADO psicologo
    Ignacio          EMPLEADO dietista
    María            EMPLEADO intendencia
    Silvia           EMPLEADO intendencia
    Verónica         EMPLEADO intendencia
    Gilberto         EMPLEADO mantenimiento
      Elena          EMPLEADO cocinero
     Belén           EMPLEADO cocinero
    Ana María        EMPLEADO doctor
      Rita           EMPLEADO enfermero
    Martha           EMPLEADO trabajo social
      Lissette       EMPLEADO administrativo
      Gloria         EMPLEADO administrativo
    Mónica           EMPLEADO seguridad
      Lilia          EMPLEADO seguridad

33 filas seleccionadas.
```

Update Modificar los datos de una tabla. Y Commit Confirmar como permamentes las modificaciones realizadas.

UPDATE tabla SET {columna = expresión,}+ [WHERE condición];

```
SQL> UPDATE tipo_empleado
        SET sueldo = 9000
        WHERE nombre_tipo_emp = 'director';
        COMMIT;
1 fila actualizada.
```

Delete Eliminar filas de datos de una tabla.

```
SQL> DELETE recetas
        WHERE num_secuencial_expediente='22316';
1 fila suprimida.
```

Rollback Deshacer todas las modificaciones realizadas desde la última confirmación**.**

```
SQL> ROLLBACK;
Rollback terminado.
```

6. Construcción de la aplicación

Se ha elegido una función (**Reportar niños por nivel académico 4.1.4**) para aplicarla en un formulario:
Aquí la pantalla representativa programada utilizando en lenguaje de PHP:

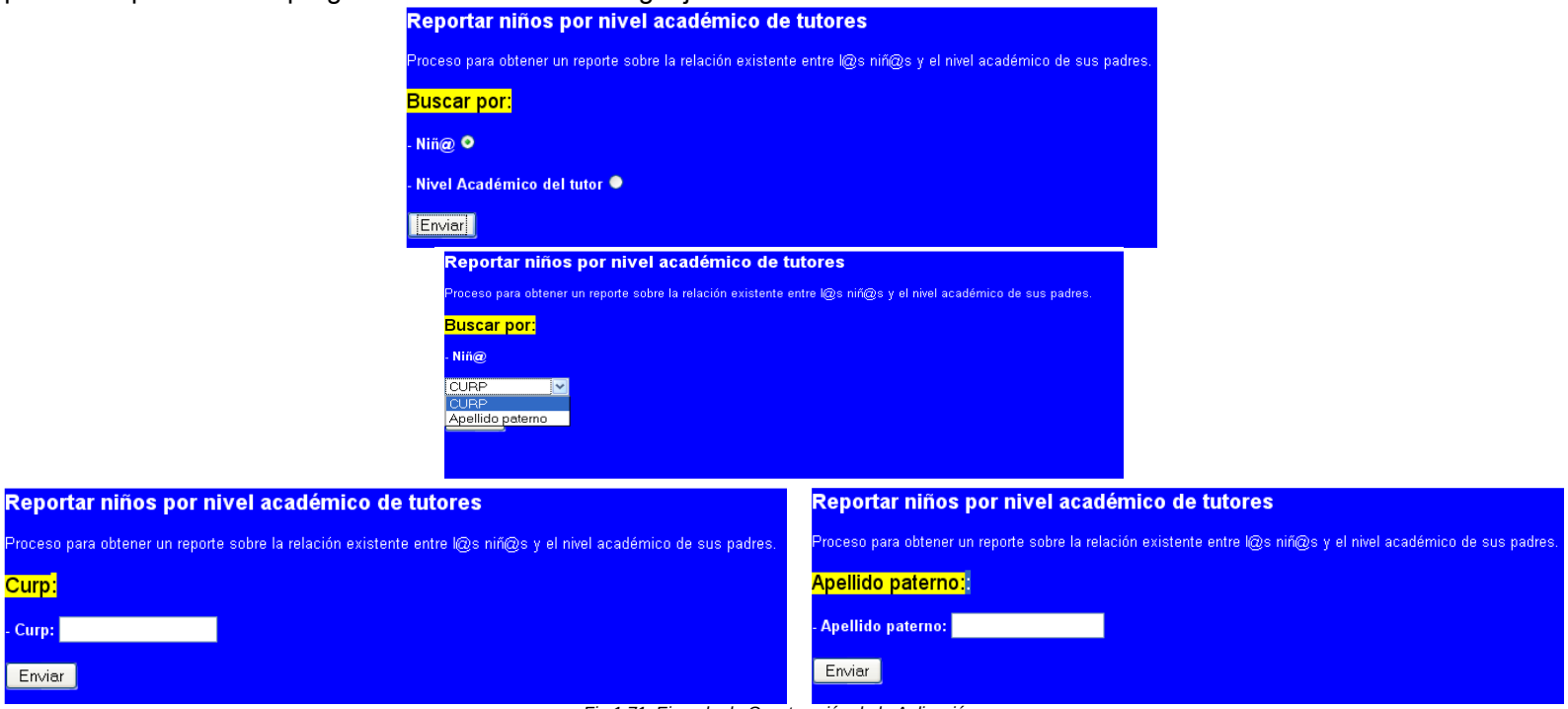
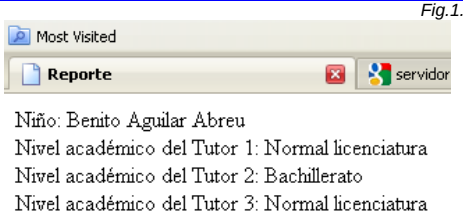
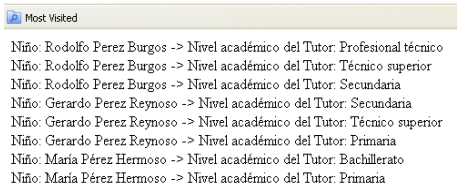


Fig.1.71. Ejemplo de Construcción de la Aplicación





Éste es el código que se utilizó para comunicarnos con la Bases de datos, para dicha función:

```
<html>
<head>
<title>Reporte</title>
</head>
<body>

<?
// Conexión con la  Base  de Datos
$conexion = ora_logon ('estancia', 'estancia');   // Aquí especificamos los parámetros usuario y contraseña
$miCursor = ora_open($conexion); // Declaramos el cursor que recorrerá las filas de nuestro query

// Si el usuario elige Curp en el Filtro de búsqueda, se necesitan extraer los datos con el siguiente query.
if(isset($_POST["formulario_curp"])){
$sql = 'SELECT distinct infantes.nombre,
                infantes.ap_pat,
                infantes.ap_mat,
                padres_tutores.nombre,
                padres_tutores.ap_pat,
                padres_tutores.ap_mat,
                 padres_tutores.nivel_academico
        FROM    infantes,
                padres_tutores,
                tutores_infante
        WHERE infantes.curp_infante like '".$_POST["texto_curp"]."'
        AND     infantes.curp_infante=tutores_infante.curp_infante
        AND     tutores_infante.curp_padre=padres_tutores.curp';

                // Query en el que seleccionamos los datos
                // del infante teniendo su curp como referencia

ora_parse($miCursor, $query);
ora_exec($miCursor);
ora_fetch($miCursor);

echo 'Niño: '.ora_getcolumn($miCursor, '1').' ';
echo ora_getcolumn($miCursor, '2').' ';
echo ora_getcolumn($miCursor, '3').'<br>';
echo 'Nivel académico del Tutor 1: '.ora_getcolumn($miCursor, '8').'<br>';
$i=2;
while(ora_fetch($miCursor)){
  echo 'Nivel académico del Tutor '.$i.': '.ora_getcolumn($miCursor, '8').'<br>';
  $i++;
}

}
// Si el usuario elige Apellido Paterno en el Filtro de búsqueda, se necesitan extraer los datos con el
//siguiente query.
if(isset($_POST["formulario_apellido"])){

$sql = 'SELECT  distinct infantes.nombre,
                infantes.curp_infante,
                infantes.ap_pat, infantes.ap_mat, padres_tutores.nivel_academico
        FROM    infantes, padres_tutores, tutores_infante }
        WHERE infantes.ap_pat like '".$_POST["texto_apellido"]."'
        AND     infantes.curp_infante=tutores_infante.curp_infante
        AND     tutores_infante.curp_padre=padres_tutores.curp';
        // Query en el que seleccionamos los datos
        // del infante teniendo su apellido como referencia
ora_parse($miCursor, $query);
ora_exec($miCursor);
ora_fetch($miCursor);

while(ora_fetch($miCursor)){
  echo 'Niño: '.ora_getcolumn($miCursor, '1').' ';
  echo ora_getcolumn($miCursor, '3').' ';
```

```
  echo ora_getcolumn($miCursor, '4').' -> ';
  echo 'Nivel académico del Tutor: '.ora_getcolumn($miCursor, '5').'<br>';
}
}
if(isset($_POST["niño"])){ // Si el usuario eligió reportar por niño (primera opción)

// Y si Además eligio Curp en el Filtro de búsqueda
if ($_POST["niño"]=="CURP"){
echo '<body>

<div align="center">
        <table border="0" cellpadding="0" cellspacing="0" width="900" bgcolor="#0000FF"
id="tabla_principal">
                <tr>
                        <td>
                        <p class="MsoNormal"><b><span style="font-family: Arial">
                        <font color="#FFFFFF" size="4">Reportar niños por nivel académico de
                        tutores</font></span></b></p>
                        <p class="MsoNormal"><font color="#FFFFFF" size="2">
                        <span style="font-family: Arial">Proceso para obtener un reporte
                        sobre la relación existente entre l@s niñ@s y el nivel académico de
                        sus padres.</span></font></p>
                        <p class="MsoNormal"><span style="background-color: #FFFF00">
                        <font face="Arial" size="4">Curp</font></span><font face="Arial"
size="4"><span style="background-color: #FFFF00">:</span></font></p>
                        <form method="POST" action="reporteMySQL.php">
                                <p class="MsoNormal">
                                <font face="Arial" style="font-weight: 700" color="#FFFFFF" size="2">
                                - Curp: <input type="text" name="texto_curp" size="18"
maxlength="18"></font></p>
        // Obteniendo el curp que escribe el usuario y lo guardamos en la variable texto_curp
                                <p class="MsoNormal">
                                <input type="hidden" name="formulario_curp" size="18"
maxlength="18">
                                <input type="submit" value="Enviar" name="B1"></p>
                        </form>
                        <p class="MsoNormal"> </p>
                        <p></td>
                </tr>
        </table>
</div>

</body>';
}
// O si Además eligió Apellido Paterno en el Filtro de búsqueda

if ($_POST["niño"]=="Apellido paterno"){
echo '
<body>

<div align="center">
        <table border="0" cellpadding="0" cellspacing="0" width="900" bgcolor="#0000FF"
id="tabla_principal">
                <tr>
                        <td>
                        <p class="MsoNormal"><b><span style="font-family: Arial">
                        <font color="#FFFFFF" size="4">Reportar niños por nivel académico de
                        tutores</font></span></b></p>
                        <p class="MsoNormal"><font color="#FFFFFF" size="2">
                        <span style="font-family: Arial">Proceso para obtener un reporte
                        sobre la relación existente entre l@s niñ@s y el nivel académico de
                        sus padres.</span></font></p>
                        <p class="MsoNormal"><span style="background-color: #FFFF00">
                        <font face="Arial" size="4">Apellido paterno:</font></span><font face="Arial"
size="4"><span style="background-color: #FFFF00">:</span></font></p>
                        <form method="POST" action="reporteMySQL.php">
                                <p class="MsoNormal">
```

```
                                        <font face="Arial" style="font-weight: 700" color="#FFFFFF" size="2">
                                        - Apellido paterno: <input type="text" name="texto_apellido"
size="18" maxlength="18"></font></p>
        // Obteniendo el apellido paterno que escribe el usuario y lo guardamos en la variable
        //texto_apellido

                                        <p class="MsoNormal">
                                        <input type="hidden" name="formulario_apellido" size="18"
maxlength="18">
                                        <input type="submit" value="Enviar" name="B1"></p>
                                </form>
                                <p class="MsoNormal"> </p>
                                <p></td>
                        </tr>
        </table>
</div>

</body>
';

}

}

// Si el usuario eligió reportar por Nivel académico (segunda opción)
if(isset($_POST["nivel"])){
echo $_POST["nivel"];
}
// Formularios para cada opción.(1 y 2)
// Si eligió la opción 1: Buscar por Niño

if(isset($_POST["op"]) and $_POST["op"]==1){
echo '<body>

<div align="center">
        <table border="0" cellpadding="0" cellspacing="0" width="900" bgcolor="#0000FF"
id="tabla_principal">
                <tr>
                        <td>
                        <p class="MsoNormal"><b><span style="font-family: Arial">
                        <font color="#FFFFFF" size="4">Reportar niños por nivel académico de
                        tutores</font></span></b></p>
                        <p class="MsoNormal"><font color="#FFFFFF" size="2">
                        <span style="font-family: Arial">Proceso para obtener un reporte
                        sobre la relación existente entre l@s niñ@s y el nivel académico de
                        sus padres.</span></font></p>
                        <p class="MsoNormal"><font face="Arial" size="4">
                        <span style="background-color: #FFFF00">Buscar por:</span></font></p>
                        <form method="POST" action="reporteMySQL.php">
                                <p class="MsoNormal">
                                <font face="Arial" style="font-weight: 700" color="#FFFFFF" size="2">
                                - Niñ@</font></p>
                                <p class="MsoNormal"><select size="1" name="niño">
                                <option>CURP</option>
                                <option>Apellido paterno</option>
                                </select></p>
                                <p class="MsoNormal">
                                <input type="submit" value="Enviar" name="B1"></p>
                        </form>
                        <p class="MsoNormal"> </p>
                        <p></td>
                </tr>
        </table>
</div>

</body>';
```

```
}
//Si eligió la opción 2: Buscar por Nivel Académico del Tutor

if(isset($_POST["op"]) and $_POST["op"]==2){

echo '
<body>

<div align="center">
	<table border="0" cellpadding="0" cellspacing="0" width="900" bgcolor="#0000FF"
id="tabla_principal">
		<tr>
			<td>
			<p class="MsoNormal"><b><span style="font-family: Arial">
			<font color="#FFFFFF" size="4">Reportar niños por nivel académico de
			tutores</font></span></b></p>
			<p class="MsoNormal"><font color="#FFFFFF" size="2">
			<span style="font-family: Arial">Proceso para obtener un reporte
			sobre la relación existente entre l@s niñ@s y el nivel académico de
			sus padres.</span></font></p>
			<p class="MsoNormal"><font face="Arial" size="4">
			<span style="background-color: #FFFF00">Buscar por:</span></font></p>
			<form method="POST" action="reporteMySQL.php">
				<p class="MsoNormal">
				<font face="Arial" style="font-weight: 700" color="#FFFFFF" size="2">
				- Nivel Académico del tutor:</font></p>
				<p class="MsoNormal"><select size="1" name="nivel">
				<option>Primaria</option>
				<option>Secundaria</option>
				<option>Bachillerato</option>
				<option>Normal licenciatura</option>
				<option>Profesional técnico</option>
				<option>Técnico superior</option>
				<option>Licenciatura universitaria y tecnológica</option>
				<option>Posgrado</option>
				</select></p>
				<p class="MsoNormal">
				<input type="submit" value="Enviar" name="B1"></p>
			</form>
			<p class="MsoNormal">&amp;nbsp;</p>
			<p></td>
		</tr>
	</table>
</div>

</body>';

}
// Formulario Principal
if(!$HTTP_POST_VARS){
echo '
<body>

<div align="center">
	<table border="0" cellpadding="0" cellspacing="0" width="900" bgcolor="#0000FF"
id="tabla_principal">
		<tr>
			<td>
			<p class="MsoNormal"><b><span style="font-family: Arial">
			<font color="#FFFFFF" size="4">Reportar niños por nivel académico de
			tutores</font></span></b></p>
			<p class="MsoNormal"><font color="#FFFFFF" size="2">
			<span style="font-family: Arial">Proceso para obtener un reporte
			sobre la relación existente entre l@s niñ@s y el nivel académico de
			sus padres.</span></font></p>
			<p class="MsoNormal"><font face="Arial" size="4">
```

```
                        <span style="background-color: #FFFF00">Buscar por:</span></font></p>
                        <form method="POST" action="reporteMySQL.php">
                                <p class="MsoNormal">
                                <font face="Arial" style="font-weight: 700" color="#FFFFFF" size="2">
                                - Niñ@</font><input type="radio" value="1" checked name="op"></p>
                                <p class="MsoNormal">
                                <font face="Arial" style="font-weight: 700" color="#FFFFFF" size="2">
                                - Nivel Académico del tutor</font><input type="radio" name="op"
value="2"></p>
                                <p class="MsoNormal">
                                <input type="submit" value="Enviar" name="B1"></p>
                        </form>
                        <p class="MsoNormal"> </p>
                        <p></td>
                </tr>
        </table>
</div>

</body>';
}

ora_close($miCursor); // cerramos el cursor que recorrió los datos obtenidos.
ora_logoff($conexion); // Nos desconectamos de la Base de Datos.
?>
</body>
</html>
```

CONCLUSIÓN CAPÍTULO 1:

Este capítulo fungió como la base fundamental de la que se compone la teoría de Bases de datos y las metodologías CASE y RUP para la construcción de una aplicación.
Nos resultó interesante y altamente educativo trabajar desde la idea de la concepción de una Base de Datos, su diseño y posterior implementación. De esta manera pudimos dejar más que en claro muchos conceptos que dentro del aula tal vez se obviaron o se tomaron a la ligera, pero que una vez que tenemos un proyecto real en nuestras manos, nos damos cuenta que todo detalle resulta de gran importancia para llevar a cabo cualquier proyecto.

Consideramos que es muy importante tener bien claros dichos conceptos y principios pues cualquier proyecto en el que esté involucrada una Base de Datos, implicará un análisis como el planteando aquí, y esta guía será de gran utilidad, pues además utilizamos diferentes herramientas de software que nos ayudarán sin duda, ya sea para práctica, o para elaborar algún proyecto específico.

Capítulo 2
Bases de Datos Avanzadas

Introducción

Este capítulo tiene como objetivo ejemplificar los principales conceptos.

2.1 Fundamentos de Sistemas Operativos
2.2 Entorno Cliente / Servidor
2.3 Administración de Bases de Datos
2.4 Seguridad de Bases de Datos
2.5 Afinación
2.6 Bases de Datos en aplicaciones basadas en internet

2.1 Fundamentos de Sistemas Operativos.

Comprenderemos los conceptos básicos en Sistemas operativos, conociendo las plataformas que existen.

2.2 Ambiente cliente servidor
Explicaremos los conceptos básicos para la configuración del ambiente de trabajo necesario en un entorno cliente/servidor.

Entorno Cliente / Servidor.
Aquí ilustraremos las funciones que están disponibles en los sistemas cliente/servidor tanto en el lado del DBMS o servidor como del lado de la aplicación o cliente.
Al final del tema serás capaz de:

- Comprender la división del trabajo que es posible hacer con el enfoque cliente / servidor y ver cómo esa división beneficia a las organizaciones porque permite una utilización completa de la capacidad de sus sistemas.
- Definir la base de datos en el DBMS servidor, utilizando muchas capacidades potentes.
- Utilizar el DML para escribir programas que saquen ventaja del poder único de los conceptos de las Bases de Datos relacionales
- Comprender los entornos modernos de aplicación en el cual se producen sistemas clientes que pueden interactuar con DBMS servidores.

2.3 Administración de Bases de Datos Ejemplificaremos los conceptos básicos en la administración de las Bases de Datos y la preparación del servidor para comenzar a trabajar con la aplicación. También explicaremos las tareas de un administrador de Bases de Datos, así como la manera de llevarlas a cabo dentro del DBMS.

Rol del Administrador de las Bases de Datos (DBA)
Conoceremos quien es el DBA y las tareas que le conciernen, así como los privilegios que le corresponden.

Operaciones con la Base de Datos
Aquí crearemos y montaremos la Base de Datos, determinaremos el tamaño y control de las transacciones, así como la concurrencia y consistencia de nuestra Base de Datos.

Espacios de Tablas y Segmentos
Determinaremos los espacios de tablas, el espacio inicial llamado System, así como los diferentes segmentos.

Objetos de Usuarios de la Base de Datos
Hablaremos de los diferentes objetos como son tablas, vistas, agrupamientos, índices, secuencias, triggers y procedimientos.

2.4 Seguridad de Bases de Datos Expondremos los conceptos básicos de seguridad y su importancia dentro de los DBMS para la conservación de la integridad física de los datos. También identificaremos las principales vulnerabilidades en seguridad y reforzaremos está tomando las medidas necesarias para su corrección y total aprovech6amiento del DBMS.

Integridad de datos
Conoceremos los tipos de integridad.

Administración de Usuarios
Sabremos cómo crear, modificar y eliminar usuarios, hablaremos de perfiles y sesiones.

2.5 Desempeño y afinación. Mejoraremos el desempeño de nuestro DBMS. También realizaremos la optimización de las tareas realizadas por éste.

2.6 Bases de Datos en aplicaciones basadas en internet. Desarrollaremos nuestra aplicación de Bases de Datos, la cual como se mencionamos antes sólo será programada en un 10%.

2.1 Fundamentos de Sistemas Operativos

Instalación de Sistema Operativo

Conceptos previos:

Las *Funciones de un Sistema Operativo* son presentar al usuario el equivalente de una máquina extendida o máquina virtual que sea más fácil de programar que el software subyacente, llevar un registro de la utilización de los recursos, dar paso a solicitudes de recursos, llevar la cuenta de su uso y mediar entre la solicitudes en conflicto de los distintos programas y usuarios.

Un *archivo* es un conjunto de caracteres o bytes.

Un *proceso* es un programa en ejecución.

Un intérprete de comandos (*shell*) es un proceso que lee los comandos de usuario desde una terminal y crea procesos hijo para ejecutar ese comando.

El *kernel* Lleva un registro de la utilización de los recursos (disco, memoria y procesador), da paso a solicitudes de recursos, es el mediador entre las solicitudes en conflicto de los distintos programas y usuario.

Un *directorio* es un archivo que contiene una lista de nombres de archivos junto con información adicional de esos archivos.

Sistema de archivos es una jerarquía de directorios y archivos.

UNIX	WINDOWS
Linux es un clon del sistema operativo Unix, escrito desde cero por el finlandés Linus Torvalds. El *proceso de instalación* consiste en copiar los archivos del sistema operativo al disco duro de nuestro equipo. Puede hacerse de cdrom o vía red. *GNU/Linux* dispone de un estándar que debe cumplir todo sistema operativo que pretenda ser Unix, su filosofía son comandos cortos, simples y muy eficientes que hacen una sola cosa pero la hacen muy bien, compuesto de entradas y salidas estandarizadas que permiten la interconexión de comandos. Esto se llama entubamiento (pipelining): la salida de un comando es tomada por el siguiente como entrada. Un *Firewall* es un mecanismo de control de acceso entre el equipo y la red y determina qué recursos de su equipo están accesibles para los usuarios remotos de la red. Los archivos que intervienen para dar de *alta un usuario* son: /etc/passwd. Contiene la lista de usuarios en el sistema. /etc/group. Contiene los nombres de los grupos UNIX definidos en el sistema y una lista de sus miembros. /etc/shadow/. Este archivo contiene las contraseñas cifradas *Anaconda*. Es el instalador de Linux, reconoce todos los recursos del sistema. El usuario *root* (también conocido como superusuario) posee acceso completo al sistema.	Toda una familia de sistemas operativos desarrollados y comercializados por Microsoft. Existen versiones para hogares, empresas, servidores y dispositivos móviles, como computadores de bolsillo y teléfonos inteligentes. Hay variantes para procesadores de 16, 32 y 64 bits. Los *roles de las computadoras* en red según Microsoft son: Servidores de Bases de Datos, Servidor de Correo, Servidor de Archivos, Servidor de Servicio de Directorio, Servidor de Impresión. Un sistema operativo en red permite a la computadora operar en red, proporciona servicios básicos a computadoras en red, se integra con otros sistemas operativos. El *Dominio* se administra como unidad y procedimientos comunes, existen: *Cuentas de usuario de dominio*, permiten entrar a los recursos de una máquina específica, permite el acceso a los recursos de red, reside en el Active Directory, permiten realizar tareas administrativas o tener acceso temporal a los recursos de la red. *Grupos de seguridad*, son usados para asignar permisos o como vistas de distribución grupos de distribución, Y el *ámbito de un grupo global* es usado para organizar usuarios con similares requerimientos de acceso a la red.

Administración de sistemas.

El objetivo principal del *administrador de sistemas* consiste en proporcionar y mantener acceso a los recursos del sistema.

Actividades del administrador

1 Planear las actividades

Planear las modificaciones que tenemos que hacer en el servidor.

Como tareas podemos encontrar:

- Mantenimiento de los usuarios del sistema
- Agregar o quitar hardware
- Realizar respaldos de información
- Instalación de nuevo software
- Monitoreo del sistema.
- Detección y reparación de fallas.
- Mantenimiento de documentación local.
- Seguridad
- Ayuda a los usuarios
- Responsabilidad del administrador del sistema
- La creación y eliminación de usuarios es una tarea de rutina en la administración de sistemas.

2 Conocer las utilerías del sistema (find, cron, etc.)

Un buen administrador debe de conocer las herramientas con las que cuenta para administrar un sistema unix.

3 Conocer la documentación

Es imprescindible disponer del conjunto completo de manuales e información propios de la versión de UNIX instalada. Ninguna otra fuente puede sustituir la del propio fabricante.

4 Conocer el hardware de la máquina

Mínimamente debe conocer lo siguiente:

- - Número y velocidad de los procesadores
- - Número y capacidad de los discos duros
- - Organización lógica de los discos (RAID, Mirror, etc.)
- - Número de serie e inventario
- - Marca o Modelo

Resumen de comandos básicos de Linux y Unix.

Comunicaciones

- ftp protocolo de transferencia de archivos
- login entrar al sistema unix
- mailx enviar o leer correo
- talk mandar mensajes a usuarios
- telnet protocolo inseguro para sesiones remontas
- ssh protocolo seguro para sesiones remotas
- vacation responder a un correo automáticamente
- **cmp** **compara dos archivos byte a byte**
- comm compara dos objetos en dos archivos ordenados
- **diff** **compara dos archivos línea por línea**
- **diff3** **compara 3 archivos**

Manejo de archivos

- *cat* *concatena archivos o los despliega*
- **cd** **cambia de directorio**
- chmod cambia permisos a archivos
- **cp** **copia archivos**
- file determina el tipo de archivo
- **head** **muestra las primeras líneas de un archivo**
- ln crea aliases o ligas a archivos
- **ls** **lista el contenido de directorios**
- **mkdir** **crea directorios**
- **more** **despliega archivos en forma paginada**
- **mv** **mueve o renombra archivos**
- **pwd** **muestra el directorio actual**
- **rm** **borra archivos**
- **rmdir** **borra directorios vacíos**
- **tail** **muestra las últimas líneas de un archivo**
- **wc** **cuenta líneas, palabras y caracteres**

Varios

- banner hace una especie de poster con una palabras
- bc calculadora
- cal despliega calendarios
- clear limpia pantalla
- man ayuda en línea
- nice reduce la prioridad de algún proceso
- nohup inmortaliza a un proceso
- passwd cambia la contraseña a un usuario
- script guarda una sesión
- su cambio de usuario, usualmente usado para convertirse en root

Impresión

- cancel — cancela una solicitud de impresión
- lp — imprime
- lpstat — obtiene el estado de la impresora
- lpr — da formato y pagina un archivo para impresión

Programación

- make — ejecuta comandos en un orden preestablecido
- gcc — invoca al compilador de GNU-C

Busqueda

- egrep — versión mejorada del comando grep
- fgrrep — busca por palabras en archivos
- grep — busca por patrones en archivos
- find — busca en un filesystem por archivos
- string — busca patrones en archivos binarios

Programación Shell

- echo — muestra el contenido de variables
- expr — ejecuta operaciones aritméticas
- line — lee datos de la entrada
- printf — busca en un filesystem por archivos
- sleep — hace una pausa en un proceso
- test — prueba una condición

Almacenamiento

- gunzip — descomprime archivos
- gzip — comprime archivos
- tar — empaqueta archivos

Estado del sistema

- at — ejecuta un comando a una hora programada
- **chgrp** — **cambia de grupo a un archivo**
- **chown** — **cambia el dueño del archivo**
- crontab — ejecuta un comando a una hora programada
- date — muestra la fecha y hora
- df — muestra el espacio libre en disco
- du — muestra el uso del disco
- env — muestra las variables de ambiente
- finger — despliega información de los usuarios
- **kill** — **mata procesos**
- **ps** — **muestra procesos**
- stty — muestra y asigna de configuración a una terminal
- **who** — **muestra los usuarios firmados en el sistema**

Procesamiento de texto

- **cut** — **selecciona una columna y la despliega**
- **ex** — **ejecuta comando dentro de vi**
- **join** — **convierte 2 archivos en una base de datos**
- nawk — versión mejorada de awk
- **paste** — **pega 2 columnas**
- sed — editor de texto no interactivo
- **sort** — **ordena archivos**
- **tr** — **redefine caracteres**
- **uniq** — **encuentra cadenas repetidas**
- vi — editor visual de texto
- xargs — procesa argumentos

2.2 Ambiente cliente servidor

Plataforma cliente/servidor

Es una red de computadoras en la que una o más *ofrecen servicios*, de ahí el nombre de servidores. Las restantes computadoras de la red son clientes, es decir *reciben servicio de estos servidores.*

Los servicios de Bases de Datos suelen ser de los servicios más frecuentes.

Los sistemas cliente se optimizan para la entrada y presentación de los datos, incluyen normalmente *interfaces gráficas de usuario* (GUIs): Pantallas que brindan medios gráficos para que un usuario final acceda a un sistema.

A continuación se explicarán las características principales de dos sistemas servidores de Bases de Datos Oracle y SQLServer.

El proceso de definir las tablas de la base de datos consiste en los pasos siguientes:

1. **Crear los tipos de datos definidos por el usuario**

Tipo de dato definido por el usuario:

Subtipo de uno de los tipos de datos suministrados por el sistema, el cual es adaptado a las necesidades de la base de datos.

2. **Definir las tablas**

2. a. Para cada columna definir su nombre, tipo de dato, opcionalidad y valores por defecto

En SQL Server:

Con la pantalla Manager Tables del SQL Server, las columnas de la nueva tabla se definen en el área cuadriculada, proporcionando (1) Nombre, (3) Tipo de dato, (4) Longitud, y las restricciones que se aplican a los valores de la columna: Not Null, (5) un valor por defecto (valor que entra automáticamente por el sistema cuando el usuario lo omita) y (6) Regla.

Usando el cuadro de diálogo Manage Rule. En la caja Rule Contents se escribe la regla:

Por ejemplo; si se requiere un id numérico entre 1 y 99999, obligatorio:

```
CREATE RULE id_valido AS @id > 0 and @id < 100000
```

2.b. Definir las claves primarias y foráneas

En Oracle:

Las claves primarias y foráneas se definen como restricciones:

Ejemplo:

*Para la tabla INFANTES curp_infante *debe ser único*. Esta restricción se denomina restricción de columna, es decir se establece en la definición de una columna de una tabla.

```
CREATE TABLE INFANTES

(curp_infante CHAR(18) CONSTRAINT PK_INFANTES PRIMARY KEY,……)
```

Para la *clave primaria compuesta* por ejemplo de la tabla PADRES_INFANTE formada por 2 columnas (curp_infante, curp_padre) se debe definir como una restricción de tabla (restricción que se aplica simultáneamente a múltiples columnas de una tabla)

CREATE TABLE PADRES_INFANTE (curp_infante, curp_padre,
...
CONSTRAINT PK_PADRES_INFANTE PRIMARY KEY (curp_infante, curp_padre);

Definiendo Claves Foráneas con Oracle:

- CREATE TABLE PADRES_INFANTE
  (curp_infante CHAR(18), **REFERENCES** INFANTES ON DELETE RESTRICT,
  curp_padre CHAR(18),**REFERENCES** PADRES_TUTORESON DELETE CASCADE,…,
  CONSTRAINT PK_PADRES_INFANTE PRIMARY KEY (curp_infante, curp_padre)

Observe que no se ha usado la palabra constraint, es opcional, la palabra reservada

REFERENCES indica una definición de clave foránea.

Nota:

Si se desea eliminar algún infante, no podrá realizarse si existen asignaciones, primero se deberán eliminar padres_infante, caso contrario en padres_tutores, que si eliminará padres_infante en cascada al eliminarse algún registro en la tabla padres_tutores.

Para la *clave foránea recursiva* por ejemplo curp_infante_hermano

CREATE TABLE infantes
(curp_infante CHAR(18) CONSTRAINT PK_INFANTES PRIMARY KEY,…
curp_infante_hermano CHAR(18) CONSTRAINT FK_INFANTE_HERMANO REFERENCES INFANTES)

Otro ejemplo de *clave foránea multicolumna* mediante una restricción de tabla, por ejemplo la tabla municipios_delegaciones

CREATE TABLE MUNICIPIOS_DELEGACIONES
(id_pais NUMBER(3) NOT NULL, id_estado NUMBER(2) NOT NULL, …..
CONSTRAINT FK_ESTADO_MUN_DEL FOREIGN KEY (id_pais,id_estado) REFERENCES ESTADOS)

Después que se definen todas las claves primarias y foráneas, el DBMS *Oracle* automáticamente *fuerza* que lo sean, por ejemplo cuando se insertan datos se asegura que no sean nulas las claves primarias y no sean repetidas y que las claves foráneas tengan valores que correspondan con claves primarias válidas.

2.c. Definir las restricciones check

Restricción CHECK en SQLServer

- Funciona como regla en SQLServer

CREATE RULE dia_valido AS @dia in ('L','M','MI','J','V')

Restricción CHECK en Oracle

- Definiendo una restricción CHECK como parte de la definición de la columna oficio

CHECK (dia IN ('L', 'M', 'MI', 'J', 'V')),

Ó

CHECK (status_bl IN ('D'))

NOTA:

Las restricciones CHECK en Oracle dan más poder que las reglas del SQLServer, puesto que pueden usarse en múltiples columnas. Sin embargo, éstas no son tan poderosas como los disparadores, porque están limitadas a comprobar valores en una sola tupla a la vez.

UNIQUE garantiza que los valores en la columna deben ser Únicos, sin embargo esto no significa que la columna sea una clave, ya que sí es posible que la *columna tenga valor nulo*. Las dos restricciones NOT NULL y UNIQUE juntas son equivalentes a la restricción PRIMARY KEY.

La restricción UNIQUE aplicada a múltiples columnas en una restricción de tabla tiene el mismo significado que si se aplica a cada columna en una restricción de columna.

Programación del Servidor de Datos:

Los lenguajes de flujo de control del SQLServer y el PL/SQL de Oracle incluyen instrucciones condicionales (IF) e instrucciones para hacer ciclos. También se pueden usar variables para almacenar y calcular valores. Estos lenguajes permiten crear procedimientos almacenados (programa compilado en lenguaje máquina, que se guarda para una ejecución repetida y más eficiente y permiten el paso de parámetros de entrada y salida.

Lenguaje de flujo-de-control:

Tiene las siguientes instrucciones:

1. BEGIN..END que definen bloques para ser tratados como unidades de ejecución.
2. IF…ELSE para la ejecución condicional.
3. WHILE para ejecución repetida o iterativa.
4. BREAK y CONTINUE para la salida anticipada de ciclos WHILE
5. DECLARE que permiten la definición de variables locales.
6. RETURN para terminar un procedimiento y regresar el control al programa que lo llamo
7. PRINT para enviar mensajes a los usuarios
8. COMENTARIOS para documentación interna en los programas.

1. Bloque de Instrucciones BEGIN..END:

Las instrucciones condicional (IF) e iterativa (WHILE) controlan la ejecución de una instrucción simple o de un bloque de instrucciones. Un bloque de instrucciones se indica de la forma siguiente:

```
BEGIN
    instrucción SQL
    …
    instrucción SQL
END
```

2.-Ejecución Condicional IF…ELSE

Sintaxis

IF <expresión condicional> <bloque de instrucciones>

[ELSE <bloque de instrucciones>]

La expresión condicional, la cual debe poderse evaluar a verdadero o falso, contendrá a menudo consultas.

Ejemplo:

```
IF (select avg(sueldo) from TIPO_EMPLEADO where tipo_empleado='1') <8000
    update TIPO_EMPLEADO  set sueldo = suelo + 0.50
```

Condición con 2 consultas

```
IF (select avg(sueldo) from TIPO_EMPLEADO where tipo_empleado='1')<
IF (select avg(sueldo) from TIPO_EMPLEADO where tipo_empleado='2')
      update TIPO_EMPLEADO
      set sueldo = sueldo + 0.50
      where tipo_empleado = '1'
```

Subconsultas en if con exists y uso de 7. PRINT y 8. Uso de comentarios

```
IF EXISTS (select * from tipo_empleado where tipo_emp  in
                  (select tipo_emp from tipo_empleado
                   where nombre_tipo_emp = 'educador'))
  BEGIN
     update tipo_empleado -- Aumento de Salario
     set sueldo = sueldo + 0.50
     print ' ¡¡¡ Esto hay que celebrarlo !!! '
  END
 ELSE
  BEGIN
     update tipo_empleado –Disminución de Salario
    set tarifa_hr = tarifa_hr - 0.50
    print ' ¡ Lo siento, pero así están las cosas ! '
  END
```

Nota: Si un apóstrofe es parte de un mensaje, entonces debe indicarse con dos apóstrofes seguidos.

3. WHILE Ejecución iterativa.

Sintaxis

```
WHILE <expresión condicional>

   <bloque de instrucciones>
```

Ejemplo

```
 WHILE (select sueldo from tipo_empleado
           where nombre_tipo_emp = 'seguridad') < 3000
  BEGIN
    Print '¡¡ Le estamos doblando el pago!! '
    update tipo_emp
    set sueldo = 2 * sueldo
  END
```

4.- Usando BREAK y CONTINUE

BREAK. Palabra clave que causa la salida de la ejecución de un ciclo WHILE

CONTINUE. Palabra clave que causa que el control de ejecución de un lazo WHILE regrese a la primera instrucción del lazo.

```
 (I)    WHILE (select max(sueldo) from tipo_empleado) < 8000
              BEGIN
 (II)       Update tipo_empleado
                 set sueldo = 1.1 * sueldo
 (III)      IF (select sueldo from tipo_empleado) < 4500
                   CONTINUE     (regresa a I)
            (IV)    IF (select min(sueldo) from tipo_empleado) > 4000
                      BREAK          (pasa a VI)
            (V)     update tipo_empleado
                    set sueldo = sueldo + 100
                    where oficio = 'ayudante'
                 END
            (VI) ….
```

5. DECLARE y Variables Locales

Variable local: es una variable definida para usar dentro de un procedimiento y almacenar valores temporales de trabajo.

Una variable local se declara haciendo que el nombre comience con @:

```
Declare @indice int
Select @indice = 3
WHILE @indice > 0
BEGIN
   delete tipo_emp where sueldo > 9000
   update tipo_empleado set sueldo = sueldo * 1.1
   select @indice = @indice – 1
END
```

PROCEDIMIENTOS ALMACENADOS (sqlserver2005)

Son programas SQL que fueron compilados la primera vez que se ejecutaron y luego se almacenaron para su uso posterior.

Se ejecutan muy rápidamente y pueden recibir y regresar parámetros.

Los parámetros de salida se identifican poniendo la palabra “output” o “out”

La palabra clave “as” señala el final de la definición de los parámetros y el comienzo de la definición del procedimiento.

Para ejecutar un procedimiento.

Primero se define la variable local que recibirá el valor de salida del procedimiento. Luego se ejecuta el procedimiento:

Ejemplo:

```
Create procedure calc_sueldo_prom
     @sueldo_prom money output,
     @tipo_oficio varchar(60)
As
    Select @sueldo_prom = avg(sueldo)
    From tipo_empleados
    Where nombre_tipo_emp = @tipo_oficio
```

Ejecución

```
Declare @sueldo_prom money
Exec calc_sueldo_prom
     @sueldo_prom output, educador;
```

Nota: Observar el orden de los parámetros y observe la abreviatura exec

Valores por defecto (default) del parámetro

Se crea, si el programa que llama omite pasar un valor a un parámetro de entrada, entonces el programa usará el valor por defecto para dicho parámetro.

En este ejemplo utiliza el nulo.

```
Create procedure calc_sueldo_prom
        @sueldo_prom money output,
        @tipo_oficio varchar(60) = null
As
  if @tipo_oficio = null
    select @ sueldo _prom = avg (sueldo)
    from tipo_empleados
  else
    select @ sueldo _prom = avg (sueldo)
    from tipo_empleados
    where oficio = @tipo_oficio ;
```

Nota: el valor por defecto se define inmediatamente después de la definición del tipo de dato del parámetro

6. Uso del RETURN

Causa que el procedimiento almacenado termine inmediatamente y regrese el control al programa que lo llamo.

Ejemplo: Para solicitar tarifa salarial máxima, mínima o media de los trabajadores

```
Create procedure funcs_calc_salario
@tipo_fun char(3), @val_ret money output
As
If tipo_func = “max”
Begin
  select @val_ret = max(sueldo) from tipo_empleados
  return
End
If tipo_func = “min”
Begin
  select @val_ret = min(sueldo) from tipo_empleados
  return
End
If tipo_func = “avg”
Begin
  select @val_ret = avg(sueldo) from tipo_empleados
  return
end ;
Ejecución:  Declare @val_ret money
              Exec funcs_calc_salario max, @val_ret output
```

3. Crear los disparadores (triggers)

Introducción a los disparadores (triggers):

Programa que se ejecuta automáticamente cuando se intenta hacer una determinada transacción sobre una tabla específica.

Se pueden definir 3 tipos de triggers: de inserción, de actualización y de borrado.

Supóngase que se quiere mantener una columna en la base de datos que se deriva del cálculo de otras columnas.

Existen 2 tablas de verificación de disparadores que almacenan las tuplas (renglones) insertadas y borradas cada vez que ocurre una actualización (inserted y deleted)

Si se actualizan filas en una tabla, entonces deleted consiste en las viejas versiones de las filas actualizadas e inserted está formada por las nuevas versiones de las mismas filas.

Es importante reservar el uso de los disparadores para aquellas operaciones que siempre se deben realizar a continuación de una actualización específica.

Asumamos que la tabla tipo_empleados incluye una nueva columna pago_acumulado.

La fórmula para calcular el pago acumulado es:

Pago_acumulado = num_total_meses *sueldo

La siguiente solución se aplica no importando si se han añadido, borrado o cambiado una fila o varias filas a la vez, sobre la tabla asignaciones.

SQLServer

Ejemplo:

```
Create trigger actualizacion_pago
On tipo_empleados
For insert, update, delete
As
  Update tipo_empleados
  Set pago_acumulado = pago_acumulado* sueldo*280;
  Update tipo_empleados
  Set pago_acumulado = pago_acumulado *360;
```

Explicación: La primera línea identifica el nombre del disparador, la segunda línea indica que éste se aplica a la tabla asignaciones. La tercera línea establece que el disparador hará fuego en cada operación insert, update o delete.

La próxima línea “as” introduce la parte código. La primera instrucción de actualización suma al pago acumulado usando la tabla inserted, la segunda instrucción resta del pago acumulado usando la tabla deleted.

Es importante reservar el uso de los disparadores para aquellas operaciones que siempre se deben realizar a continuación de una actualización específica.

2.3 Administración de Bases de Datos

La “administración de una base de datos” consiste básicamente en asegurar que la información que ella contiene sea precisa, consistente y que se encuentre disponible en el tiempo y forma en que los usuarios y aplicaciones la necesiten.

El administrador de la base de datos (DBA) es la persona cuyas responsabilidades se centran en la gestión de los aspectos técnicos del sistema de base de datos.

Tareas del DBA

1. Controlar la disponibilidad de la base de datos.

2. Planear y crear Bases de Datos. Revisando tamaños y ubicación en el servidor.

3. Administrar las estructuras físicas (ubicación, creación, modificación).

4. Administrar el almacenamiento basado en diseño, supervisando las tablas o índices de mayor crecimiento (entidades más volátiles en el diagrama Entidad Relación).

5. Controlar la seguridad de la base de datos.

6. Administrar la red. Planear y permitir la conectividad, administrar la red a través de utilerías y herramientas de administración, preservar la seguridad de la red y resolver problemas de red.

7. Respaldar y recuperar información.

8. Sintonizar la base de datos, ajustando los parámetros de configuración de la base de datos.

9. Instalación, selección, operación, mantenimiento y actualización del RDBMS.

10. Examinar el diccionario, así como auxiliarse de herramientas CASE.

Usuarios DBA’s

En ORACLE, existen 2 usuarios que son creados automáticamente: SYS y SYSTEM.

Estos usuarios tienen privilegios extras para ejecutar tareas administrativas.

SYS **(Dueño)**	**SYSTEM** **(Gerente General)**
Propietario del diccionario de la base de datos en el esquema SYS.	Propietario de las tablas y vistas que contienen información administrativa utilizada por las herramientas Oracle.
Password: Change_on_install	Password: Manager
A nivel remoto solo SYS puede dar “*Shutdown*” a la base de datos.	

Diccionario de datos

El Diccionario de Datos es el conjunto central de tablas y vistas de la Base de Datos.

Un Diccionario de Datos proporciona entre otras cosas: Nombre de los usuarios, Privilegios, Nombres de los objetos, Defaults y Espacio ocupado por los objetos.

Estructura del Diccionario de Datos

El diccionario de datos esta compuesto de: Tablas Base (escribe únicamente el DBMS), User accesible (vistas accesibles de los objetos de los usuarios) y Views (Vistas accesibles).

Como se usa el Diccionario de Datos

Durante la operación de la Base de Datos, el DBMS lee el Diccionario de datos para ver que objetos hay, quién los acceso y para actualizar estructuras del desarrollador.

Componentes de la arquitectura Oracle

Oracle es un sistema manejador de Bases de Datos de objetos relacional que proporciona un acercamiento integrado, comprensivo y abierto de la administración de la información.

Usuarios en la base de datos Oracle

Un usuario puede conectarse a un servidor: Registrándose directamente en el host, Usando una conexión de dos capas o Utilizando una conexión de tres capas.

Conectándose a una base de datos

EL usuario que necesita interactuar con el servidor Oracle primero necesita establecer una conexión a la base de datos.

Para conectarse a una base de datos:

- El usuario genera un proceso usuario (user process).

- Cuando un usuario se registra en el servidor Oracle con un nombre de usuario, clave y base de datos, se crea un proceso que corre en el servidor Oracle (server process).

Conexión

Una conexión es una ruta de comunicación entre el user process y el servidor Oracle.

Sesiones

Una sesión es una conexión específica de un usuario al servidor Oracle.

Instancia Oracle

Una instancia Oracle consiste de una estructura de memoria, llamada “Área Global del Sistema” (SGA; System Global Area), y procesos de fondo (background process). Una Instancia Oracle se identifica estableciendo ORACLE_SID en el sistema operativo.

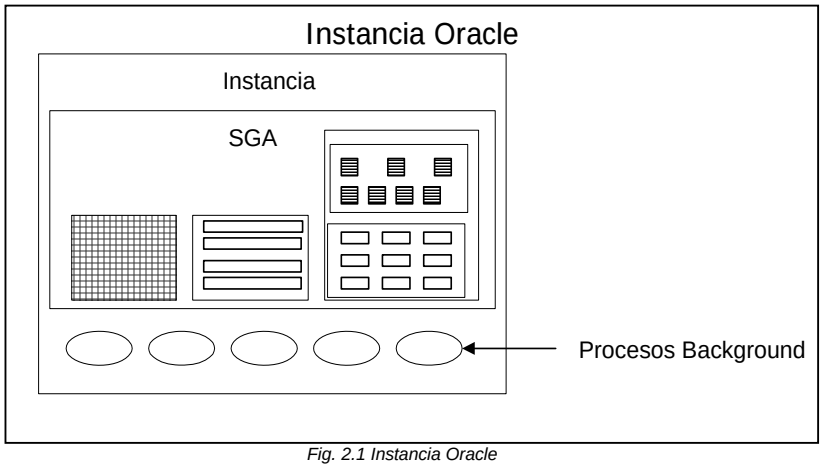


Fig. 2.1 Instancia Oracle

Área Global del Sistema

Es la región de la memoria en donde las estructuras de memoria de una instancia en Oracle están contenidas.
Se localiza en la memoria virtual de la computadora y comprende varias estructuras de memoria, incluyendo:
- Pool compartido: Almacena la información de ejecución más reciente la información del diccionario de datos más recientemente.
- Buffer Caché de la base de datos: Almacena el dato más reciente que se ha utilizado.
- Redo log buffer: Se utiliza para registrar cambios en la Bases de Datos utilizando la instancia.

Procesos de fondo (background process)
Los procesos de fondo en una instancia ejecutan funciones que se necesitan para responder las peticiones de los usuarios concurrentes, sin comprometer la integridad y desempeño del sistema. Cada instancia en Oracle puede utilizar varios procesos de fondo, dependiendo de la configuración.

Bases de Datos Oracle
Una base de datos Oracle (DB_NAME), está compuesta de archivos de sistema operativo.

Archivos de la base de datos
Los archivos de una base de datos contienen datos de usuario e información necesaria para asegurar la adecuada operación de la base de datos.
Una base de datos Oracle consiste de los siguientes tipos de archivos:
- Data files: Almacena el diccionario de datos, objetos de usuario e imágenes anteriores.
- Redo log files: Contiene un registro de cambios de la base de datos.
- Control files: Contiene información necesaria para mantener la integridad de la base de datos.

Otras estructuras físicas clave
Oracle también necesita de otros archivos importantes como:
- Parameter File: Utilizado para definir las características de una instancia Oracle.
- Password File: Utilizado para autenticar privilegios de los usuarios de la Bases de Datos.
- Archived log files: Utilizado para realizar copias fuera de línea de los archivos “redo log”.

Ejecutando consultas
Los siguientes pasos son las principales etapas en el proceso de consultas:

- Análisis (Parse): El proceso de usuario manda una consulta al proceso del servidor con la petición para analizar o compilar la consulta.
El proceso del servidor realiza la validación de la consulta y la compila en el SGA. Finalmente regresa el status (de éxito o fracaso) al proceso de usuario.
- Ejecutar (Execute): El proceso del servidor se prepara para recuperar los datos.
- Obtención (Fetch): El servido de usuario regresa el resultado de la consulta.

Elementos del SGA

El pool compartido (shared pool)
Es una parte del SGA utilizada durante la fase de análisis (parse).
Su tamaño se especifica con el parámetro SHARED_POOL_SIZE en el archivo de parámetros. Sus componentes son:

Librería Caché
Al ejecutarse una sentencia SQL, la librería caché almacena el texto de las sentencias, código analizado (parsed), Árbol de análisis (versión compilada de la sentencia) y plan de ejecución (define los pasos que se deben seguir al correr las sentencias).

El diccionario de datos Caché
Contiene la información mas reciente del diccionario de datos.
Es utilizado durante la fase de análisis para resolver los nombres de los objetos y validar los privilegios de acceso.

Buffer caché de la base de datos
Almacena los bloques de datos mas recientemente utilizados.
El tamaño y número de cada buffer se especifican en los parámetros DB_BLOCK_SIZE y DB_BLOCK_BUFFERS.

El Área Global del Programa (PGA)
Es una región de la memoria que contiene información de datos y controles para un proceso de servidor único. Cuando se utiliza la configuración de servidor dedicado, el PGA contiene: Área de Ordenamiento, Información de Sesión, Estado del curso y Espacio de lista (Stack Space):

Ejecutando una sentencia DML
Una sentencia de manipulación de datos requiere dos fases de procesamiento: análisis y ejecución.

Fases de Ejecución
Considere el ejemplo de ejecución de un comando de actualización:

```
UPDATE tipo_empleados SET sueldo=sueldo*1.1 WHERE nombre_tipo_emp='educador';
```

Los pasos que se siguen son:
1. Si no están listos en el buffer caché, el servidor de procesos lee los bloques de datos y bloques rollback desde los data files.
2. Copia los bloques leídos en el buffer caché.
3. El servidor de procesos bloquea los datos (locks).
4. El servidor registra los cambios a ser hechos en el rollback (before-image) y a los datos con nuevo valor en el buffer redo log.
5. El servidor de procesos registra el before-image al bloque rollback y actualiza el bloque de datos. Estos bloques son marcados como dirty buffers.

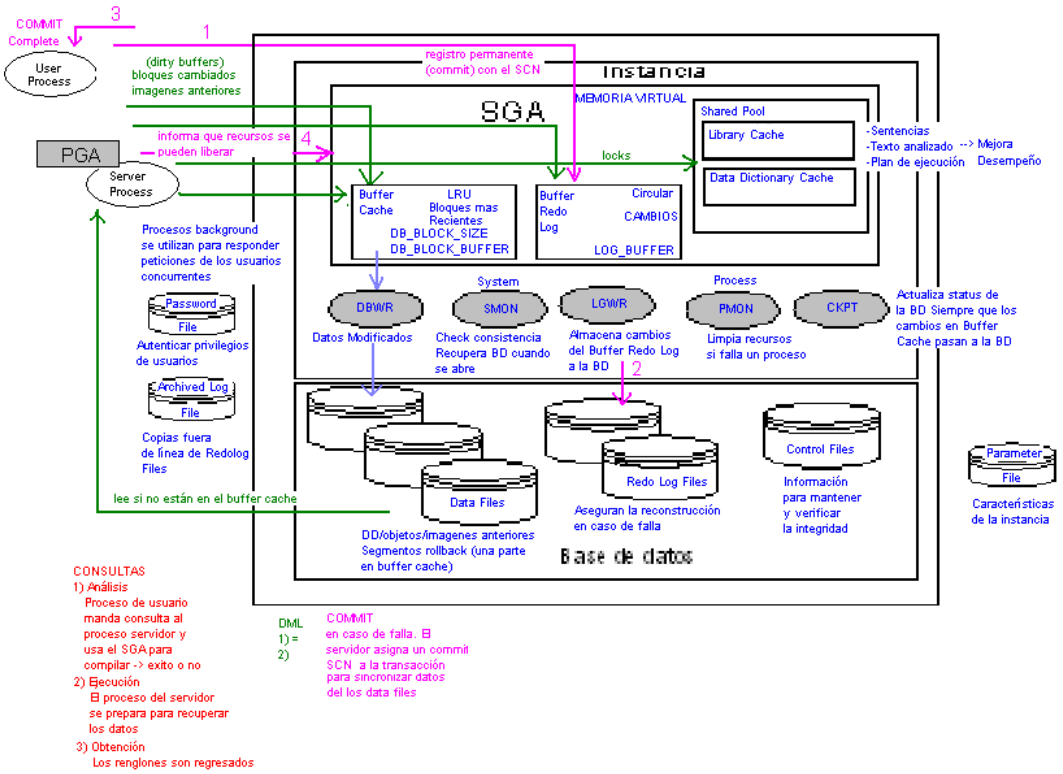


Fig. 2.2 Fases de ejecución Arquitectura Oracle

Segmento Rollback
Segmento de memoria donde se salvan viejos valores antes de cambiarlos por nuevos.
Utilizado para deshacer cambios si la transacción se regresa, asegurar que otras transacciones no vean cambios "no permanentes" y para recuperar la base de datos de una falla.

Redo log buffer
Parte de la memoria en donde el servidor de procesos registra los cambios realizados.
Almacena registros en registros redo y su tamaño se define en el parámetro LOG_BUFFER.

Procesos de fondo (Background process)

Database Writer (DBWR)
El escritor de base de datos (DBWR) escribe los buffers sucios (dirty) u ocupados del buffer caché a los data files y asegura que el número de buffers libres sean suficientes y que estén disponibles en el buffer caché de la base de datos.

Log Writer (LGWR)
Es un proceso que escribe las entradas del buffer redo log dentro de archivos redo log.

Procesando commit (Commit Processing)
Oracle utiliza rápidos mecanismos de commit que garantizan que los cambios acordados pueden recuperarse en caso de fallas.

El sistema de cambios de números (System change number)
Al realizarse un commit, el servidor de Oracle asigna un "commit SCN" a la transacción, el cual es único y ascendente. Dicho numero ayuda a proveer lecturas consistentes de los data files.

Administrando una instancia de Oracle

- Una base de datos Oracle no está disponible a los usuarios hasta que se haya levantado la instancia y abierto la base de datos.

- Los pasos necesarios para dar inicio a una base de datos son: Levantar una instancia, Montar la base de datos y Abrir la base de datos.

- Cada vez que se inicia una instancia, Oracle usa un archivo de parámetros de inicialización los cuales son asignados en el SGA para empezar los procesos background.

Métodos de Autenticación

Autenticación de Sistema Operativo
Oracle adoptara contraseñas existentes del sistema operativo en que se encuentre instalado.

Autenticación desde un Archivo Contraseña
Oracle proporciona una contraseña que conecta al usuario al esquema de SYS en lugar del usuario proporcionado.

Archivo de parámetros

Archivo de texto que guarda parámetros de inicialización. Se identifica por: init<SID>.ora

Usos de parámetros

Los parámetros se utilizan para: Dimensionar los componentes del SGA, Configurar los valores default de la instancia y de la base de datos, Configurar los límites de la base de datos, Definir los diferentes atributos físicos de la base de datos y Especificar los control files, los archived log y las localidades del archivo de rastreo.

Parámetros que deben especificarse en Instancia (initESTANCIA.ora)

Parámetro	**Descripción**
BACKGROUND_DUMP_DEST	Ubicación de los archivos de rastreo.
COMPATIBLE	Versión del servidor compatible con la instancia.
CONTROL_FILES	Nombres de los archivos de control.
DB_NAME	Identificador de la base de datos.
SHARED_POOL_SIZE	Tamaño en bytes del pool compartido. Su valor por default es 3500000
USER_DUMP_DEST	Localidad dónde se crean los archivos de rastreo de usuarios.

Parámetros modificados comúnmente en Instancia (initESTANCIA.ora)

Parámetro	**Descripción**
IFILE	Nombre de otro archivo de parámetros a ser incrustado dentro del archivo de parámetros.
PROCESSES	Número máximo de procesos de sistema operativo que se pueden conectar simultáneamente a esta instancia
SQL_TRACE	Habilita o desactiva el SQL trace.
TIME_STATISTICS	Habilita o desactiva cronometrando en archivos de rastreo (trace) y en las pantallas del monitor.

Etapas en inicio y cierre

A continuación se explican las diferentes fases en que se dividen las etapas de inicio y cierre de una base de datos.

Arranque (Startup)

Levantando la instancia

Para levantar una instancia se necesita: Leer el archivo de parámetros, Asignar el SGA, Levantar los procesos background y Abrir los archivos ALERT y de rastreo (trace).

Montando la base de datos

Para montar una base de datos se necesita: Asociar una base de datos con una instancia iniciada previamente, localizar y abrir los archivos de control especificados en el archivo de parámetros y Leer los archivos de control.

Abriendo la base de datos

Se dice que una base de datos está abierta, cuando se abren los “datafiles en línea” y los “archivos redo log en línea”.

Falla de una instancia

Ocurre una falla de instancia cuando esta no puede continuar trabajando.

Este tipo de fallas se puede originar por la caída del sistema operativo, corte eléctrico, etc.

Recuperación de instancia

La recuperación de la instancia consiste en los pasos siguientes:

- Rolling forward. Se recuperan datos registrados en el redo log en línea y que no se grabaron.

- Abrir la base de datos y esperar a que todas las transacciones pendientes se recuperen.

- Hacer disponible la base de datos.

- Rolling back. Se deshacen las transacciones no precedidas por una instrucción “commit”.

Shutdown en Etapas

Hay tres pasos para cerrar una instancia de la base de datos que está conectada.

Cierre la base de datos.

El primer paso para bajar una base de datos es cerrar la base de datos. Cuando la base de datos está cerrando, Oracle escribe los cambios del buffer caché y las entradas del buffer redo log a los archivos datafiles y redo log en línea. Después de esta operación, Oracle cierra todos los archivos datafiles y los redo log en línea. Los archivos de control permanecen abiertos mientras se cierra la base de datos cuando aún está montada.

Desmontando la Base de datos

El desmontar la base de datos consiste en cerrar los archivos de control de la base.

Shutdown a la instancia

Se cierran los archivos de rastreo (trace) y ALERT, se desasigna el SGA y se terminan los procesos background.

Comando STARTUP

Para levantar una instancia, use el siguiente comando:

```
STARTUP [FORCE] [RESTRICT] [PFILE =FILENAME]
[EXCLUSIVE|PARALLEL|SHARED][OPEN [RECOVER] [database]
| MOUNT | NOMOUNT]
```

Ejemplo:

```
startup pfile= C:\oracle\product\10.2.0\db_1\dbs\initESTANCIA.ora
nomount
Instancia ORACLE iniciada.

Total System Global Area  167772160 bytes
Fixed Size                  1247876 bytes
Variable Size              62915964 bytes
Database Buffers           96468992 bytes
Redo Buffers                7139328 bytes
```

Donde:

FORCE	Interrumpe la instancia que está corriendo antes de realizar un arranque normal.
RESTRICT	Habilita sólo a los usuarios con privilegios de RESTRICTED SESSION para acceder la base de datos.
PFILE=parfile	Permite usar un archivo de parámetros no predefinido para configurar la instancia.
EXCLUSIVE	Solo permite a la instancia actual acceder a la base de datos.
PARALLEL	Permite que múltiples instancias accedan a la base de datos.
SHARED	Pfrece un término alternativo para PARALELO.
OPEN	Permite a los usuarios acceder a la base de datos.
RECOVER	Indica que se realizara una recuperación de medios al iniciar la base de datos.
MOUNT	Monta la base de datos únicamente para el DBA.
NOMOUNT	Crea el SGA e inicia los procesos background pero no permite el acceso a la base de datos

Cambiando la disponibilidad de la base de datos

Para cambiar entre los estados STARTUP NOMOUNT, MOUNT y OPEN, use el comando:

ALTER DATABASE {MOUNT|OPEN}
alter database mount; alter database open;

Comando SHUTDOWN

Opciones del Shutdown

Modo del Shutdown	**Abort**	**Inmediate**	**Transactional**	**Normal**
Permite conexiones nuevas	X	X	X	X
Espera hasta que terminan las sesiones actuales	X	X	X	√
Espera hasta que terminan las transacciones actuales	X	X	√	√
Forza un checkpoint y cierra archivos	X	√	√	√

Para cerrar una instancia, use el siguiente comando:

SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT]
SQL> shutdown immediate Database closed. Database dismounted. ORACLE instance shut down.

Shutdown Normal

No permitirá conexiones nuevas, el servidor espera a que todos los usuarios se desconecten antes de completar el shutdown y se cierra y desmonta la base de datos antes de que se cierre la instancia.

Shutdown Transaccional

Ningún cliente podrá iniciar una transacción nueva. El cliente es desconectado cuando termina la transacción que está en progreso y una vez que todas las transacciones han finalizado, ocurre un shutdown inmediato.

Shutdown Inmediato

No se completarán las transacciones en progreso. El servidor no espera que los usuarios conectados a la base de datos se desconecten y vacía las transacciones activas. Se desconecta a todos los usuarios y se cierra y desmonta la base de datos antes de cerrar la instancia.

Shutdown de Aborto

Todas las sentencias que estaban siendo procesadas son terminadas inmediatamente. Oracle no espera a que usuarios conectados a la base de datos se desconecten, las transacciones sin commit no se deshacen, la instancia es terminada sin cerrar los archivos. El siguiente arranque requerirá una recuperación de instancia.

Obtener y configurar valores de parámetros de instancia

Vistas de desempeño dinámico

Su contenido está relacionado con el desempeño aunque también proveen datos de las estructuras internas y de memoria del disco.

Las vistas son identificadas por el prefijo V_$, o con su sinónimo públicos V$.

La vista V$FIXED_TABLE despliega todas las vistas de desempeño dinámico. Ejecución Fig. 2.3

Vista de desempeño dinámico (accesible en la etapa NOMOUNT)	**Descripción**
V$PARAMETER	Contiene información de los parámetros de inicialización.
V$SGA	Contiene información resumida en el SGA .
V$OPTION	Opciones que son instaladas con el servidor de Oracle.
V$PROCESS	Contiene información acerca de los procesos activos.
V$SESSION	Lista la información de sesión actual.
V$VERSION	Listas el número de versión y los componentes.
V$INSTANCE	Muestra el estado de la instancia actual.
V$THREAD	Contiene información encolada (thread).
V$CONTROLFILE	Lista los nombres de los archivos de control.
V$DATABASE	Contiene información de la base de datos.
V$DATAFILE	Contiene información del archivo de datos desde el archivo de control.
V$DATAFILE_HEADER	Muestra información del encabezado del archivo de datos desde el archivo de control.
V$LOGFILE	Contiene información de archivos redo log en línea

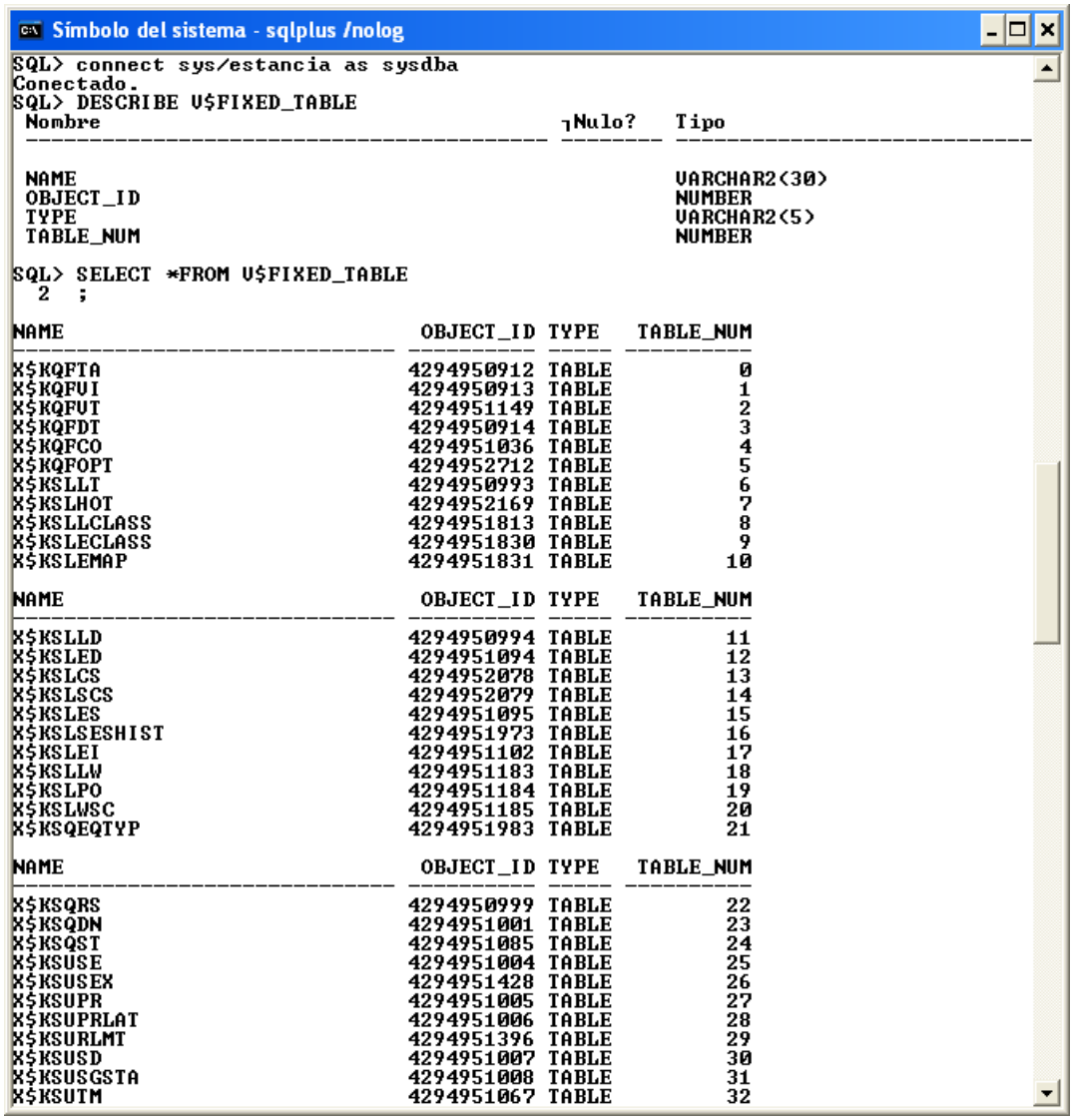


Fig. 2.3 Vista V$FIXED_TABLE

Desplegar valores actuales de los parámetros

Para ver los parámetros establecidos para una base de datos use el comando del Server Manager SHOW PARAMETER:

```
SHOW PARAMETER control
```

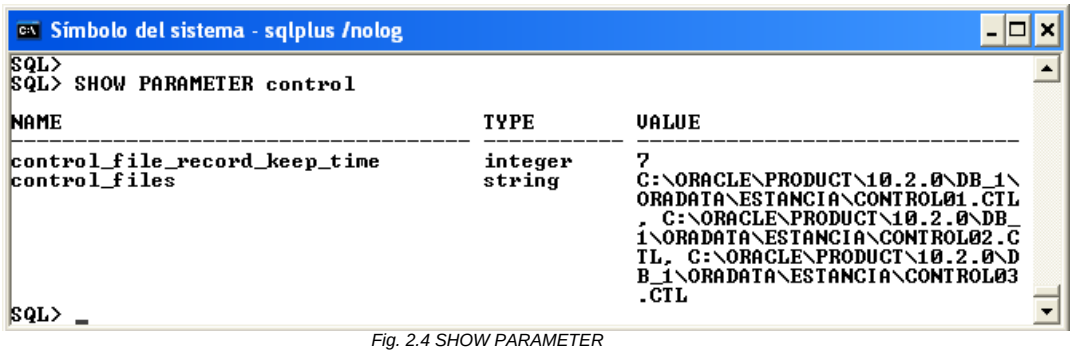


Fig. 2.4 SHOW PARAMETER

También puede usar la vista de desempeño dinámico V$PARAMETER para determinar la configuración actual de cualquier parámetro:

```
SELECT name FROM v$parameter
WHERE name LIKE '%control%';
```

Parámetros de inicialización dinámica

Algunos parámetros pueden modificarse mientras se está ejecutando una instancia:

- ALTER SESSION: Modifica el valor del parámetro para la sesión que ejecuta el comando.

- ALTER SYSTEM: Modifica el valor del parámetro globalmente.

- ALTER SYSTEM DEFERRED: Modifica el valor para las sesiones futuras que se conecten.

```
ALTER SESSION SET SQL_TRACE=true;
ALTER SYSTEM SET TIMED_STATISTICS=true;
ALTER SYSTEM SET SORT_AREA_SIZE=131072 DEFERRED;
ALTER SESSION SET parameter_name = valor
ALTER SYSTEM SET parameter_name = valor  [DEFERRED]
```

Use la vista V$PARAMETER o V$SYSTEM_PARAMETER para listar información modificada:

```
SELECT isses_modifiable, issys_modifiable,
ismodified, name FROM v$system_parameter
WHERE ismodified != 'FALSE';
```

* V$PARAMETER muestra los valores de sesión actual y el comando V$SYSTEM_PARAMETER los valores del sistema actual independientes de la sesión.

Administrando sesiones

Sesión restringida

La base de datos puede levantarse solo para usuarios con privilegios RESTRICTED SESSION.

Habilitando e inhabilitando sesiones restringidas

Use el comando STARTUP para restringir el acceso a la base de datos:

```
STARTUP RESTRICT
```

Use el comando ALTER SYSTEM para colocar una instancia en modo restringido:

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
```

La base de datos también puede ponerse en modo restringido usando SQL ALTER SYSTEM:

```
ALTER SYSTEM {ENABLE| DISABLE} RESTRICTED SESSION
```

Donde:

ENABLE RESTRICTED SESSION	Permite accesos futuros solo para usuarios con privilegios RESTRICTED SESSION.
DISABLE RESTRICTED SESSION	Permite que todos los usuarios ingresen a la base de datos.

El comando ALTER SYSTEM no desconecta las sesiones actuales.

La vista de desempeño dinámica V$INSTANCE contiene información del modo restringido.

```
SELECT logins FROM v$instance;
```

Finalizando sesiones

Para “matar” todas las sesiones de usuario actuales realice los siguientes pasos:

1. Identificar la sesión para terminar con la vista de desempeño dinámico V$SESSION:

```
SELECT sid, serial# FROM v$session
        WHERE username='ESTANCIA';
```

2. Ejecutar el comando ALTER SYSTEM:

```
ALTER SYSTEM KILL SESSION '147,25';
```

Efectos de terminar una sesión

ALTER SYSTEM KILL SESSION genera el proceso background PMON, regresa la transacción del usuario actual, libera todas las tablas guardadas o todos los renglones bloqueados y libera todos los recursos reservados del usuario.

Matar los procesos del sistema operativo

Cuando una sesión termina, el servidor no mata los procesos de sistema operativo.

El siguiente comando desconecta y termina el proceso del servidor una sesión que su transacción actual es finalizada

```
ALTER SYSTEM DISCONNECT SESSION '147,25 ' POST_TRANSACTION
```

Donde *integer1* = V$SESSION.SID, *integer2* = V$SESSION.SERIAL#.

Para estancia

Archivo ALERT y archivos de rastreo (trace)

Archivos ALERT

El archivo ALERT consiste de una bitácora cronológica de mensajes y errores.

Archivos de rastreo

Son archivos que contienen información acerca de errores.

El rastreo puede habilitarse o inhabilitarse por el parámetro de inicialización SQL_TRACE.

La siguiente sentencia permite escribir a un archivo de rastreo una sesión particular:

```
ALTER SESSION SET sql_trace=true;
```

Controlando los archivos de rastreo

Los siguientes parámetros controlan la localización y tamaño de los archivos de rastreo

BACKGROUND_DUMP_DEST	Localización de los archivos ALERT y de rastreo.
USER_DUMP_DEST	Localización de los archivos de rastreo a petición de los usuarios.
MAX_DUMP_FILE_SIZE	Máximo tamaño de los archivos de rastreo de usuarios.

Creando una base de datos

Con la creación de una base de datos se preparan por única vez archivos de sistema operativo.

El comando CREATE DATABASE inicia la creación de los archivos de control, redo log y la estructura del diccionario de datos que Oracle requiere para acceder a la base de datos.

Preparando el Sistema Operativo

Prerrequisitos de creación

Para comenzar con la creación de una base de datos es necesario contar con una cuenta privilegiada autenticada, memoria para iniciar la instancia y espacio en disco suficiente

Planeando la localización de los archivos de la base de datos

- Mantenga al menos dos copias activas de los archivos de control de la base de datos activos en al menos dos diferentes dispositivos (control01.ctl).

- Los archivos redo log de la base de datos consisten en múltiples grupos de archivos redo log en línea. Un grupo de archivos log en línea consiste en copias idénticas, que deben localizarse en diferentes discos.

- Separe los data files de acuerdo a sus datos.

Creación de una base de datos manualmente sin asistente

1. Decida una instancia única, un nombre para la base de datos, y el conjunto de caracteres de la base de datos.

2. Configure las variables de sistema operativo.

3. Prepare el archivo de parámetros.

4. Cree un archivo contraseña.

5. Inicie la instancia.

6. Cree la base de datos.

7. Corra los scripts para generar el diccionario de datos

Ambiente de sistema operativo en UNIX

Configure las siguientes variables de ambiente:

ORACLE_SID	Nombre de B.D., al cambiarlo se apunta hacia otra B.D.

Decida un nombre único para la instancia y configure las siguientes variables de ambiente:

Variable	**Descripción**
ORACLE_SID	Especifica el nombre único de instancia.

Preparando el archivo de parámetros

1. Cree un nuevo Init<SID>.ora

```
$cp INIT.ora $ORACLE_HOME/dbs/ initESTANCIA.ora
```

2. Modifique el initESTANCIA.ora editando los parámetros.

Preparando el archivo de parámetros

Cambie la configuración de algunos parámetros, otros pueden quedar como default.

Se debe especificar al menos los siguientes parámetros antes de iniciar la instancia:

Parámetro	**Descripción**
DB_NAME	Identificador de la base de datos de 8 caracteres o menor
CONTROL_FILES	Especifica una lista de archivos de control.
DB_BLOCK_SIZE	Determina el tamaño del bloque de la base de datos.

Levantando la instancia

1. Conéctese como SYSDBA.

2. Levante la instancia en la etapa NOMOUNT

```
STARTUP NOMOUNT \PFILE= initESTANCIA.ora
```

Creando la base de datos

Para crear una base de datos, use el siguiente comando de SQL:

```
CREATE DATABASE [database] [CONTROLFILE REUSE]
[LOGFILE [GROUP integer] filespec
[,          [GROUP integer] filespec]...]
[MAXLOGFILES integer] [MAXLOGMEMBERS integer]
[MAXLOGHISTORY integer] [MAXDATAFILES integer]
[MAXINSTANCES integer] [ARCHIVELOG | NOARCHIVELOG]
[CHARACTER SET charset] [NATIONAL CHARACTER SET charset]
[DATAFILE filespec [autoextend_clause] [,      filespec [autoextend_clause]...]]
fileespec :== 'filename' [SIZE integer] [K | M] [REUSE] autoextend_clause :==
[AUTOEXTEND {OFF ON [NEXT integer [K | M] [MAXSIZE {UNLIMITED | integer [K | M] } ]
} ]
```

Ejemplo:

```
create database ESTANCIA
maxlogfiles   32
MAXLOGMEMBERS 5
maxdatafiles 400
maxinstances   8
ARCHIVELOG
CHARACTER SET WE8ISO8859P15
NATIONAL CHARACTER SET AL16UTF16
EXTENT MANAGEMENT LOCAL
SYSAUX DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\sysaux_01.dbf' SIZE 200M REUSE
DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\system_01.dbf' SIZE 500M
logfile  GROUP 1 ('C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\redo01a.log') SIZE 5M,
        GROUP 2 ('C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\redo02a.log') SIZE 5M
DEFAULT TEMPORARY TABLESPACE TEMP
       TEMPFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\temp_01.dbf' SIZE 50M
       REUSE AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
UNDO    TABLESPACE UNDO
       DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\undo_01.dbf' SIZE 50M
       REUSE AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED
;
```

Donde:

database	Es el nombre de la base de datos a crearse.
CONTROLFILE REUSE	Específica que un archivo de control existente.
LOGFILE GROUP	Nombre y grupo de los archivos redo log a usarse.
MAXLOGFILES	Máximo de grupos de archivos log que pueden ser creados.
MAXLOGMEMBERS	Máximo de archivos de miembros para el grupo de archivos log.
MAXLOGHISTORY	Máximo de archivos redo log almacenados.
DATAFILE *filespec*	Especifica los archivos de datos a usarse.
AUTOEXTEND	Habilita o inhabilita la extensión automática de un archivo de datos.
MAXDATAFILES	Máximo de datafiles que pueden ser creados para la base de datos.
MAXINSTANCES	Máximo de instancias que pueden simultáneamente montar y abrir la base de datos.
ARCHIVELOG	Establece que los redo log deben almacenarse antes de que puedan reutilizarse.
NOARCHIVELOG	Establece que los redo log puedan reutilizarse sin antes almacenar su contenido
CHARACTER SET	Grupo de caracteres usados por la base de datos en el almacenamiento de datos.
NATIONAL CHARACTER SET	Grupo de caracteres nacional usado para almacenar datos en columnas definidas como: NCHAR, NCLOB o NVARCHAR2.

Después de la creación de la base de datos

Una vez que se ha creado la base de datos esta contiene: Datafiles, Archivos de control y archivos redo log, Usuario SYS/change_on_install, Usuario SYSTEM/manager, Segmentos rollback SYSTEM y Tablas internas.

Administrando tablespaces y data files

La arquitectura de una base de datos se compone de estructuras lógicas y físicas.

Estructura lógica de la base de datos

Tablespaces y Data Files

Una base de Datos de Oracle puede dividirse en pequeñas áreas lógicas de espacio conocidas como tablespaces, las cuales a su ves se dividen en uno o más archivos llamados data files.

TABLESPACES

Características de los Tablespaces

- Pertenecen a una sola base de datos.
- Cada uno consiste de uno o más archivos de sistema operativo.
- Pueden traerse en línea mientras la base de datos está corriendo.
- Excepto por el tablespace SYSTEM o un tablespace con un segmento rollback activo, pueden tomarse fuera de línea, dejando la base de datos corriendo.
- Pueden cambiarse entre estados de lectura-escritura y sólo lectura.

Usos de tablespaces

Los tablespaces nos sirven para:

- Controlar la asignación de espacio y asignación de cuotas de espacio a usuarios.
- Controlar la disponibilidad de datos en línea o fuera de línea.
- Distribuir almacenamiento de datos a través de dispositivos.
- Mejorar las operaciones de respaldo parcial y recuperación parcial.
- Mantener cantidades grandes de datos estáticos en dispositivos de solo lectura.

Tipos de tablespaces
Tablespaces SYSTEM y Non-SYSTEM

Esencialmente, el servidor percibe dos tipos de tablespaces: SYSTEM y NON-SYSTEM.

Tablespaces SYSTEM	Tablespaces Non-SYSTEM
• Requerido en todas las Bases de Datos para la operación de base de datos. • Contiene información del diccionario de datos, definiciones de stored procedures, paquetes y triggers de Bases de Datos. • Contiene segmentos rollback SYSTEM.	• Permite mayor flexibilidad en la administración de la base de datos • Almacena segmentos rollback, segmentos temporales, datos de aplicación e índices de aplicación

Creando tablespaces

Puede crear un tablespace con el comando CREATE TABLESPACE:

```
CREATE TABLESPACE tablespace DATAFILE filespec [autoextend_clause]
 filespec [autoextend_clause]]. . .
[MINIMUM EXTENT integer [K|M]] [DEFAULT storage_clause]
[PERMANENT|TEMPORARY] [ONLINE|OFFLINE]
storage_clause:==
STORAGE ( [INITIAL integer [K|M]] [NEXT integer [K|M]]
[MAXEXTENT {integer|UNLIMITED}] [MINEXTENTS integer]
[PCTINCREASE integer] )
```

Ejemplo:

```
CREATE TABLESPACE TS_CATALOGOS
        DATAFILE 'C:\oracle\product\10.2.0\db_1\oradata\ESTANCIA\ts_catalogos.dbf' SIZE 10M AUTOEXTEND on
        LOGGING
        DEFAULT STORAGE  (
                INITIAL 2330
                NEXT 2330
                MINEXTENTS 1
                MAXEXTENTS 3
                PCTINCREASE 1
        )
        ONLINE
        PERMANENT
;
```

Donde:

tablespace	Es el nombre del tablespace a crearse.
DATAFILE	Especifica el o los data files que construyen el tablespace.
DEFAULT STORAGE	Parámetros de almacenamiento para los objetos dentro del tablespace.
MINIMUM EXTENT	Asegura que cada tamaño de extensión usado en el tablespace es un múltiplo del valor entero.
ONLINE	Hace disponible el tablespace después de la creación.
OFFLINE	Hace no disponible el tablespace después de la creación.
PERMANENT	Especifica que el tablespace puede usarse para objetos permanentes.
TEMPORARY	Especifica que el tablespace será usado para objetos temporales.

* El número de tablespaces no puede exceder el número de data files. El número máximo de data files por tablespace es de 1023.

* Con la configuración de la opción MINEXTENT, el DBA controla la fragmentación en el tablespace. Esta opción sólo puede especificarse por un tablespace, no para el storage de objetos individuales.

Parámetros de Almacenamiento

Los parámetros INITIAL, NEXT, MAXEXTENT, MINEXTENT y PCTINCREASE influyen en la ubicación del segmento.

INITIAL

Define el tamaño de la primera extensión (extent).

El tamaño mínimo de la primera extensión es de 2 bloques, esto es (2*DB_BLOCK_SIZE).

El tamaño por default es de 5 bloques, esto es (5*DB_BLOCK_SIZE).

NEXT

Se refiere al tamaño de la segunda extensión.

El tamaño mínimo de la extensión Next es 1 bloque.

El tamaño por default es de 5 bloques, esto es (5*DB_BLOCK_SIZE).

MINEXTENTS
Es el número de extensiones localizadas cuando se crea el segmento. Su valor mínimo y default es uno.

PCTINCREASE

Es el porcentaje a través del cual crece la extensión.

MAXEXTENTS

Determina el número máximo de extensiones que puede tener un segmento.

El tamaño máximo también puede especificarse a través del comando UNLIMITED y es equivalente a un valor de 2147483645.

Tablespace temporal
Un tablespace temporal solo puede usarse por segmentos de orden (sort segments) y no pueden contener objetos permanentes.

```
CREATE TABLESPACE sort
DATAFILE '/DISK2/sort01.dbf' SIZE 50M
MINIMUM EXTENT 1M
DEFAULT STORAGE (INITIAL 2M NEXT 2M
MAXEXTENTS 500 PCTINCREASE 0)
TEMPORARY;
```

Cambiando el tamaño de tablespaces
Puede aumentar el tablespace añadiendo un data file a un tablespace o cambiando el tamaño de un data file.

Añadiendo Data files a un tablespace
Utilice el comando ALTER TABLESPACE ADD DATAFILE:

```
ALTER TABLESPACE tablespace
ADD DATAFILE filespec [autoextend_clause]
[, filespec [autoextend_clause]]. . .
```

DATAFILES

Cambiando el tamaño de los Data files
Puede cambiar el tamaño de un data file usando automáticamente la opción AUTOEXTEND o usando manualmente el comando ALTER DATABASE.

Redimensionamiento automático de Data files
La opción del comando AUTOEXTEND habilita o deshabilita la extensión automática de datafiles. Cuando se crea un data file, los siguientes comandos SQL pueden usarse para especificar la extensión automática de archivo.

Configurar AUTOEXTEND mientras crea un datafile

Use el siguiente comando para añadir un data file con la extensión automática habilitada:

```
ALTER TABLESPACE tablespace
ADD DATAFILE filespec [autoextend_clause]
[,          filespec [autoextend_clause]]. . .
```

Especificando AUTOEXTEND para un Data file existente
Para habilitar o deshabilitar la extensión automática de archivos para data files existentes use:

```
ALTER DATABASE [database]
DATAFILE 'filename' [, 'filename']. . .
Autoextend_clause
```

Cambiando el tamaño de Datafiles manualmente
Use el comando ALTER DATABASE para aumentar o disminuir el tamaño de un data file:

```
ALTER DATABASE [database]
DATAFILE 'filename' [, 'filename']. . .
RESIZE integer [K|M]
```

Donde:

integer	Es el tamaño absoluto del data file resultante.

Cambiando la configuración de almacenamiento

Use el comando ALTER TABLESPACE para modificar la definición del almacenamiento por default de un tablespace:

```
ALTER TABLESPACE tablespace
{MINIMUM EXTENT integer [K|M]
|DEFAULT storage_clause}
```

Cambiando tablespaces en línea y fuera de línea

Estado OFFLINE
El tablespace SYSTEM y cualquier tablespace con segmentos rollback activos y no pueden tomarse fuera de línea.

```
ALTER TABLESPACE app_data OFFLINE;
```

El estado OFFLINE de un tablespace
El servidor de Oracle realiza un checkpoint sobre todos los data files en un tablespace antes de que sea tomado fuera de línea.

Cambiando un tablespace a offline
Aunque la base de datos esté abierta, se puede tomar cualquier tablespace excepto SYSTEM o cualquier tablespace con segmentos rollback activos o temporales. Cuando un tablespace es tomado offline, el servidor pone todos los data files asociados offline:

```
ALTER TABLESPACE tablespace
{ONLINE|OFFLINE [NORMAL|TEMPORARY|INMEDIATE] }
```

Moviendo data files
Se pueden mover data files con alguno de los siguientes métodos:

Usando el comando ALTER TABLESPACE
El comando ALTER TABLESPACE es aplicado sólo a data files en un tablespace no SYSTEM, que no contiene segmentos rollback activos o temporales:

- Tome el tablespace offline.

- Use un comando de sistema operativo para mover o copiar los archivos.

- Ejecute el comando ALTER TABLESPACE RENAME DATAFILE.

- Traiga el tablespace online.

- Use un comando de sistema operativo para borrar el archivo, si es necesario.

```
ALTER TABLESPACE tablespace
RENAME DATAFILE 'filename' [, 'filename']. . .
TO 'filename' [, 'filename']. . .
```

Usando el comando ALTER DATABASE
El comando ALTER DATABASE puede usarse para mover cualquier tipo de data file:

```
ALTER DATABASE [database]
RENAME FILE 'filename' [, 'filename']. . .
TO 'filename' [, 'filename']. . .
```

Proceso es para renombrar archivos en tablespaces que no pueden tomarse offline:
1. Baje la base de datos.

2. Use un comando de sistema operativo para mover los archivos.

3. Monte la base de datos.

4. Ejecute el comando ALTER DATABASE RENAME FILE.

5. Abra la base de datos.

Tablespaces solo lectura
El tablespace APP_DATA está solo disponible para operaciones de lectura.

Use el comando SQL ALTER TABLESPACE para cambiar un tablespace a solo lectura o solo escritura:

```
ALTER TABLESPACE tablespace READ {ONLY|WRITE}
```

Recomendaciones al hacer tablespaces solo lectura
- El tablespace debe estar en línea.

- El tablespace no debe contener segmentos rollback activos.

- El tablespace actual no debe estar involucrado con un respaldo en línea.

Eliminando tablespaces

Puede borrar un tablespace con el comando SQL DROP TABLESPACE:

```
DROP TABLESPACE tablespace
[INCLUDING CONTENTS [CASCADE CONSTRAINTS]]
```

Donde:

tablespace	Especifica el nombre del tablespace a ser borrado.
INCLUDING CONTENTS	Suprime todos los segmentos en el tablespace.
CASCADE CONSTRAINTS	Elimina las restricciones de integridad referencial de las tablas fuera del tablespace.

Información del diccionario de datos acerca de los tablespaces y datafiles

Obteniendo información del tablespace
Algunos parámetros importantes del tablespace son: TABLESPACE_NAME, NEXT_EXTENT, MAX_EXTENTS, PCT_INCREASE, MIN_EXTLEN, STATUS y CONTENTS.

Utilice la siguiente consulta para obtener información del tipo de tablespace:

```
SELECT tablespace_name, contents, status
FROM dba_tablespaces;
```

Obteniendo Información del Data File
(***DBA_DATA_FILES***)

Algunos parámetros importantes de un data file son: FILE_NAME, TABLESPACE_NAME, BYTES, AUTOEXTENSIBLE, MAXBYTES e INCREMENT_BY.

La siguiente consulta regresa información del tablespace al cual pertenecen los datafiles y la configuración de la opción AUTOEXTEND:

```
SELECT file_name, tablespace_name, bytes_autoextensible,
        maxbytes, increment_by
FROM dba_data_files;
```

Obteniendo información de data files y del tablespace desde el archivo de control
La siguiente consulta lista el nombre y tamaño de los data files, el nombre del tablespace al que pertenecen y la disponibilidad de los data files:

```
SELECT file#, rfile,d.name, status, enabled,
        bytes, create_bytes, t.name
FROM v$datafile d, v$tablespace t
WHERE t.ts#=d.ts#;
```

Estructura de almacenamiento y relaciones

Jerarquía Lógica

- Una base de datos está lógicamente agrupada en tablespaces.

- Un tablespace puede consistir de uno o más segmentos.

- Cuando se crea un segmento, consiste de al menos una extensión, que es un conjunto continuo de bloques.

- Un bloque lógico es la unidad más pequeña usada para operaciones de lectura-escritura.

Tipos de Segmentos
Los segmentos son objetos que ocupan un espacio dentro de una base de datos.

Tabla
Los datos dentro de una tabla son almacenados sin un orden en particular.

Toda la información en la tabla no particionada debe almacenarse en un tablespace.

Tabla particionada (Table partition)
La información dentro de la tabla puede almacenarse en varias particiones, cada una de las cuales reside en diferentes tablespaces. El servidor soporta el particionamiento a partir de un rango de valores llave.

Clúster
Los renglones en un clúster son almacenados basados en los valores de la columna llave. Un clúster puede contener una o más tablas y es un tipo de segmento de información. Las tablas en un clúster pertenecen al mismo segmento y comparten las mismas características de almacenamiento.

Índice
El propósito de éste segmento es buscar la localización de renglones en una tabla basada en una llave específica. Las entradas para un índice son almacenadas dentro de un segmento

Índice particionado
Un índice puede particionarse y extenderse a través de algunos tablespaces. Cada partición en el índice corresponde a un segmento.

Segmento Rollback
Es usado por una transacción que está haciendo cambios a una base de datos. Antes de cambiar la información, el valor anterior es almacenado en el segmento rollback.

Segmento Temporal
Cuando un usuario ejecuta comandos como CREATE INDEX, SELECT DISTINCT y SELECT GROUP BY, Oracle intenta llevar a cabo el ordenamiento en memoria tanto como sea posible. Cuando un ordenamiento necesita demasiado espacio, los resultados intermedios son escritos en el disco. En estos casos se crean segmentos temporales.

Segmento LOB
Cuando se inserta en una tabla un objeto grande como una imagen o video, Oracle lo almacena en segmentos separados conocidos como segmentos LOB. La tabla en la que se realizo la inserción solo contendrá un apuntador hacia la localidad de información correspondiente.

Índice LOB
Un segmento de índice LOB es creado implícitamente cuando el segmento LOB se crea.

Tabla anidada
Una columna en una tabla puede ser una tabla definida por un usuario. La tabla interna, que es conocida como tabla anidada es almacenada como un segmento separado.

Generalmente las tablas anidadas tienen las siguientes características:

- Los registros en una tabla anidada tienen la misma estructura.

- Los registros en una tabla anidada se almacenan por separado de la tabla padre con un apuntador desde el registro correspondiente en la tabla padre.

Segmento Bootstrap
Un segmento bootstrap (o segmento caché) es creado por el script sql.bsq cuando se crea una base de datos. El segmento ayuda a inicializar el diccionario de datos cache cuando la base de datos es abierta por una instancia.

El segmento bootstrap no puede consultarse ni actualizarse.

Controlando el uso de extensiones por Segmentos.

Una cláusula de almacenamiento puede especificarse a nivel del segmento para controlar como son asignadas las extensiones a un segmento.

* Un parámetro de almacenamiento especificado a nivel de segmento reemplazara a la opción puesta al nivel de tablespace, excepto para el parámetro de tablespace MINIMUM EXTENT.

Extensiones libres y usadas
Cuando se crea un tablespace, los data files en el tablespace contienen un bloque cabecera y una extensión libre (parte restante del data file).
Cuando los segmentos son creados, son asignados al espacio de las extensiones libres en un tablespace y cuando los segmentos liberan espacio, las extensiones que son liberadas son añadidas al pool de extensiones libres disponible en el tablespace.

Unión de espacio libre (Coalescing)
Cuando varias extensiones son desasignadas dentro de un tablespace puede liberarse espacio contiguo. En el caso de que existan extensiones liberadas contiguas, estas pueden unirse en una sola cuando: SMON inicia una transacción de espacio para fusionar extensiones libres adyacentes, El servidor de Oracle necesita asignar una extensión que necesita más espacio de una extensión libre adyacente o Cuando es demandado por el DBA.

* SMON une extensiones sólo en aquellos tablespaces donde PCTINCREASE es mayor que cero. Asignar PCTINCREASE=1 en la cláusula de almacenamiento para el tablespace que contiene los objetos de usuario para permitir la unión automática del espacio libre.

Uniendo por demanda
La vista DBA_FREE_SPACE_COALESCED puede usarse para identificar las extensiones que pueden unirse dentro de un tablespace.

```
SELECT tablespace_name, total_extents,
            percent_extents_coalesced
FROM    dba_free_space_coalesced
WHERE percent_extents_coalesced <> 100;
```

Contenido de los bloques de Bases de Datos
Los bloques de datos de Oracle contienen:

- Cabecera del bloque: Contiene la dirección del bloque de datos, directorio de la tabla, directorio del renglón y ranuras de transacción.
- Espacio de datos: Un renglón de datos es insertado dentro del bloque hacia arriba.
- Espacio libre: El espacio libre en un bloque es inicialmente contiguo.

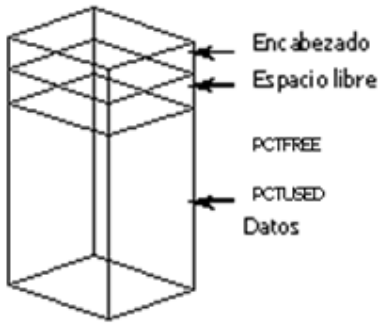


Fig. 2.5 Bloques de base de datos

Parámetros de utilización de espacio de bloque
Pueden usarse para controlar el uso de espacio en los segmentos de datos e índices.

Parámetros para control de concurrencia

INITRANS y MAXTRANS

Especifican el número inicial y máximo de ranuras de transacción, las cuáles son creadas en un bloque de datos o índice. Las ranuras de transacción son usadas para almacenar información de las transacciones que están haciendo cambios al bloque en un punto de tiempo.

INITRANS

Mínimo nivel de concurrencia. Valor default es 1 para segmentos de datos y 2 para índices.

MAXTRANS

Asigna el límite para el número de transacciones concurrentes que pueden hacer cambios a la información o al índice. Cuando es puesto, éste valor restringe el uso de espacio para las ranuras de transacción y por lo tanto garantiza que haya suficiente espacio en el bloque para usar por los datos o el índice. Su valor por default es 255.

Parámetros para el control del uso de espacio de información

PCTFREE
Porcentaje de espacio en cada bloque reservado para crecer resultando de actualizaciones a los renglones del bloque. Su valor por default es 10%:

Un PCTFREE grande permite más actualizaciones dentro del bloque de base de datos. Configure un valor mayor si la tabla contiene columnas que están inicialmente nulas y después se actualizan con un valor o cuando contiene columnas que parecen incrementarse en tamaño como resultado de una actualización.

PCTUSED

Representa el porcentaje de espacio mínimo usado que el servidor intenta mantener por cada bloque de datos de la tabla. Cuando el espacio de un bloque cae abajo de PCTUSED, el bloque es candidato para recibir inserciones futuras en una lista libre. Su valor por default 40%

Configure PCTUSED para asegurar que el bloque sea regresado a la lista libre cuando haya espacio suficiente para acomodar un registro promedio.

Obteniendo información de las estructuras de almacenamiento

Las relaciones entre tablespaces, data files, segmentos y extensiones pueden verse consultando el diccionario de datos. Cuando se crea un segmento, un renglón es visible en DBA_SEGMENTS. El espacio asignado para las extensiones en éste segmento puede verse en DBA_EXTENTS, mientras que DBA_FREE_SPACE es ajustado para mostrar el espacio libre en los archivos donde las extensiones han

sido creadas para el segmento. Todo el espacio en un archivo (excluyendo la cabecera del bloque) debe ser contado para DBA_FREE_SPACE o en DBA_EXTENTS.

Consultando información de segmentos (DBA_SEGMENTS)

Información General	Tamaño	Configuración de almacenamiento
OWNER	EXTENTS	INITIAL_EXTENT
SEGMENT_NAME	BLOCKS	NEXT_EXTENT
SEGMENT_TYPE		MIN_EXTENTS
TABLESPACE_NAME		MAX_EXTENTS
		PCT_INCREASE

La vista DBA_SEGMENTS muestra el número de extensiones actual y bloques asignados en un segmento.

```
SELECT segment_name, tablespace_name, extents, blocks
FROM     dba_segments
WHERE  owner='SCOTT';
```

Obteniendo información de extensiones usadas (DBA_EXTENTS)

Identificación	Ubicación y tamaño
- OWNER	- TABLESPACE_NAME
- SEGMENT_NAME	- RELATIVE_FNO
- EXTENT_ID	- FILE_ID
	- BLOCK_ID
	- BLOCKS

Use de la vista DBA_EXTENTS para ver las extensiones de un segmento dado.

```
SELECT extent_id, file_id, block_id, blocks FROM dba_extents
WHERE owner              ='SCOTT'
AND      segment_name='EMP';
```

Revisando información de extensiones libres(DBA_FREE_SPACE)

Ubicación y tamaño
- TABLESPACE_NAME
- RELATIVE_FNO
- FILE_ID
- BLOCK_ID
- BLOCKS

Use de la vista DBA_FREE_SPACE para ver las extensiones de un segmento dado:

```
SELECT tablespace_name, count(*),
            max (blocks), sum(blocks)
FROM dba_free_space
GROUP BY tablespace_name;
```

Planeación de la localización de segmentos

Tipos de objetos y fragmentación

Tablespace	Uso	Fragmentación
SYSTEM	Diccionario de Datos	Cero
TOOLS	Aplicaciones	Muy baja
DATA	Segmentos de Datos	Baja
INDEX	Segmentos de Índice	Baja
RBS	Segmentos rollback	Alta
TEMP	Segmentos temporales	Muy alta

La estructura recomendada para los tablespaces, sus usos y sus riesgos de fragmentación son mostrados en la tabla. Los siguientes son los diferentes tipos de objetos en un orden creciente de riesgo de fragmentación:

- Objetos del diccionario de datos. Al nunca ser borrados no podrán fragmentar el tablespace.
- Espacio usado por repositorios de aplicaciones. Son poco propensos a ser fragmentados.
- Segmentos de datos e índices usados por las aplicaciones escritas por el usuario. Tienen más riesgo a ser fragmentados que los repositorios de aplicaciones.
- Segmentos rollback. Causan fragmentación en sistemas con gran actividad de actualizaciones.
- Segmentos temporales en tablespaces permanentes. Como liberan espacio frecuentemente, por lo tanto deben localizarse en tablespaces separados.

Administración de tablas

Almacenando Datos de Usuario

Los datos dentro de una base de datos pueden ser almacenados en Tablas regulares, Tablas particionadas, Tablas de índice organizado y Tablas clustered.

Estructura de un Registro

Los datos de un registro se almacenan en bloques de la base de datos como registros de longitud variable. Cada registro en una tabla tiene:

- Un encabezado de registro.
- Datos de registro: Para cada columna, el servidor almacena la longitud de la cadena y el valor. Es necesario un byte para almacenar la longitud de la columna siempre y cuando esta no exceda 250 bytes, una columna más grande necesita tres bytes de longitud.

Tipos de datos del sistema

Cada columna dentro de una tabla debe tener asociado un tipo de dato, siendo la labor del diseñador de la base de datos, el de encontrar el mejor tipo de dato que satisfaga las necesidades de almacenamiento y recuperación de información.

Tipo en SQL99	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server	Descripción
tinyint	tinyint			tinyint	tinyint	
smallint	smallint	smallint	smallint (lo convierte a number)	smallint	smallint	Entero con signo de 2 bytes
int, integer	int,integer	integer	int (lo convierte a number)	Int	int	Entero con signo de 4 bytes
float()	float		float()	float	float	Número de punto flotante
double	double	double precision	double precision (lo convierte a float)	double precision	Double precision (se convierte en float)	Número Doble
real		real	real (Lo convierte a float)	real	real	Número Real
numeric(p,d)	numeric(p,d)	numeric(p,d)	number(p,d)	numeric(p, d)	decimal(p,d)	Numérico con precisión p y d decimales
character varying(n)	varchar(n)	varchar(n)	varchar2(n)	varchar(n)	varchar(n)	Carácter de longitud variable
char, character(n)	char(n)	char(n)	char(n)	char(n)	char(n)	Cadena de caracteres de longitud fija
date	date	date				Fecha sin hora del día
time	time	time				Hora del día
timestamp	timestamp	timestamp	date	datetime	datetime	Fecha y hora del día
Boolean		Boolean		Bit	bit	Valor booleano
Blob	blob	bytea	blob	image	image	Binary large object
Clob	text	text	clob	text	text	Character large object
Interval		interval				Intervalo de tiempo

Oracle Data types
Oracle proporciona data types para almacenar datos escalares, colecciones y relaciones.

Datatypes escalares

Datos carácter
Los datatype carácter de longitud fija, como CHAR y NCHAR, se almacenan con un bloque de blancos. El tamaño máximo está determinado por el número de bytes requeridos para almacenar un carácter y el límite superior es de 2000 bytes por registro.

Datos numéricos
Los números siempre se almacenan como datos de longitud variable. Pueden almacenarse hasta 38 dígitos significativos.

Los data types numéricos requieren: Un byte para el exponente, Un byte para cada dos dígitos significativos en la mantissa y Un byte para números negativos si el número de dígitos significativos es menor de 38 bytes.

Data Type DATE
El servidor almacena fechas en campos de longitud fija de 7 bytes.

Data type RAW
Este tipo de datos permite el almacenamiento de pequeños datos binarios.

Data types para almacenar objetos grandes

LONG, LONG RAW	LOB
Columna única por tabla	Múltiples columnas por tabla.
Mayor a 2 gigabytes	Mayor a 4 gigabytes.
SELECT devuelve datos	SELECT regresa el localizador.
Datos almacenados en línea	Datas almacenados en línea o fuera de línea.
No soporta tipos objeto	Soporta tipos objeto.
Acceso secuencial a pedazos	Permite que los datos sean almacenados en un segmento y tablespace separados, o en un archivo host.

Oracle proporciona 6 datatypes para almacenar LOBs:

- CLOB y LONG. Datos carácter de ancho fijo grande.
- NCLOB. Datos de grupo de caracteres nacionales de ancho fijo grande.
- BLOB y LONG RAW. Datos de almacenamiento no estructurado.
- BFILE. Datos de almacenamiento no estructurado en archivos de sistema operativo.

Data Type ROWID

ROWID es una seudo columna que tiene las siguientes características:

Es un identificador único para cada registro en la base de datos, no se almacena explícitamente como un valor columna, puede usarse para localizar el registro, proporciona la forma más rápida de acceso a un registro en una tabla y son almacenados en índices para especificar registros con un grupo de valores llave dado.

Formato ROWID

ROWID necesita 10 bytes de almacenamiento en disco y se despliega usando 18 caracteres. Consiste de los siguientes componentes:

- Data Object Number. Es único y es asignado a cada objeto cuando este es creado (32 bits)
- Relative File Number. Es único para cada archivo dentro de un tablespace (10 bits)
- Block Number. Representa la posición del bloque que contiene el registro dentro del archivo. (22bits)
- Row Number. Es la posición del slot del directorio de registros en el encabezado del bloque(10 bits)

```
SELECT NIVEL ROWID
FROM NIVELES;
```

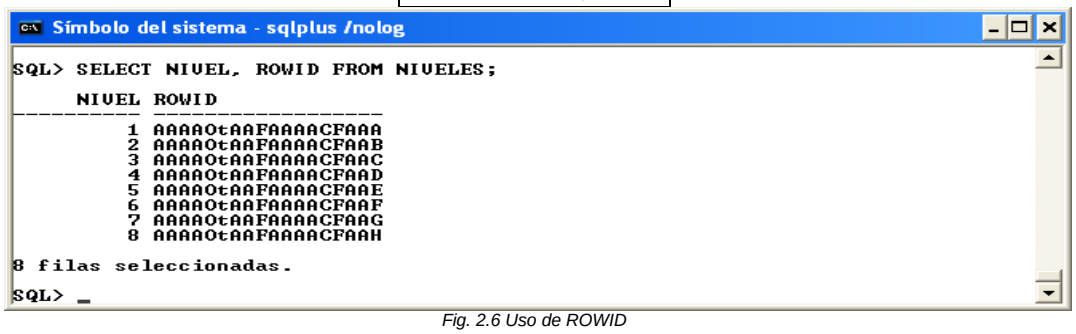


Fig. 2.6 Uso de ROWID

Creando una Tabla
Use el siguiente comando para crear una tabla:

```
CREATE TABLE [schema.] table
(column datatype [, column datatype]…)
[TABLESPACE tablespace]
[PCTFREE integer]
[PCTUSED integer]
[INITRANS integer]
[MAXTRANS integer]
[STORAGE storage-clause]
[LOGGING | NOLOGGING]
[CACHE | NOCACHE]]
```

Ejemplo:

```
CREATE TABLE INFANTES (
      curp_infante        CHAR(18) NOT NULL,
      timestamp           DATE NOT NULL,
      nombre              VARCHAR2(70) NOT NULL,
      ap_pat              VARCHAR2(70) NOT NULL,
      fecha_nacimiento    DATE NOT NULL,
      peso_nacimiento     NUMBER(2,1) NOT NULL,
      estatura_nacimiento NUMBER(2,2) NOT NULL,
      calle_num           VARCHAR2(100) NOT NULL,
      colonia             VARCHAR2(120) NOT NULL,
      cp                  NUMBER(5,0) NOT NULL,
      id_municipio        NUMBER(3) NOT NULL,
      id_estado           NUMBER(2) NOT NULL,
      num_estancia        NUMBER(3) NOT NULL,
      sexo                CHAR(1) NOT NULL,
      fecha_ingreso       VARCHAR2(10) NOT NULL,
      leche_materna       CHAR(1) NOT NULL,
      fecha_egreso        DATE NULL,
      id_pais_vive        NUMBER(3) NOT NULL,
      ap_mat              VARCHAR2(70) NULL,
      id_pais_nacionalidad NUMBER(3) NULL,
      curp_infante_hermano CHAR(18) NULL,
      status_bl           CHAR(1) NULL
                          CONSTRAINT check_status_baja_logica257
                              CHECK (status_bl IN ('D'))
)
            PCTFREE 48
            PCTUSED 45
            INITRANS 1
            MAXTRANS 14
            TABLESPACE TS_VOLATILES
            STORAGE  (
                        INITIAL 1004
                        NEXT 100
                        MINEXTENTS 1
                        MAXEXTENTS 2
                        PCTINCREASE 0
                     )
            CACHE
;
```

Donde:

schema	Propietario de la tabla.
Table	Nombre de la tabla.
column	Nombre de la columna.
Data type	Tipo de data type de la columna.
TABLESPACE	Tablespace en donde será creada la tabla.
LOGGING	Específica que la creación de la tabla será loggeada en el archivo redo log.
NOLOGGING	La creación de la tabla no será loggeada en el archivo redo log.
CACHE	Específica que el bloque recuperado para esta tabla es ubicado al final de la lista LRU usada más recientemente en el buffer caché.
NOCACHE	Específica que los bloques recuperados para esta tabla son ubicados al final de la lista LRU usada más lejanamente en el buffer caché.

Script de parámetros STORAGE

```
STORAGE( INITIAL entero{K|M} NEXT entero{K|M}
{MINEXTENTS entero} {MAXEEXTENTS entero|UNLIMITED}
{PCTINCREASE %entero});
```

Use la cláusula CACHE para tablas pequeñas utilizadas frecuentemente

Concepto de Migración de Registros y Encadenamiento

Migración de Registro

Si el PCTFREE se configura a un valor pequeño, puede haber espacio insuficiente en un bloque para acomodar un registro que crece como resultado de una actualización. Cuando esto sucede, el servidor moverá el registro entero a un nuevo bloque y dejará el apuntador del bloque original en la nueva localidad. Este proceso es referido como una migración de registro.

Encadenamiento de Registro

Ocurre cuando un registro es demasiado grande para encajar en cualquier bloque. En este caso, el servidor divide el registro en pequeños pedazos llamados piezas de registro. Cada pieza de registro se almacena a lo largo del bloque usando apuntadores.

Controlando el espacio utilizado por tablas
Cambiando los parámetros de almacenamiento y utilización de bloques

Algunos de los parámetros de almacenamiento y cualquiera de los parámetros de utilización de bloque pueden modificarse utilizando el comando ALTER TABLE.

```
ALTER TABLE [squema.] tabla
{[claúsula_almacenamiento]
[PCTFREE integer]
[PCTUSED integer]
[INITRANS integer]
[MAXTRANS integer]}
```

Efectos de parámetros de almacenamiento de encadenamiento
Los parámetros que pueden modificarse y las implicaciones de cambiarlos son las siguientes:

NEXT
Cuando el servidor Oracle aloja otras extensiones de la tabla, el nuevo valor podrá usarse. Los subsecuentes tamaños de extensiones serán incrementados por PCTINCREASE.

PCTINCREASE
El nuevo valor será usado para recalcular NEXT cuando la siguiente extensión sea alojada.

MINEXTENTS
Su valor puede cambiarse a cualquier valor que sea mayor o igual al número actual de extensión en la tabla. Este valor solo tendrá efecto si la tabla se trunca.

MAXEXTENTS
Su valor puede configurarse a cualquier valor mayor o igual que el número actual de extensión.

Restricciones
- El valor de INITIAL no puede modificarse para la tabla.

- El valor de NEXT especificado será redondeado a un valor que sea múltiplo del tamaño del bloque mayor o igual que el valor especificado.

Los efectos de los parámetros de utilización de bloque de encadenamiento son los siguientes

PCTFREE
Un cambio a PCTFREE afectará las inserciones futuras. Los bloques que no son usados para inserts porque ya habían sido llenos a (100-PCTFREE) no podrán ser afectados hasta que regresen a la lista libre.

PCTUSED
Cualquier cambio a PCTUSED podrá afectar todos los bloques en la tabla.

INITRANS
Un cambio a INITRANS solamente afectará nuevos bloques.

MAXTRANS
Un cambio a MAXTRANS afectará todos los bloques en la tabla.

Asignando extensiones manualmente
Las extensiones pueden asignarse manualmente para controlar la distribución de extensiones de una tabla a través de archivos.

Use los siguientes comandos para asignar una extensión a una tabla:

```
ALTER TABLE [schema.] tabla
ALLOCATE EXTENT [(SIZE integer [K|M]]
[DATAFILE 'filename'] )]
```

High Water Mark
La marca de agua alta para una tabla indica el último bloque que se usó para la tabla, no se restablece cuando los registros se borran de la tabla, se guarda en el encabezado del segmento de la tabla. Cuando el servidor Oracle realiza full table scan, lee todos los bloques hasta los que están arriba de la marca de agua alta.

Encontrando la marca de agua alta
(DBMS_SPACE.UNUSED_SPACE)

El paquete DBMS_SPACE contiene un procedimiento que puede usarse para encontrar la marca de agua alta y el número de bloques sobre ésta. Aunque esta información puede obtenerse desde el diccionario de datos después de analizar la tabla, el paquete DBMS_SPACE habilita un acceso rápido a la información sin afectar el comportamiento de la optimización.

El siguiente bloque PL/SQL puede usarse para encontrar e imprimir el número de bloques asignados a la tabla y el número bloques sin usar:

```
DECLARE
      v_owner VARCHAR(30):= 'ESTANCIA';
      v_segment_name VARCHAR(30):= 'INFANTES';
      v_segment_type VARCHAR(30):= 'TABLE';
      v_total_blocks NUMBER;
      v_total_bytes NUMBER;
      v_unused_blocks NUMBER;
      v_unused_bytes NUMBER;
      v_last_used_extent_file_id NUMBER;
      v_last_used_extent_block_id NUMBER;
      v_last_used_block NUMBER;
BEGIN
       dbms_space.unused_space (v_owner,
        v_segment_name,
        v_segment_type,
        v_total_blocks,
        v_total_bytes,
        v_unused_blocks,
        v_unused_bytes,
        v_last_used_extent_file_id,
        v_last_used_extent_block_id,
        v_last_used_block, );
        dbms_output.put_line (INITCAP (v_segment_type)||': '||v_owner||'.'||v_segment_name);
        dbms_output.put_line ('Total blocks: '||TO_CHAR(v_total_blocks));
        dbms_output.put_line ('Bloques sobre HWM: '||TO_CHAR (v_unused_blocks));
END /
```

Reasignando el espacio sin uso
Es posible reasignar el espacio de la tabla manualmente.

Use el siguiente comando para reasignar espacio sin uso para una tabla:

```
ALTER TABLE [schema.] table
DEALLOCATE UNUSED [KEEP integer [K|M]]
```

Donde:

KEEP	Especifica el número de bytes sobre la marca de agua alta que deben retenerse.

El uso frecuente de este comando puede llevar a la fragmentación del espacio en data files. Para evitar este problema, ponga MINIMUM EXTENT, para el tablespace.

Para liberar todo el espacio debajo de la marca de agua alta, incluso si la marca está debajo de MINEXTENTS, utilice KEEP 0.

Truncando una tabla
Truncando una tabla borra todos los renglones y libera espacio utilizado:

```
TRUNCATE TABLE [schema.] tabla
[{DROP|REUSE} STORAGE]
```

Eliminando tablas
Se puede eliminar la tabla si ya no se necesita o si será reorganizada

```
DROP TABLE scott.empleados
CASCADE CONSTRAINTS;
```

Al eliminar una tabla, se liberan las extensiones usadas por la tabla.

Comando ANALYZE TABLE
El propósito principal de este comando es unir las estadísticas de una tabla usadas por el optimizador y guardarlas en el diccionario de datos.

Algunos de los usos del comando son: el borrar características de la tabla del diccionario de datos, validar la estructura de la tabla e Identificar la migración y encadenamiento de registros de la tabla.

Validando la estructura de la tabla
Cuando se valida la estructura de una tabla sus bloques son verificados por integridad.

Los parámetros DB_BLOCK_CHECKSUM pueden ponerse a TRUE para calcular una verificación de suma y almacenarlo en el encabezado de cada bloque de datos cuando escriba esto a disco.

Detectando la migración de registros

El servidor reúne estadísticas basadas en datos muestra y en actualizaciones del diccionario de datos.

```
ANALYZE TABLE scott.employees
ESTIMATE STATISTICS;
```

* ANALYZE puede ser usado para detectar migración o encadenamiento de registros.

Use el siguiente comando para generar estadísticas:

```
ANALYZE TABLE [schema.] table
{COMPUTE STATISTICS |ESTIMATE STATISTICS [SAMPLE integer {ROWS | PERCENT}]
```

Donde:

COMPUTE STATISTICS	Generará estadísticas basadas en un escaneo de tabla completo.
ESTIMATE STATISTICS	Genera estadísticas basadas en una muestra.

Cuando son generadas las estadísticas, la columna CHAIN_CNT de la vista DBA_TABLES es actualizada con el número de registros que son encadenados o migrados. Si se migra un gran número de registros, entonces la tabla necesita ser reorganizada para eliminar la migración.

Recuperando información de una tabla
La información de las tablas puede obtenerse del diccionario de datos.

El data object number y la localización del encabezado de una tabla para las tablas propiedad de SCOTT, pueden verse con:

```
SELECT t.table_name, o.data_object_id,s.header_file, s_header_block
FROM dba_tables t, dba_objects o, dba_segments s
WHERE  t.owner         =o.owner
AND       t.table_name=o.object_name
AND       t.owner         =s.owner
AND       t.table_name=s.segment_name
AND       t.owner         ='SCOTT';
```

Información de Encadenamiento y Uso de Bloque
Use la siguiente consulta para obtener el número de registros encadenados o migrados, para traer el número de bloques que están arriba de la marca de agua superior y para traer el número de bloques por debajo de esta misma marca:

```
SELECT blocks AS HWM, empty_blocks, chain_cnt AS "Chained Blocks"
FROM dba_tables
WHERE  owner         ='ESTANCIA'
AND       table_name='INFANTES';
```

Distribución del espacio asignado
El número de extensiones, localizaciones y tamaños se consultan en DBA_EXTENTS.

El ejemplo siguiente muestra el número de extensiones y el total de bloques usados por una tabla en cada archivo en la base de datos:

```
SELECT file_id, COUNT(*) AS Extents, SUM(blocks) AS Blocks
FROM dba_extents
WHERE  owner              ='ESTANCIA'
AND       segment_name='INFANTES'
GROUP BY file_id;
```

Paquete DBMS_ROWID
Oracle proporciona un paquete conocido como DBMS_ROWID, el cual es creado desde el script dbmsutil.sql, que activado es conocido como catproc.sql. El paquete proporciona funciones que pueden usarse para convertir entre formatos ROWID y traducirlos entre ROWID y sus componentes individuales. Las funciones usadas comúnmente:

Nombre de la Función	Descripción
ROWID_CREATE	Crea un ROWID desde componentes individuales.
ROWID_OBJECT	Regresa el identificador del objeto para un ROWID.
ROWID_RELATIVE_FNO	Regresa el número de archivo relativo para un ROWID.
ROWID_BLOCK_NUMBER	Regresa el número de bloque para un ROWID.
ROWID_ROW_NUMBER	Regresa el número de registro para un ROWID.
ROWID_TO_ABSOLUTE_FNO	Regresa el número de archivo absoluto para un ROWID.
ROWID_TO_EXTENDED	Convierte un ROWID de restringido a extendido.
ROWID_TO_RESTRICTED	Convierte un ROWID de extendido a restringido.

Obteniendo Componentes ROWID
Use la siguiente consulta para obtener la localidad física de los registros en una tabla:

```
SELECT deptno, ROWID
        DBMS_ROWID.ROWID_OBJECT(ROWID) AS OBJECT,
        DBMS_ROWID.ROWID_RELATIVE_FNO(ROWID) AS “RELATIVE FILE”,
        DBMS_ROWID.ROWID_BLOCK_NUMBER(ROWID) AS BLOCK
FROM scott.departments;
```

Encontrando el número absoluto de archivo
El siguiente script regresa los números absolutos de archivo de los registros en SCOTT.DEPT:

```
SELECT deptno, ROWID,
        BMS_ROWID.ROWID_TO_ABSOLUTE_FNO(ROWID, ‘SCOTT’, ‘DEPARTMENT’) AS “FILE”
FROM scott.departments;
```

Utilizando clusters y tablas de índice organizado

Distribución de renglones dentro de una tabla
En una tabla regular, se tiene un control muy limitado de la distribución de las filas de datos.

Los cluster ofrecen un grado de control sobre las filas almacenadas. Cuando se utiliza un cluster el servidor almacena todas las filas que tienen el mismo valor llave en el mismo bloque.

Clusters

Un cluster es utilizado para almacenar grupos de filas dentro del mismo bloque del servidor.

Características de los Clusters

Los cluster tienen las siguientes características:

- Tienen una llave cluster. Identifica las filas que necesitan ser almacenadas juntas.

- La llave cluster puede contar con una o más columnas.

- El clustering es un mecanismo transparente a las aplicaciones usadas en las tablas.

- Actualizar una de las columnas en la llave cluster repercute físicamente en reubicar la fila.

- La llave cluster es independiente de la llave primaria.

- El acceso aleatorio de los datos cluster suele ser más rápido, pero una revisión completa de tablas clusters es generalmente más lenta.

Tipos de Cluster

Existen dos tipos de cluster: Cluster Índice y Cluster Hash.

Cluster Índice
Utiliza un índice llamado cluster index, para mantener los datos dentro del cluster:

- Debe estar disponible para almacenar, acceder o mantener datos en un índice cluster.

- Se usa para señalar el bloque que contenga las filas con un valor de llave dado.

- Almacenan las llaves NULL.

- Si varias filas en un cluster índice tienen la misma llave cluster, ésta no se repite para cada fila. En una tabla con un mayor número de filas por valor de llave, utilizar un cluster índice puede reducir la cantidad de espacio necesaria para almacenar datos.

Cluster Hash
Un Cluster Hash utiliza una función para calcular la ubicación de una fila. La función Hash usa la llave cluster y puede ser definida por el usuario o generado por el sistema.

Cuando una fila es insertada en la tabla en un cluster Hash: Las columnas de llave Hash son utilizadas para procesar un valor Hash y la fila es almacenada basándose en el valor Hash.

CREANDO CLUSTER'S
Use los siguientes comandos para crear un cluster:

```
CREATE CLUSTER [ schema. ] cluster (column datatype [, column datatype ] . . . )
PCTFREE integer ] [PCTUSED integer ] [INITRANS integer ] [MAXTRANS integer ] [SIZE integer [ K | M ]
] [ storage-clause ] [TABLESPACE tablespace ] [INDEX ]
```

Donde:

Schema	Es el dueño del cluster.
Cluster	Es el nombre del cluster.
Column	Es el nombre de la columna llave.
Data type	Es el tipo de datos y el tamaño de la columna llave.
SIZE	Espacio requerido por todas las filas correspondientes a un valor de llave.
INDEX	Especifica que está en un cluster índice.

Creando el Cluster Indexado

Utilice los siguientes comandos para crear un cluster indexado:

```
CREATE INDEX [ schema. ] index ON CLUSTER [ schema. ] cluster
[ PCTFREE integer ] [ INITRANS integer ] [ MAXTRANS integer ]
[ TABLESPACE tablespace ] [ storage-clause ]
```

Las columnas llave no necesitan ser especificadas para un índice cluster porque ya están definidas al crear el cluster. Coloque el índice cluster en un tablespace diferente del usado para crear el cluster.

Creando Tablas en los Clusters

Especifique el nombre del cluster para crear tablas en el cluster:

```
CREATE TABLE [schema.] table
( column_definition
[, column_definition ]. . .
[, [CONSTRAINT constraint] out_of_line_constraint ] . . .
)
CLUSTER [schema.] cluster (column [, column ] . . . )
```

Donde:

CLUSTER	Especifica que la tabla debe ser colocada en un cluster.

Una tabla que esta colocada en un cluster no puede tener atributos físicos por sí misma, porque no es un segmento propio y es una parte del cluster.

Manteniendo Clusters
Algunas actividades de mantenimiento para los clusters son:

- Cambiar el espacio de bloque y almacenamiento usando parámetros.

- Cambiar SIZE para clusters indexados.

- Asignar y desasignar espacio.

Acciones de mantenimiento que pueden ser aplicadas en clusters indexados

Comando	**Operación**	**Cluster indexado**	**Cluster Hash**
ALTER CLUSTER	Cambia los parámetros de utilización de bloques	SI	SI
	Cambia la configuración de almacenamiento excepto INITIAL	SI	SI
	Asigna extensión	SI	SI
	Desasigna espacio sin uso	SI	SI
	Cambia el SIZE	SI	NO
	Cambia HASHKEYS y HASH IS	N/A	NO
TRUNCATE CLUSTER	Libera espacio	SI	NO
	Reutiliza espacio	SI	NO

Los comandos ALTER CLUSTER; TRUNCATE CLUSTER y ANALYZE CLUSTER siguen la misma sintaxis que los comandos para una tabla.

Eliminando clusters

Un cluster puede ser eliminado después de que todas las tablas en el cluster son eliminadas.

Use el siguiente comando para eliminar un cluster:

```
DROP CLUSTER [schema.] cluster
[ ] INCLUDING TABLES [ CASCADE CONSTRAINTS] ]
```

Donde:

CASCADE CONSTRAINTS	Debe ser usada si cualquier tabla en el cluster referenciada por una restricción de llave foránea en una tabla que no es parte del cluster.
INCLUDING TABLES	Use INCLUDING TABLES para eliminar tablas contenidas en el cluster

Distribución de Valores llave

Los Clusters están generalmente mejor situados en situaciones donde la frecuencia de la ocurrencia en los valores llave es uniforme. .

Frecuencia de actualización de las columnas llave

Cuando es actualizado el valor de la llave en un cluster, el renglón puede ser movido físicamente a otro bloque lo que provoca una degradación en el desempeño.

Frecuencia de Joins

Las tablas que son frecuentemente unidas en una consulta usando una relación de llaves foráneas son una buena opción para implementar en un cluster.

.**Condición de consulta**

La función hash puede ser usada para recuperar un dato solo si el valor de la llave es conocido. Si las consultas usan igualdad en los predicados en la llave cluster, ellos se pueden beneficiar del numerado. Sin embargo, es posible crear un índice en la llave del cluster hash si muchas consultas usan rangos de búsqueda en las columnas llave del cluster hash.

Recuperando información de los Clusters.

Clusters y columnas llave cluster
Para encontrar los nombres de los clusters indexados propiedad del usuario ESTANCIA y las columnas llave cluster, use la siguiente consulta:

```
SELECT c.cluster_name, c.cluster_type, c.key_size,
          cc.Column_name, cc.data_type,
          decode (cc.data_type, 'number',
          decode (cc.data_precision,NULL,NULL,
          cc.data_presicion || ',' || cc.data_scale),
          'DATE', NULL, cc.data_length) AS "COLSIZE"
FROM    dba_clusters c, dba_tab_columns cc
WHERE   c.owner          =cc.owner
AND     c.cluster_name= table_name
AND     c.owner          = 'ESTANCIA';
```

Consulte el DBA_CLU_COLUMNS para obtener una lista de clusters, las tablas del cluster, y la comparación de los nombres de las columnas:

```
 SELECT *
FROM dba_clu_columns
WHERE owner='ESTANCIA'
ORDER BY cluster_name, table_name;
```

La columna FUNCTION para un cluster hash en DBA_CLUSTERS puede tener uno de los siguientes valores:

DEFAULT2	Si el servidor interno de funciones Oracle es usado.
COLUMN	Si la llave columna es auto especificada en la cláusula HASH IS

Tablas de índice organizado

Estructura de almacenamiento

Guarda todos los datos de una tabla dentro de una estructura de árbol B.

Ingresando a una tabla de índice organizado

El acceso de índices a una tabla regular requiere que uno o más bloques de índice sean leídos para recuperar el ROWID y el I/O en la tabla basada en el ROWID.

En contraste, leer una tabla de índice organizado solo requiere leer bloques de índice porque el renglón completo se encuentra disponible en el nodo hoja.

Usando tablas de índice organizado

Creando tablas de índice organizado

Una tabla de índice organizado es útil para aplicaciones de recuperación de información, aplicaciones especiales y para aplicaciones de procesamiento analítico en línea (OLAP).

La organización indexada es útil para una tabla que es accesada frecuentemente usando la llave primaria y que tiene solo unas pocas y cortas columnas no llave.

Use el siguiente comando para definir una tabla de índice organizado:

```
CREATE TABLE [schema.] table
     (column-definition [, column-definition ] ….
      [, out-of-line-constraint [, out-of-line-constraint ] … ])
ORGANIZATION INDEX
     [ PCTFREE integer ]
     [INITRANS integer ]
     [MAXTRANS integer ]
     [ storage-clause ]
[ TABLESPACE tablespace ]
      [ PCTTHRESHOLD integer]
      [INCLUDDING column] ]
      [ OVERFLOW segment_attributes_clause ]
```

Donde:

Schema	Es el dueño del clúster.
Table	Es el nombre de la tabla.
ORGANIZATION INDEX	Especifica que es una tabla de índice organizado
PCTTHRESHOLD	Especifica el porcentaje de espacio reservado en el bloque de índice. PCTTHRESHOLD es por default 50 y debe ser un valor desde 0 a 50.
INCLUDING column	Especifica una columna a la cual dividir una fila de tabla de índice organizado y porciones de desbordamiento.
OVERFLOW	Especifica que los datos en las filas de una tabla de índice organizado que exceden el umbral especificado son situados en un segmento de datos definido por segment_attributes_clause, los cuales especifican los parámetros del tablespace, almacenamiento y utilización de bloques.

- Una llave primaria debe ser especificada por una tabla de índice organizado.

- Si PCTTHRESHOLD está definido y un segmento de desbordamiento no está especificado, las filas que excedan el umbral son rechazadas.

Desbordamiento de Filas
Un renglón grande en una tabla de índice organizado podría destruir el almacenamiento robusto de los renglones en el índice. Este problema es resuelto por el uso de un área de desbordamiento. **Segmentos creados para una tabla de índice organizado**

Cuando es creada una tabla de índice organizado especificando la cláusula OVERFLOW, se crea lo siguiente:

- Una tabla "lógica" con el nombre definido en la cláusula CREATE TABLE.

- Un índice con el mismo nombre que la restricción de llave primaria, en el tablespace definido en la sentencia CREATE TABLE.

- Una tabla para acomodar los pedazos de filas de desbordamiento. El nombre de esta tabla es SYS_IOT_OVER_n, donde n es OBJECT_ID de la tabla de índice organizado.

Operaciones de una tabla de índice organizado
Una tabla de índice organizado puede ser utilizada como cualquier otra tabla.

Recuperando información acerca de las tablas de índice organizado

Recuperando información IOT desde el Diccionario de Datos

Use la siguiente consulta para listar las tablas de índice organizado y la información de su estructura:

```
SELECT t.table_name AS "IOT", o.table_name AS "Overflow",
          i.index_name AS "Index",
          o.tablespace_name AS "Overflow TS",
           i.tablespace_name AS "Index TS", i.pct_threshold
FROM dba_tables t, dba_tables o, dba_indexes I
 WHERE   t.owner         =o.owner
 AND      t.table_name=o.iot_name
 AND      t.owner         = i.owner
AND       t.table_name = i.table_name
AND       t.owner         = 'SCOTT';
```

Administrando índices

Un índice es una estructura de árbol que permite el acceso directo a un renglón en una tabla.
Los índices pueden ser clasificados basados en su diseño lógico o en su implementación física. La clasificación lógica agrupa a los índices desde una perspectiva de aplicación, mientras que la clasificación física se deriva en la forma en que los índices son almacenados.

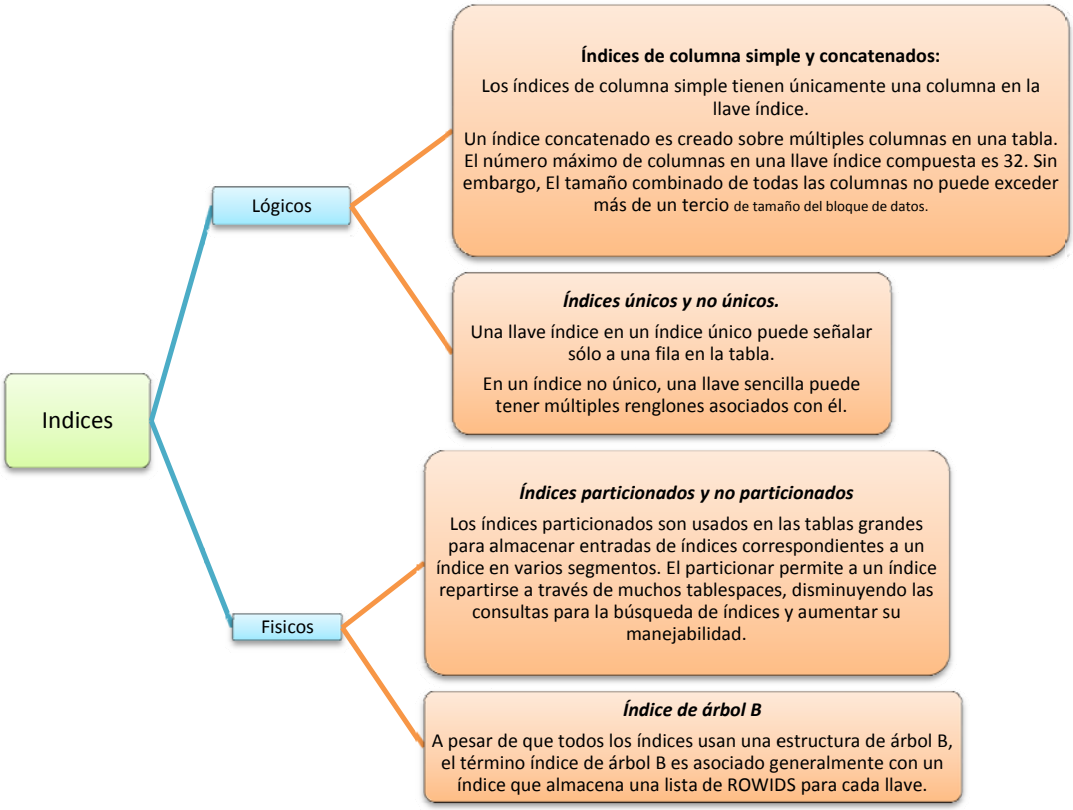


Fig. 2.7 Clasificación de los índices

Formato de entradas de la hoja del índice
La entrada de un índice está conformada de los siguientes componentes:

- Cabecera de la entrada. Almacena el número de columnas y asegura la información.
- Valores de longitud de la columna llave. Definen el tamaño seguida por el valor de la columna.
- ROWID de un renglón.

Efecto de las operaciones DML en un índice
- Inserta los resultados de las operaciones de una entrada de un índice en el bloque apropiado.
- La eliminación de renglones resulta sólo en la eliminación lógica de la entrada de índice.
- Actualizaciones en columnas llave originan una eliminación lógica y una inserción en el índice.

Índice bitmap
Los índices Bitmap son más ventajosos que los de árbol B cuando:
- Una tabla tiene millones de renglones y las columnas llave tiene baja cardinalidad.
- Las consultas utilizan múltiples condiciones WHERE involucrando el operador OR.
-Hay una actividad de sólo lectura o de baja actualización en las columnas llave.

Estructura de un índice Bitmap
Un índice Bitmap está organizado también como un árbol B, pero el nodo hoja almacena un bitmap por cada valor de llave en lugar de una lista de ROWIDS.

Usando un índice bitmap
El ROWID de inicio y el bitmap son usados para localizar los renglones que contienen el valor de la llave. Cuando se hacen los cambios a la columna llave en la tabla, los bitmaps deben ser modificados y no puede ser actualizada por otras transacciones hasta que la esta termina.

Comparando los índices de árbol b y bitmap

Árbol B	**Bitmap**
Ideal para columnas de alta cardinalidad	Ideal para columnas de baja cardinalidad
Actualizaciones para columnas llaves no costosos	Actualizaciones para columnas llaves muy costosas
Ineficiente para consultas que utilicen predicados OR	Eficiente para consultas que utilicen predicados OR
Útil para OLTP	Útil para DSS (Decision Support System
	Es recomendable utilizarlos cuando las tablas que sufren pequeñas inserciones o actualizaciones, en el entorno del data warehouse, donde los datos son normalmente estáticos
Ocupa mayor espacio	utiliza solo una centésima parte del espacio que ocuparía un árbol b

Creando Índices

Creando índices normales de árbol b

Use el siguiente comando para crear el índice de árbol B:

```
CREATE [ UNIQUE ] INDEX [schema.] index
ON [schema.] table
(column [ ASC | DESC ] [, column [ASC | DESC ]] ...)
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGING| NOLOGGING ]
[ NOSORT]
```

Donde:

UNIQUE	Es utilizado para especificar un índice único.
schema	Es el propietario de índice/tabla.
Índex	Es el nombre del índice.
Tabla	Es el nombre de la tabla.
Column	Es el nombre de la columna.
ASC/DESC	Para la compatibilidad sintáctica con otras Bases de Datos.
TABLESPACE	Identifica el tablespace en el que será creado el índice.
PCTFREE	Es la cantidad de espacio reservado en cada bloque.
INITRANS	Especifica el número de entradas de operaciones pre alojadas en cada bloque (el predeterminado y mínimo es 2)
MAXTRANS	Número de entradas de operaciones que pueden ser alojadas en cada bloque (El predeterminado es 255)
STORAGE	Cláusulas de almacenamiento.
LOGGING	La creación del índice es accesada en el archivo redo log.
NOLOGGING	La creación del índice no es accesada en el archivo redo log
NOSORT	Los renglones son almacenados en la base de datos en orden ascendente.

* PCTUSED no puede ser especificado para un índice.

Consideraciones

- Minimice el número de índices necesario en tablas volátiles
- Colocar los índices en tablespaces separados, no en aquellos que contengan segmentos rollback, segmentos temporales y tablas.
- Usar tamaños de extensiones uniformes: múltiplos de 5 bloques o tamaño MINIMUM EXTENT para espacios de tabla.

Para minimizar la fragmentación, use algunos tamaños de extensión estándar que son múltiplos de 5*DB_BLOCK_SIZE.

- Considerar NOLOGGING para índices grandes.
- Las entradas de índices son pequeñas comparadas a los renglones que indexan, los bloques de índices tienden a tener más entradas por bloque. Por esta razón, INITRANS suele ser más alta en índices que en tablas correspondientes.

Índices y PCTFREE
El parámetro PCTFREE para un índice funciona diferente que para una tabla, Este parámetro es usado solamente durante la creación del índice para reservar espacio para las entradas de índice que puedan necesitar ser insertadas en el mismo bloque de índice.
Las entradas de índice no son actualizadas. Cuando una columna llave es actualizada, esto involucra una eliminación lógica de la entrada del índice y una inserción.

* Utilice un PCTFREE bajo para índices en columnas que se incremente monótonamente.

Cuando el valor de una columna indexada de un renglón insertado toma cualquier valor, se puede proporcionar un PCTFREE más alto. En este caso, es útil especificar un valor de PCTFREE como se indica en la siguiente fórmula:

$$\frac{Maximum\_number\_of\_rows - Initial\_number\_of\_rows}{Maximum\_number\_of\_rows} \times 100$$

Creando índices bitmap

```
CREATE BITMAP INDEX [schema.] index
ON [schema.] table
(column [ ASC | DESC ] [, column [ASC | DESC ]] ...)
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGING| NOLOGGING ]
[ NOSORT]
```

* El índice de mapa de bits no puede ser único.

CREATE_BITMAP_AREA_SIZE
El parámetro de inicialización CREATE_BITMAP_AREA_SIZE determina la cantidad de espacio que será utilizada para almacenar en memoria los segmentos bitmap. El valor preestablecido es de 8MB. Un valor mayor puede conducir a una creación más rápida del índice. Si su cardinalidad es muy pequeña, este valor puede cambiarse a un valor pequeño.

Reorganizando índices

Cambiando los parámetros de almacenamiento para índices
Algunos de los parámetros de almacenamiento y parámetros de utilización de bloques pueden ser modificados usando el comando ALTER INDEX.

```
ALTER INDEX [schema.]index
[ storage-clause ]
[ INITRANS integer ]
[ MAXTRANS integer ]
```

Asignando y desasignado espacio de índices
Asignación manual de espacio a un índice
Puede ser necesario añadir extensiones a un índice después de un periodo de alta actividad de inserción en una tabla. Añadir extensiones previene la extensión dinámica de índices y el resultado de la degradación en el desempeño.

Des asignación manual de espacio a un índice
Utilice la cláusula de DEALLOCATE del comando de ALTER INDEX para liberar el espacio no utilizado por encima de las marcas de agua en un índice.
Utilice el siguiente comando para asignar o desasignar el espacio de un índice:

```
ALTER INDEX [schema.]index
{ALLOCATE EXTENT ([SIZE integer [K|M]]
[DATAFILE `filename´])
| DEALLOCATE UNUSED [KEEP integer [ K|M ] ]}
```

* La asignación y la des asignación de espacio para un índice siguen las mismas normas que las que se utilizan en estos comandos para tablas.
* Truncar una tabla resulta truncar un índice asociado.

Reconstruyendo índices
Use este comando para: Mover un índice a un tablespace distinto, Mejorar la utilización de espacio removiendo las entradas eliminadas y Cambiar un índice de llave inversa a un índice de árbol B normal y viceversa.

```
ALTER INDEX scott.ord_region_id_idx
REBUILT
TABLESPACE indx02;
```

Utilice el siguiente comando para reconstruir un índice:

```
ALTER INDEX [schema.] index REBUILT
[ TABLESPACE tablespace ]
[ PCTFREE integer ]
[ INITRANS integer ]
[ MAXTRANS integer ]
[ storage-clause ]
[ LOGGING| NOLOGGING ]
[ REVERSE | NOREVERSE ]
```

El comando ALTER INDEX…REBUILD no puede utilizarse para cambiar un índice bitmap a uno de árbol B y viceversa. Los comandos REVERSE o NOREVERSE pueden especificarse sólo para índices de árbol B.

Examinando la validez de un índice
Analice el índice para: Detectar la corrupción de bloques y Poblar la vista INDEX_STATS con información acerca del índice.

```
ANALIZE INDEX [ schema.]index VALIDATE STRUCTURE
```

Después de correr este comando, se ejecuta la consulta INDEX_STATS para obtener información acerca del índice como se muestra en el siguiente ejemplo:

```
SELECT blocks,
       pct_used,
       distinct_keys
       lf_rows,
       del_lf_rows
FROM index_stats;
```

Reorganice el índice si tiene una proporción mayor de líneas eliminadas.

Eliminando índices
Elimine índices:
- Antes de una carga de volumen y recree un índice después de la carga.
- Cuando no son necesitados frecuentemente y constrúyalos cuando sea necesario.
- Existan índices inválidos.

Utilice el siguiente comando para omitir un índice:

```
DROP INDEX [schema.] index
```

Obteniendo la información de los índices
Las vistas del diccionario de datos DBA_INDEXES y DBA_IND_COLUMNS proporcionan la información de los índices y las columnas indexadas.

Revisando los índices y su validez
El siguiente comando verifica el nombre, tipo y estatus de los índices propiedad de SCOTT:

```
SELECT index_name,
       tablespace_name,
       index_type,
       Uniqueness,
       status
FROM dba_indexes;
WHERE owner= 'SCOTT';
```

Utilice la siguiente búsqueda para listar los nombres de todos los índices de llave inversa:

```
SELECT o.object_name
FROM dba_objects o
WHERE owner= 'SCOTT';
AND o.object_id IN (SELECT i.obj# FROM IND$i
                    WHERE BITAND(i.property,4) = 4);
```

Encontrando columnas en un índice

La siguiente búsqueda lista todos los índices propiedad del usuario SCOTT y muestra las tablas y columnas en las cuales están construidos los índices:

```
SELECT o.object_name, table_owner, table_name, column_name
FROM dba_objects o
WHERE owner= 'SCOTT';
ORDER BY index_name, column_position;
```

2.4 Seguridad de base de datos

Usuarios

Dominio de seguridad

Para dar seguridad a la base de datos el administrador define los nombres de los usuarios autorizados para accesar a la base de datos. Y define las configuraciones que aplican a cada uno de los usuarios.

Fig. 2.8 Dominio de Seguridad

La forma de autenticación se define al momento en que se define el usuario en la base de datos y puede modificarse después.

Cuotas del tablespace

La cuota de tablespace controla la cantidad de almacenamiento físico asignada a un usuario en el tablespace en la base de datos

Tablespace default

El tablespace default define la localización en donde se almacenan los segmentos creados por el usuario si el usuario no especifica un tablespace particular al momento de la creación del segmento.

Tablespace Temporal

El tablespace temporal define donde serán asignadas las extensiones por el servidor de Oracle si el usuario realiza una operación que requiere escribir ordenaciones de datos a disco.

Congelamiento de cuenta

Las cuentas pueden ser congeladas para prevenir que un usuario se registre en la base de datos. Esto puede configurarse para ocurrir automáticamente o el administrador de la base de datos puede activar o desactivar cuentas manualmente.

Límites de recursos

Se pueden poner límites al uso de recursos como el tiempo de CPU, I/O lógicas y el número de sesiones abiertas por un usuario. Los límites de recursos son discutidos en la lección “Administrando Perfiles”.

Privilegios directos

Los privilegios son utilizados para controlar las acciones que puede realizar un usuario en la base de datos.

Privilegios de roles

Se le pueden otorgar privilegios a un usuario indirectamente a través del uso de roles.
Esta lección cubre la definición de un usuario con el mecanismo de autenticación apropiado, limitando el uso de espacio para los usuarios en el sistema y controlando la activación de cuentas manualmente.

Pasos básicos para la dotar de seguridad a un usuario, los cuales realizamos con nuestro nuevo usuario admin_estancia, los cuales están marcados de la siguiente manera.

1) Creando un usuario
2) Creación de un perfil
3) Asignación de Perfiles a Usuarios
4) Otorgando Privilegios
5) Asignación de Roles

Creando los usuarios en la base de datos

Secuencia para la creación de usuarios
1. Elija un username y el mecanismo de autenticación
2. Identifique los tablespaces que necesita el usuario para almacenar objetos
3. Asigne la cuota para cada tablespace
4. Asigne los tablespaces default y temporal
5. Cree un usuario
6. Otorgue privilegios y roles al usuario

Creando un usuario nuevo: Autenticación de Servidor, establezca el password inicial

Sintaxis
Use el siguiente comando para crear un usuario nuevo:

```
CREATE USER user
IDENTIFIED {BY password | EXTERNALLY}
[ DEFAULT TABLESPACE tablespace ]
[ TEMPORARY TABLESPACE tablespace ]
[ QUOTA {integer [K | M] | UNLIMITED } ON tablespace
    [ QUOTA {integer [K | M] | UNLIMITED ] ON tablespace ]…]
[ PASSWORD EXPIRE ]
[ ACCOUNT {LOCK | UNLOCK }]
[ PROFILE {profile | DEFAULT }]
```

Donde:

user	Nombre del usuario
BY *password*	Especifica que el usuario es autenticado por la base de datos y necesita proporcionar *password* mientras se registra
EXTERNALLY	Especifica que el usuario es autenticado por el sistema operativo
DEFAULT/TEMPORARY TABLESPACE	Identifica el tablespace default/temporal para el usuario
QUOTA	Define el espacio máximo autorizado para los objetos propios del usuario en el tablespace (la cuota puede definirse como bytes, kilobytes y megabytes. El parámetro UNLIMITED es utilizado para especificar que los objetos del usuario pueden usar más espacio que el disponible en el tablespace. Por default, los usuarios no tienen cuota en cualquier tablespace).
PASSWORD EXPIRE	Forza a que el usuario resetee el password cuando se registra en la base de datos usando SQL*Plus (Esta opción solamente es válida si el usuario es autenticado por base de datos)
ACCOUNT LOCK/UNLOCK	Puede usarse para activar/desactivar la cuenta del usuario explícitamente (UNLOCK es el default)
PROFILE	Se utiliza para controlar el uso de recursos y para especificar el mecanismo de control de password para ser usado por el usuario

1) **Creando un usuario**

```
CREATE USER adminEstancia
IDENTIFIED BY estancia
DEFAULT tablespace data01
temporary tablespace temp
quota 15m on data01
 password expire;
```

Entrando a la base de datos con el nuevo usuario y asignándole nueva contraseña Fig. 2.9

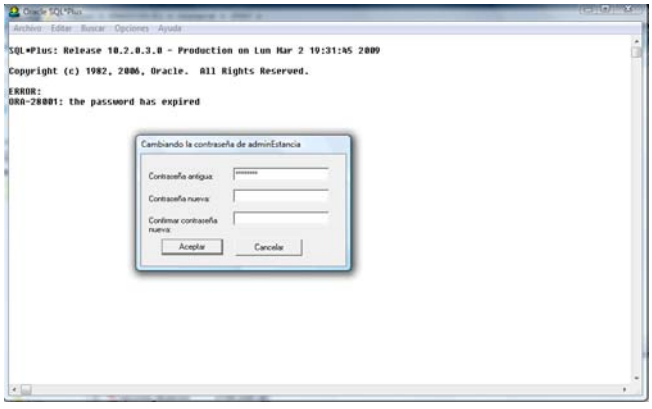


Fig. 2.9 Asignando nueva contraseña

Modificando y Eliminado usuarios de la Base de Datos

Controlando los candados y contraseñas de las cuentas

ALTER USER adminEstancia
IDENTIFIED BY Estancia
PASSWORD EXPIRE;

Es posible utilizar el comando ALTER USER para cambiar clave de acceso y bloquear cuentas. Algunas situaciones donde puede ser útiles son:

- Cambiar la clave de acceso cuando el usuario la ha olvidado.
- Desbloquear la cuenta de algún usuario cuando ha sido bloqueada por el sistema.
- Bloquear explícitamente una cuenta.
- Caducar manualmente claves de acceso; esta cláusula es útil cuando resetea contraseñas de usuarios.

Sintaxis:

Utilizar el siguiente comando en esas situaciones:

```
ALTER USER usuario
[ IDENTIFIED {BY PASSWORD | EXTERNALLY } ]
[ PASSWORD EXPIRE ]
[ ACCOUNT {LOCK | UNLOCK} ];
```

Cuando una cuenta ha sido bloqueada y el usuario intenta conectarse, aparece el siguiente mensaje:

```
ERROR:
ORA-28000: la cuenta ha sido bloqueada
Advertencia: no puedes conectarte a Oracle
```

Cambiar la cuota de usuario en el Tablespace

```
ALTER USER adminEstancia
QUOTA 0 ON data01
```

Puedes necesitar hacer una modificación en las cuotas del tablespace en las siguientes situaciones:

- Cuando las tablas propias de un usuario tienen un crecimiento desmedido.
- Cuando es mejorada una aplicación y requiere tablas o índices adicionales.
- Cuando los objetos son reorganizados y colocados en diferentes tablespaces.

Sintaxis:

Utilizar el siguiente comando para modificar las cuotas en el tablespace o para reasignar tablespaces:

```
ALTER USER  usuario
  [ DEFAULT TABLESPACE  tablespace ]
  [ TEMPORARY TABLESPACE tablespace ]
  [ QUOTA {integer [K | M] UNLIMITED } ON tablespace
  [ QUOTA {integer [K | M] UNLIMITED } ON tablespace ] .. ]
```

Eliminando un usuario

DROP USER adminEstancia;

Utilizar la cláusula CASCADE si el esquema contiene objetos

DROP USER adminEstancia CASCADE;

Sintaxis:

DROP USER usuario [CASCADE]

- La opción CASCADE elimina todos los objetos en el esquema antes de eliminarlo. Si el esquema contiene objetos, debe ser especificado.
- Un usuario que está actualmente conectado a Oracle Server no puede ser eliminado.

Perfiles

- Son grupos de recursos nombrados y límites de contraseñas.
- Son asignados a usuarios por el comando CREATE/ALTER USER
- Pueden ser habilitados o inhabilitados.
- Pueden relacionarse con el perfil DEFAULT
- Pueden limitar los recursos del sistema sobre la sesión o nivel de llamada

Un perfil es un grupo (set) nombrado de los siguientes recursos de sistema y límites de contraseñas:

- Tiempo de CPU
- Operaciones de entrada /salida
- Tiempo libre
- Tiempo de conexión
- Espacio de memoria
- Concurrencia de sesiones
- Caducidad y envejecimiento de la contraseña.
- Historia de la contraseña.
- Verificación de la complejidad de la contraseña.
- Bloqueo de cuentas.

Límites a los niveles de sesión y llamada.

Los límites de perfiles son ejecutados en el nivel de sesión, en el nivel de llamada o en ambos. Los límites del nivel de sesión son forzados para cada conexión. Cuando se excede un límite de nivel de sesión:

- Regresa un mensaje de error, ejemplo:

ORA-02391: exceeded simultaneous SESSIONS_PER_USER limit

- Oracle server desconecta al usuario.

Las limitaciones en el nivel de llamada son forzadas para cada llamada efectuada mientras se está (ejecutando una sentencia de SQL).
Cuando un límite del nivel de llamada es excedido:

- El procesamiento de las sentencias es detenido
- La sentencia es deshecha (rolled back).
- Todas las sentencias permanecen inalteradas.
- Las sesiones de usuario permanecen conectadas.

Uso de Perfiles.

- Restringe a los usuarios para realizar algunas operaciones que requieren un uso de recursos demandante.
- Asegura que los usuarios salgan de la base de datos cuando tengan que dejar su sesión libre por algún tiempo.
- Habilita grupos limitados de recursos para usuarios similares.
- Fácil asignación de recursos limitados a los usuarios.
- Maneja recursos utilizados en sistemas multiusuario grandes y complejos.
- Control de uso de contraseñas.

Controlando el uso de recursos

Administrando recursos con perfiles.

1. Creación de perfiles.
   Cree un perfil con el comando CREATE PROFILE para determinar los límites de recursos y contraseñas.

2. Asignación de perfiles a los usuarios.
   Asigne perfiles con los comandos CREATE USER o ALTER USER

3. Habilitar límites de recursos
   Establezca límites de recursos con el comando ALTER SYSTEM o mediante la edición del archivo de parámetros de inicialización (detenga y restablezca la instancia).

2) Creación de un perfil: límite de recurso

```
create profile adminEstancia limit
SESSIONS_PER_USER 2
CPU_PER_SESSION 10000
IDLE_TIME 30
CONNECT_TIME UNLIMITED;
```

3) Asignación de Perfiles a Usuarios

```
alter user adminEstancia
profile adminEstancia;
```

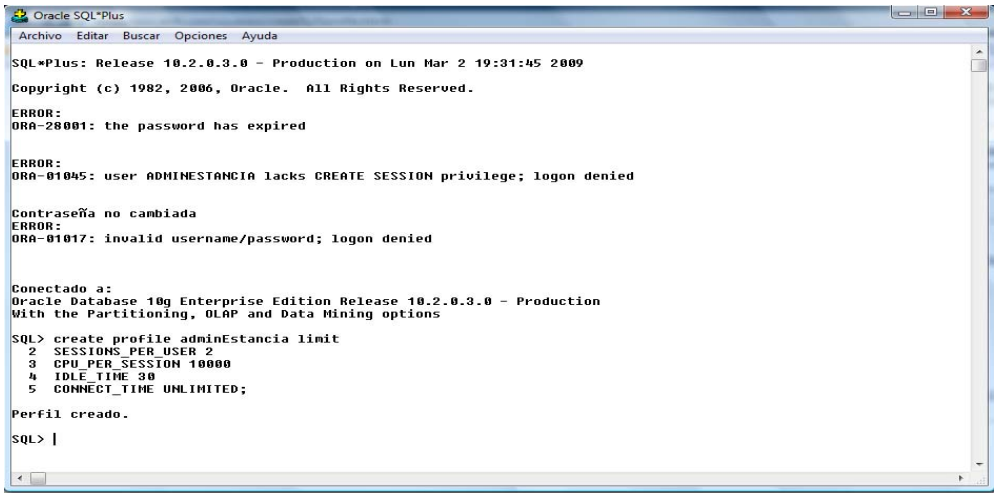


Fig. 2.10 Creación de perfil

Asignación de perfiles

Con el comando CREATE USER o ALTER USER, se asigna un perfil. A cada usuario le puede ser asignado sólo un perfil a la vez.

```
CREATE USER estancia IDENTIFIED BY estancia
DEFAULT TABLESPACE data01
TEMPORARY TABLESPACE temp
QUOTA unlimited ON data01
PROFILE perfil_estancia;
```

Modificando y Eliminando un Perfil

```
ALTER PROFILE default LIMIT
SESSIONS_PER_USER5
CPU_PER_CALL 3600
IDLE_TIME 30; (en minutos)
```

Modificando un perfil

```
ALTER_PROFILE perfil LIMIT
[SESSIONS_PER_USER          max_value]
[CPU_PER_SESSION            max_value]
[CPU_PER_CALL               max_value]
[CONNECT_TIME               max_value]
[IDLE_TIME                  max_value]
[LOGICAL_READS_PER_SESSIONS   max_value]
[LOGICAL_READS_PER_CALL       max_value]
[COMPOSITE_LIMIT            max_value]
[PRIVATE_SGA                max_bytes]
```

Eliminando un Perfil

```
DROP PROFILE perfil_estancia;
```

```
DROP PROFILE perfil_estancia
CASCADE;
```

Elimine un perfil utilizando el comando DROP PROFILE:

```
DROP PROFILE profile [CASCADE]
```

Donde:

- profile: es el nombre del perfil que va a ser quitado.
- CASCADE: revoca el perfil a los usuarios que estén asignados. (El Servidor de ORACLE asigna automáticamente el profile DEFAULT a estos usuarios. Especifique esta opción para eliminar un perfil que está asignado a los usuarios actualmente).

Privilegios

Hay 2 tipos de privilegios

- SYSTEM: Permite a los usuarios realizar acciones particulares en la base de datos.
- OBJECT: Permite a los usuarios accesar y manipular un objeto específico.

Privilegios de Sistema

Cada privilegio de sistema permite a un usuario realizar una operación particular en la base de datos o una clase de operaciones en la base de datos. Estas operaciones incluyen crear, eliminar, y alterar tablas vistas, segmentos de rollback y procedimientos.

- Hay alrededor de 80 Privilegios de Sistema y el número continúa creciendo.
- El comando ANY en los privilegios significa que los usuarios tienen el privilegio en cualquier esquema. por ejemplo; CREATE ANY TABLE ó DROP ANY TABLE.
- El comando GRANT adiciona un privilegio a un usuario o a un grupo de usuarios.
- El comando REVOKE borra los privilegios.

Categoría	Ejemplos
INDEX	CREATE ANY INDEX ALTER ANY INDEX DROP ANY INDEX
TABLE	CREATE TABLE CREATE ANY TABLE ALTER ANY TABLE DROP ANY TABLE SELECT ANY TABLE UPDATE ANY TABLE DELETE ANY TABLE
SESSION	CREATE SESSION ALTER SESSION RESTRICTED SESSION
TABLESPACE	CREATE TABLESPACE ALTER TABLESPACE DROP TABLESPACE UNLIMITED TABLESPACE

4) Otorgando Privilegios de Sistema

grant create session, create table
to adminEstancia;

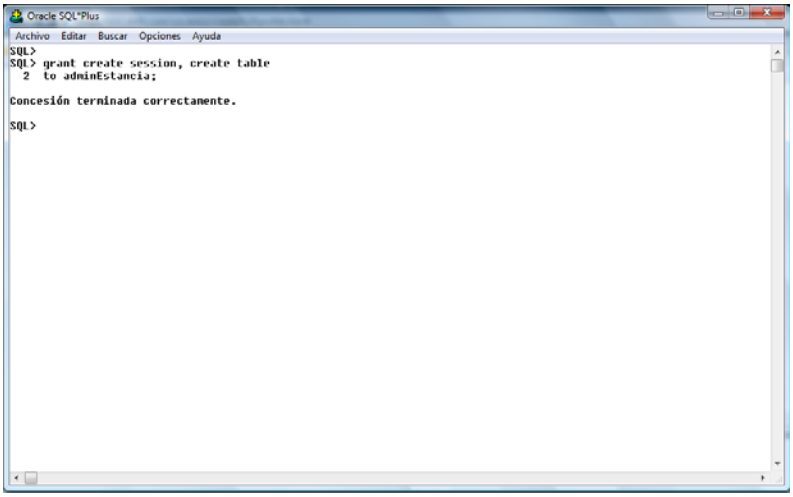


Fig. 2.11 Otorgando privilegios de sistema

Sintaxis:

Use el siguiente comando para otorgar privilegios de Sistema

```
GRANT {system_priv | role}
    [,{system_priv | role} ] . . .
  TO        {user | role | PUBLIC}
    [,{user | role | PUBLIC} ] . . .
    [WITH ADMIN OPTION]
```

Donde:

system_priv	Especifica el privilegio de sistema que se está otorgando.
role	Especifica el nombre del role al que se le otorgo
PUBLIC	Otorga un privilegio de sistema a todos los usuarios
WITH ADMIN OPTION	Habilita el otorgar un privilegio adicional al privilegio o role de otros usuarios o roles (extender, heredar)

REVOCANDO PRIVILEGIOS DEL SISTEMA

REVOKE CREATE TABLE FROM estancia;

REVOKE CREATE SESSION FROM estancia;

Sintaxis:

Use el siguiente comando para revocar los privilegios de sistema:

```
REVOKE   {system_priv | role}
      [,  {system_priv | role} ] . . .
  FROM  {user | role | PUBLIC}
      [,   {user | role | PUBLIC}  ] . . .
```

PRIVILEGIOS DE OBJETO

Cada privilegio de objeto permite a un usuario realizar una acción particular sobre un objeto específico, como en una tabla, una vista, una secuencia, un procedimiento, una función o un paquete.

Priv. Objeto	Tabla	Vista	Secuencia	Procedimiento
ALTER	✓		✓	
DELETE	✓	✓		
EXECUTE				✓
INDEX	✓			
INSERT	✓	✓		
REFERENCES	✓			
SELECT	✓	✓	✓	
UPDATE	✓	✓		

Cada privilegio de objeto que está otorgado, autoriza el otorgamiento para realizar algunas operaciones sobre los objetos.

- Los privilegios pueden estar clasificados como sigue:
  - Privilegios que permiten operaciones amplias en un sistema: por ejemplo; CREATE SESSION, CREATE TABLESPACE.
  - Privilegios que permiten el manejo de objetos en un esquema propietario de un usuario, por ejemplo; CREATE TABLE.

Pueden ser controlados con los comandos GRANT y REVOKE, los cuales adicionan o revocan privilegios del sistema a un usuario o un rol

OTOGANDO PRIVILEGIOS DE OBJETO

Transferencia de los privilegios de un objeto

GRANT EXECUTE ON dbms_pipe TO public;

GRANT UPDATE (ENAME, SAL) on EMP
TO estancia WITH GRANT OPTION;

Sintaxis

Use el siguiente comando para otorgar un privilegio de objeto:

```
GRANT   {   object_priv  [(Column_list)]   ] . . .
      | ALL [PRIVILEGES]}
ON      [schema. ] object
TO      {user | role | PUBLIC}
   [,  {user | role | PUBLIC}   ] . . .
(WITH  GRANT OPTION]
```

Donde:

object_priv	Especifica los privilegios de objeto que son otorgados
Column_list	Específica una tabla o una vista Y columnas (Estas solo pueden
REFERENCES,	estar especificadas cuando se otorga el privilegio INSERT, o UPDATE).
ALL	Otorga todos los privilegios para el objeto que ha sido otorgados con WITH GRANT OPTION
ON object	Identifica el objeto sobre el cual deben estar otorgados los privilegios
WITH GRANT OPTION	Habilita el otorgamiento para la transferencia de privilegios de los objetos, para otros usuarios o roles.

REVOCANDO PRIVILEGIOS DE OBJETO

REVOKE execute ON dbms_pipe
FROM scott;
Sintaxis:

Use el siguiente comando para revocar un privilegio de objeto:

```
REVOKE  {     object_priv
          [,   object_priv ] . . .
        | ALL  [PRIVILEGES]  }
ON          [schema. ] object
FROM        {user | role | PUBLIC}
          [,  {user | role | PUBLIC}  ] . . .
          [ CASCADA CONSTRAINTS ]
```

donde:

object_priv	Especifica el privilegio de objeto a ser revocado
ALL	Revoca todos los privilegios de objeto que son otorgados al usuario
ON	Identifica el objeto en el cual se revocaron sus privilegios
FROM	Identifica usuarios o roles desde los cuales han sido revocados los privilegios
CASCADE CONSTRAINTS	Elimina cualquier restricción de integridad referencial.

Roles

Los Roles son grupos nombrados para relacionar privilegios que son otorgados a los usuarios o a otros Roles. Son designados para el caso de la administración de privilegios en la base de datos.

Características de los Roles

- Otorga a y revoca desde los usuarios con los mismos comandos usados para otorgar y revocar los privilegios del sistema.
- Puede otorgar a cualquier usuario o roles, excepto para sí mismo (ni siquiera indirectamente)
- Pueden consistir de los privilegios de objeto y sistema
- Pueden estar habilitado o inhabilitado para cada usuario otorgado el role.
- Puede requerir de una contraseña para habilitarse.
- Cada nombre de role debe ser único entre los nombres de roles y usuarios existentes.
- No son de nadie, ni están en cualquier esquema.
- Tienen sus descripciones almacenadas en el diccionario de datos.

Creación y modificación de Roles

```
CREATE ROLE sales_clerk;

CREATE ROLE hr_clerk
  IDENTIFIED BY bonus;

  CREATE ROLE hr_manager
  IDENTIFIED EXTERNALLY;

CREATE ROLE estancia
 IDENTIFIED By estancia;
```

Sintaxis:

Use el siguiente comando para crear un Role

```
CREATE ROLE role [NOT_IDENTIFIED | IDENTIFIED ]
{By password | EXTERNALLY } ]
```

Donde:

role	el nombre del Role
NOT IDENTIFIED	Indica que no es requerida la verificación, cuando se habilita el Role
IDENTIFIED	Indica que se requiere la verificación cuando se habilita el Role.
By password	Proporciona el password del usuario cuando se habilita el role.
EXTERNALLY	Indica que un usuario debe estar autorizado por un servicio externo (como el sistema operativo o un tercer servicio) antes de habilitar el role.

Usando Roles Predefinidos

Nombre del Role	Descripción
CONNECT RESOURCE	Estos dos roles son proporcionados para la compatibilidad.
DBA	Todos los privilegios de sistema WITH ADMIN OPTION
EXP_FULL_DATABASE	Privilegios de exportación de DB
IMP_FULL_DATABASE	Privilegios para importación a DB
DELETE_CATALOG_ROLE	Privilegios DELETE sobre tablas DD.
EXECUTE_CATALOG_ROLE	Privilegios EXECUTE sobre paquetes DD.
SELECT_CATALOG_ROLE	privilegios SELECT sobre tablas DD

Roles del sistema

Nombre del rol(entorno de desarrollo y usuarios finales)	Privilegios otorgados al rol
CONNECT	Alter session, create cluster, create database link, create sequence, create session, create synonym, create table, create view
RESOURCE	Create cluster, create procedure, create sequence, create table, create trigger, UNLIMITED TABLESPACE (Debe restringirse a los entornos de desarrollo y pruebas, es decir a sus tablespaces)
DBA	Todos los privilegios de sistema WITH ADMIN OPTION (concede privilegios de sistema a cualquier otro usuario), UNLIMITED TABLESPACE
Administración de la base de datos	
EXP_FULL_DATABASE	Select any table, backup any table
WM_ADMIN_ROLE	Todos los privilegios de administrador de área de trabajo con GRANT OPTION
IMP_FULL_DATABASE	BECOME USER
DELETE_CATALOG_ROLE	DELETE en todos los paquetes del diccionario
EXECUTE_CATALOG_ROLE	EXECUTE en todos los paquetes del diccionario
SELECT_CATALOG_ROLE	SELECT en todas las tablas y vistas de catalogo
CREATE TYPE	CREATE TYPE, EXECUTE, EXECUTE ANY TYPE, ADMIN OPTION, GRANT OPTION
RECOVERY_CATALOG_OWNER	DROP ROLE, CREATE ROLE, CREATE TRIGGER, CREATE PROCEDURE
OLAP_DBA	ALTER ANY DIMENSION, ALTER ANY TABLE, ANALYZE ANY, CREATE ANY DIMENSION, CREATE ANY INDEX, CREATE ANY TABLE, CREATE ANY VIEW, DROP ANY DIMENSION, DROP ANY TABLE, DROP ANY VIEW, LOCK ANY TABLE, SELECT ANY DICTIONARY, SELECT ANY TABLE
HS_ADMIN_ROLE	HS_EXTERNAL, OBJECT, HS_EXTERNAL_USER
WKADMIN	CREATE ANY DIRECTORY, CREATE CLUSTER, CREATE PROCEDURE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE, DROP ANY DIRECTORY
WKUSER	CREATE ANY DIRECTORY, CREATE CLUSTER, CREATE PROCEDURE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE, DROP ANY DIRECTORY

Modificación de Roles.

```
ALTER ROLE estancia
  IDENTIFIED BY adminestancia;

ALTER ROLE estancia
  IDENTIFIED EXTERNALLY;
```

ALTER ROLE estancia
NOT IDENTIFIED;

Un role sólo puede modificarse para cambiar su método de la autenticación.

Sintaxis.

Use el siguiente comando para modificar el role:

```
ALTER ROLE role  {NO IDENTIFICATION | IDENTIFIED
 {BY  password  EXTERNALLY}};
```

Donde:

role	Es el nombre del role.
NOT IDENTIFIED	Indica que no se requiere verificación Cuando se habilita el role.
IDENTIFIED	Indica que se requiere comprobación al habilitar la fila.
BY password	Proporciona la contraseña cuando se habilita el role.
EXTERNALLY	Indica que un usuario debe ser autorizado por un servicio externo (como el sistema operativo o un tercer servicio) antes de habilitar el role.

Asignación de Roles

GRANT sales_clerk to scott;

GRANT hr_clerk
TO hr_manager;

GRANT hr_manager To scott,
WITH ADMIN OPTION;

Sintaxis

Para concederle un ROLE a un usuario, use el mismo comando de sintaxis que usó para concederle a un usuario un privilegio del sistema:

```
GRANT ROLE [, role]...
  TO { role | user | PUBLIC}
     [, {role | user | PUBLIC}]...
    [WITH ADMIN OPTION]
```

Donde:

role	Es un role a ser concedido o un role que recibe el role concedido
user	Es un usuario receptor del role
role	Es un role recibiendo el role.
PUBLIC	Concede el role a todos los usuarios.
WITH ADMIN OPTION	Role a otros usuarios o a roles (si usted concede un role o función con esta opción, el otorgante puede otorgar y puede revocar el role de otros usuarios y puede alterar o eliminar todos los roles).

El usuario que crea un role se le asigna implícitamente el role WITH ADMIN OPTION. Un usuario que no se le ha concedido un role WITH ADMIN OPTION, requiere de privilegios de sistema GRANT ANY ROLE para conceder y revocar roles a unos o a otros.

5) Asignación de Roles

GRANT ROLE DBA TO ADMINESTANCIA,
WITH ADMIN OPTION;

2.5 Desempeño y Afinación (Tunning)

La Sintonía es un *diseño cuidadoso de sistemas y aplicaciones* y la mayoría de las ganancias de desempeño (performance) son realizadas sintonizando la aplicación.

Tendrá mucho menos probabilidad de presentar problemas de desempeño si:

- La base de datos Oracle8 fue diseñada exitosamente.
- Sus desarrolladores de aplicaciones hacen programas SQL eficientes.

La sintonía es un proceso iterativo, no es una actividad que se hace una sola vez y luego se olvida.

Cuando sintoniza una base de datos en ambiente Oracle, el DBA debe establecer metas de sintonía medibles:

- Tiempo de respuesta.
- Disponibilidad de la base de datos.
- Utilización de memoria.

Pasos para la sintonía

1. Sintonizar el diseño
2. Sintonizar la aplicación
3. Sintonizar la memoria
4. Sintonizar las entradas y salidas.
5. Sintonizar la contención.
6. Sintonizar el sistema operativo.

Aspectos claves de la sintonía:

- *Crear un buen diseño inicial.*
- *Establecer metas cuantificables.*
- *Realizar sintonía de aplicación.*

Consideraciones de sintonía para diferentes aplicaciones:

El diseño de datos y la estructura lógica de la base de datos deben ser para el tipo de aplicación:

- Procesamiento de transacciones en línea (OLTP)
- Sistemas de apoyo de decisiones (DSS)
- Aplicaciones multi-propósito (combinación de las 2 anteriores)

Fase de diseño de datos: Necesita estructurar la información para conocer mejor las metas de desempeño. El proceso de diseño de la base de datos se remite e una etapa de normalización, sin embargo; puede necesitar desnormalizarla por razones de desempeño.

Estructura Lógica de la Base de Datos

Esto primeramente concierne a:

- Uso correcto de diferentes tipos de índices:
  - Árbol B
  - Bitmap
  - Inverso

- Uso apropiado de secuencias, clusters, tablas organizadas por índice.
- La necesidad de colección de histogramas para el CBO (Cost Based Optimizar).
- El uso de consultas paralelas.
- El uso opcional de partición de datos.

A continuación se mostrará una tabla comparativa de las diferentes aplicaciones:

Tipo de USO		
Procesamiento de Transacciones en Línea (OLTP)	Sistemas de Apoyo de Decisiones (DSS)	**Aplicaciones Multipropósito**
Son sistemas intensivos de Insert/Update, (OLTP) Contiene grandes volúmenes de información que: Crecen continuamente. Se accesa concurrentemente por cientos de usuarios. Alta disponibilidad (7 días/24 hrs.) Tiempo de recuperación reducido. Usa constraints en lugar de código de aplicación, asegúrese que esté compartidos procedimientos y funciones para bajar el tiempo de parseo. Acceso inmediato a pequeñas cantidades de información (indexada, desordenada)	Realizan consultas de grandes cantidades de información. Hacen un fuerte uso de barridos de tablas completas (Full scan table). Poca concurrencia o nula Disponibilidad variable Tiempo de respuesta rápido. Las pruebas se realizan sobre cantidades de información reales. Información fuertemente empaquetada dentro de grandes bloques.	Se trata de un sistema híbrido, combinación de OLTP y DSS. Cuentan con varias configuraciones. El conflicto del desempeño de la metas puede resolverse copiando información reunida de la base de datos OLTP en una Bases de Datos DSS. Hacer reportes y consolidados de ejecuciones después de horas que son pico.

Segmentos Rollback		
Necesita más segmentos rollback pequeños y suficientes para prevenir contención. Necesita configurar MINEXTENTS al menos 10 para Bases de Datos pequeñas y 20 para Bases de Datos grandes, se recomienda commit constantemente.	Necesita pocos segmentos rollback grandes.	Necesita configurar los diferentes segmentos rollback para diferentes usos. Si puede tener más espacio, puede querer usar • Muchos segmentos rollback pequeños durante el día que permanezcan fuera de línea durante la noche. • Unos cuantos grandes en la noche que permanezcan fuera de línea durante el día. Si no puede tener esa cantidad de espacio, antes de bajar la base de datos en la noche, programe un script que reemplace los segmentos rollback diurnos por los nocturnos y viceversa, al levantar la base de datos.
Paralelismo		
Podría usar consultas paralelas. Los índices de llave inversa evitan que los bloques de árbol B se dividan. Se requiere reconstruir regularmente los índices, utilice parallel.	Su prioridad es una ruta de acceso óptima en el plan de ejecución Emplea consultas paralelas para operaciones grandes. Usar consultas paralelizadas para trabajar simultáneamente una instrucción SQL la operación puede dividirse muchos CPU's.	*Consulta paralela para operaciones grandes.* Durante el día, el PARALLEL_MIN_SERVERS y PARALLEL_MAX_SERVERS pueden ponerse en 0 para prevenir paralelización. Soporta consultas paralelas bien configuradas

Particiones	
OLTP	**DSS**
Tablas con gran volumen de información.	Para grandes cantidades de datos cuyas consultas concentran accesos en registros que fueron generados recientemente, puede usar también *Tablas IOT*.

Índices

Los índices los usa Oracle Server con estadísticas del Cost Based Optimizer que estiman que la instrucción será ejecutada más rápido ó Rule Based Optimizer.

Métodos de Acceso de Información	
OLTP	**DSS**
Indexado La Estrategia de indexado debe ser acorde a las necesidades reales de la aplicación, el indexado de *árbol B* es preferible. Necesita más índices Utiliza también Índices de llave inversa, estos evitan que los bloques de árbol B se dividan y más rápida las aplicaciones parellel, necesitan reconstruir regularmente lo índices. Usan Analyze	**Indexado** Mantenerlos sólo para algunas tablas que se accesan selectivamente. Generar regularmente histogramas para aquellos con información distribuida sin uniformidad. Elegir índices bitmap para consultas en columnas con pocos valores distintos.

Índices de Árbol B

Son usados por tablas que son consultadas frecuentemente por menos del 15% de los registros.

¿Como resolver la degradación del desempeño?

-Regularmente reconstruyen el índice;

```
ALTER INDEX i_name REBUILD;
```

-Dividiendo la tabla indexada y creando índices particionados.

Ejemplo:

```
CREATE UNIQUE INDEX PK_INFANTES ON INFANTES
(
    curp_infante            ASC
)
        PCTFREE 47
        INITRANS 2
        MAXTRANS 17
        TABLESPACE TS_INDICES
        STORAGE  (
                INITIAL 815
                NEXT 100
                MINEXTENTS 1
                MAXEXTENTS 2
                PCTINCREASE 0
        )
;
```

Índices Bitmap

Son utilizados para columnas de baja cardinalidad, él mejor para sistemas de sólo lectura

SORT_AREA_SIZE debe configurarse apropiadamente.

Parámetros de inicialización:

CREATE_BITMAP_AREA_SIZE Cantidad de memoria para la creación de bitmap

BITMAP_MERGE_AREA_SIZE Cantidad de memoria utilizada para unir bitmaps recuperados.

Ejemplo:

```
CREATE BITMAP INDEX infantes_sexo_idx
   ON infantes(sexo)
   PCTFREE 47
   STORAGE(INITIAL 815 NEXT 100
   PCTINCREASE 0  MINEXTENTS 1    MAXEXTENTS 2)
   TABLESPACE TS_INDICES;
```

Comparando Índices de Árbol B y Bitmap	
Árbol B	**Bitmap**
Apropiado para columnas de alta cardinalidad	Apropiado para columnas de baja cardinalidad
Actualiza sobre llaves relativamente baratas	Actualiza columnas llave muy caras
Ineficiente para consultas usando predicados OR	Eficiente para consultas usando predicados OR
Candados a nivel registro	Candados a nivel segmento bitmap
Más almacenamiento	Menos almacenamiento
Útil para OLTP	Útil para OLAP

Índices de Llave Inversa (REVERSE)

El índice de llave inversa invierte los bytes de cada valor de columna llave, manteniendo el orden de columnas en caso de una llave compuesta.

Cuando No crear índices de llave inversa:

Cuando la llave se incrementa constantemente (como números de secuencia, como facturas o números de pedido o de empleados).

Desventaja: Cuando las sentencias de la aplicación especifican rangos, el plan de ejecución (explain plan) produce un barrido de tabla completo (full table scan).

Ejemplo:

```
CREATE UNIQUE INDEX infantes_curp_idx
   ON infantes(curp_infante) REVERSE
   PCTFREE 41
   STORAGE (INITIAL 815 NEXT 100
   PCTINCREASE 0  MINEXTENTS 1      MAXEXTENTS 2)
   TABLESPACE TS_INDICES;
```

IOT y Clusters:

Las tablas organizadas por índices y clusters contienen información que puede recuperar más rápido que si estuviera almacenada en tablas regulares.

IOT

Son apropiadas para accesos de información frecuente a través de la llave primaria, necesitan menos requerimiento de almacenamiento. Distribución, replicación y particionamiento no son soportados.

Cláusula PCTTHRESHOLD: porcentaje de espacio reservado en el bloque indexado

Cláusula INCLUDING: especifica una columna a la cual dividir un registro de una tabla índice organizado dentro de porciones de desborde (overflow) e índice

Cláusula OVERFLOW y segmentos: especifica que los registros excediendo el umbral están ubicados en el tablespace, almacenamiento y parámetros de utilización de bloque.

Clusters

Beneficios del desempeño. La I/O a disco es reducido y el tiempo de acceso mejora para joins de tablas, usa menos almacenamiento.

Consideración del desempeño. El barrido de la tabla completa (full table scan) generalmente es más lento en tablas clusters.

¿Cuándo no usar clusters? Se ejecuta frecuentemente un barrido completo (full table scan) en una de las tablas clusters: Si la información de todas las tablas con el mismo valor de llave cluster excede más de uno o dos bloques.

¿Cuándo no estar Clusters Hash? Si la tabla está creciendo constantemente, si su aplicación realiza frecuentemente barridos de tablas completos.

Clusters	
OLTP	**DSS**
Utiliza Clusters indexados. Usando clusters hash podría accesar, mucho más rápido en consultas. Pero no en una tabla con inserts constantes. Si está creciendo una tabla estime correctamente el valor HASHKEYS con un número grande así la probabilidad de un colisión disminuye.	Se deben considerar ambos tipos de clusters y especialmente los **clusters hash** para el mejor desempeño en el acceso, excluyendo las tablas que estén creciendo regularmente durante las cargas voluminosas, excepto si tiene la posibilidad de volver a crear el cluster.

Histogramas

Son estadísticas reunidas por el comando ANALYZE, almacenadas en las vistas del diccionario y utilizadas por el cost Based Optimizer para sintonizar la ejecución de instrucciones de SQL.

SQL> ANALYZE INDEX i_name COMPUTE STATISTICS

¿Cuándo usar Histogramas? En columnas donde se usa frecuentemente cláusulas WHERE, en columnas indexables.

¿Cuándo no usar Histogramas? En columnas utilizadas en predicado con variables bind., en columnas de información distribuida uniformemente, en columnas únicas utilizadas en predicados equitativos.

¿Cómo generar Histogramas Estadísticos?

SQL> ANALYZE TABLE table_name COMPUTE STATISTICS FOR ALL INDEXED COLUMNS;

Histogramas	
OLTP	**DSS**
Menos frecuente	Más frecuente. El comando ANALYZE sirve para recopilar y actualizar el catalogo de Oracle con datos estadísticos.

SINTONIA SQL

- La sintonía de aplicación es la parte más importante de la sintonía.
- Los administradores de las Bases de Datos:
  - Pueden no estar involucrados directamente en la sintonía de aplicación.
  - Deben estar familiarizados con el impacto que las instrucciones SQL mal escritas pueden tener hacia el desempeño de la base de datos.

El diseño y sintonía de aplicación proporcionarán los mayores beneficios al desempeño de la base de datos.

Como DBA necesita estar enterado del impacto que tienen las instrucciones SQL mal escritas en el ambiente de la base de datos. Deberá ser capaz de proporcionar ayuda con la sintonía de la aplicación e identificar fácilmente instrucciones SQL poco eficientes.

Modos Optimizadores

En Oracle se pueden elegir dos modos optimizadores:

<table>
<tr><th colspan="2">Optimizadores</th></tr>
<tr><th>Basados en reglas</th><th>Basado en costos</th></tr>
<tr><td>Usa un sistema de categorías
En este modo, el proceso de servidor elige su ruta de acceso hacia la información examinando la consulta. Este optimizador tiene un grupo completo de reglas para categorizar las rutas de acceso.</td><td>Elige la ruta de menor costo
En este modo, el optimizador examina cada instrucción e identifica todas las rutas posibles de acceso a la información. Luego calcula el costo de los recursos de cada ruta de acceso y elige el menos caro. El costo está basado principalmente en el número de lecturas lógicas.</td></tr>
<tr><td>Manejado por sintaxis

Usa la sintaxis de instrucción para determinar qué plan de ejecución será utilizado.

EXPLAIN PLAN y HINTS

Usar el comando EXPLAIN PLAN para sintonizar las instrucciones SQL y hints para controlar las rutas de acceso.</td><td>Manejado por estadísticas

Genera estadísticas por los objetos involucrados en la instrucción SQL para determinar el plan de ejecución más efectivo, sólo si algún objeto en la instrucción SQL ha tenido estadísticas generadas por él.

SQL Trace para poder obtener información de performance de sentencias SQL en forma individual. Por cada sentencia SQL que se ejecuta, SQL Trace brinda la información estadística.

DSS: Realiza más barridos completos de tablas

(full table scan), tiene minuciosa sintonía de consultas y menos atención al tiempo de parseo.</td></tr>
<tr><td colspan="2">Configurando el Modo Optimizador.

El modo optimizador puede configurarse a:

• Nivel de instancia, usando el parámetro OPTIMIZER_MODE. El DBA es responsable de configurarlo.
- CHOOSE; significa que el optimizador usa el modo basado en costos si están disponibles las estadísticas para las tablas.
- RULE; basado en reglas (por ejemplo usando el índice)
- FIRST_ROWS; minimiza el tiempo de respuesta inmediato
- ALL_ROWS; minimiza el tiempo de respuesta total.
• Nivel de sesión, configurando el OPTIMIZER_GOAL.
• Nivel instrucción, usando Optimizadores hints (sugerencias); dentro de una instrucción, SELECT/* + FIRST_ROWS*/*FROM Scott.empleados. Hits son RULE,FIRST_ROWS y ALL_ROWS.</td></tr>
</table>

Consultas Estrella

Las consultas estrella se usan en aplicaciones data warehouse y alrededor del centro, por lo que se les conoce como un esquema "estrella". *El optimizador basado en costos reconoce consultas estrella y construye una ruta de ejecución de consulta*.

Una consulta estrella consiste de una o más tablas fact grandes que contienen la información básica en el data warehouse y un gran número de pequeñas tablas dimensión (o tablas "lookup").

La tabla fact normalmente tiene un *índice concatenado* en las columnas llave para facilitar este tipo de join.

El parámetro **STAR_TRANSFORMATION_ENABLED** especifica si se aplica una transformación de consulta basada en costos a consultas estrella o no. El valor (default es TRUE).

Este parámetro puede configurarse dinámicamente usando el comando ALTER SESSION.

Joins Hash

- Usados para aplicaciones data warehouse

Los joins hash proporcionan un mecanismo eficiente para unir dos tablas donde una de ellas puede ser significativamente mayor que la otra.

Tres parámetros que son importantes para los join hash son:

- HASH_AREA_SIZE: Especifica la *cantidad máxima de memoria a ser utilizada en los join hash*.
- HASH_JOIN_ENABLED: *Especifica si el optimizador debe considerar usar un método join hash o no* (El valor default es TRUE. Este parámetro también *puede configurarse dinámicamente* usando el comando ALTER SESSION).
- HASH_MULTIBLOCK_IO_COUNT: *Especifica cuántos bloques secuenciales lee y escribe un join hash en una I/O* (Este parámetro *también puede configurarse dinámicamente* usando los comandos ALTER SYSTEM o ALTER SESSION).

Herramientas de Diagnóstico

- EXPLAIN PLAN (solo plan de ejecución) Esta se ejecuta dentro de una sesión para una instrucción SQL y los resultados saldrán hacia el PLAN_TABLE. Necesita crear una tabla llamada PLAN_TABLE (columnas útiles para muchos propósitos son OPERATION, OPTIONS y OBJECT_NAME.)

El plan de ejecución se lee de abajo para arriba, ayudan a determinar cuál será el beneficio al crear y usar índices para consultas DML.

- SQLTRACE (monitorea instrucciones que se están ejecutando), información detallada considerando la ejecución de una instrucción SQL.
- TKPROF (plan de monitorea instrucciones que se están ejecutando), utilería de sistema operativo que toma la salida desde una sesión SQLTRACE y formatea dentro de un formato legible.
- AUTOTRACE (modo de sqlplus, monitorea lo que voy haciendo en línea)

Diagnostico de desempeño de Instrucciones SQL

El primer paso en el diagnostico es asegurarse que los parámetros de inicialización son los adecuados a nivel instancia o algunos de nivel sistema o sesión. Debe invocarse a SQL Trace a nivel de instancia o sesión, es mejor desempeño total. Ejecute la aplicación o instrucción SQL que desee diagnosticar. Cancele el SQL Trace, Use TKPROF para dar formato al archivo trace si no será muy difícil interpretar los resultados, Use la salida del TKPROF para diagnosticar el desempeño de la instrucción SQL.

Parámetros de Inicialización importantes

Dos parámetros en el archivo init.ora controlan el tamaño y destino del archivo de salida desde la facilidad de SQL trace.

max_dump_file_size = n (default 500, en K o M si no se especifica es block del sistema operativo)

user_dump_dest = directory

Debe configurar un tercer parámetro para obtener la información del timing:

timed_statistics = TRUE

Las estadísticas del timing tienen una resolución de centésimas de segundo.

Activando y desactivando el trace

- Nivel instancia:
  SQL_TRACE = [TRUE | FALSE]

Nivel sesión usando uno de:

```
ALTER SESSION SET sql_trace = [TRUE | FALSE];

EXECUTE sys.dbms_session.set_sql_trace ( [TRUE | FALSE] );

EXECUTE sys.dbms_system.set_sql_trace_in_session

    (session_id, serial_id, [TRUE | FALSE] );
```

SQL trace puede activarse o desactivarse usando diferentes métodos a nivel instancia o a nivel sesión.

Dar formato al Archivo de Rastreo: Tkprof ora_19999.trc myfile.txt explain=scott/tiger

El archivo de rastreo se crea en el directorio especificado por el parámetro USER_DUMP_DEST.

- Estadísticas trace:
- Count: Veces en que fue ejecutado el procedimiento
- CPU: Segundos para el proceso
- Elapsed: Segundos para ejecución
- Disco: Lecturas físicas

- Consultas: Lecturas lógicas para consistencia de lectura
- Actual: Lecturas lógicas en modo actual
- I/O lógicas: Consulta + Actual
- Registros: Registros procesados por el fetch o execute

Storage e Init.ora		
OLTP	DSS)	**Aplicaciones Multipropósito**
Asigna espacio explícitamente a las tablas, clusters e índices y Revisa regularmente el crecimiento para planear la creación de extensiones. PCTFREE acorde a la actividad de actualización esperada. Sin asignación de espacio durante las horas pico	Para inserciones y actualizaciones voluminosas, debe configurar los parámetros init.ora de ordenamiento SORT_AREA_SIZE, BITMAP_MERGE_AREA_SIZE, CREATE_BITMAP_AREA_SIZE. Use tablas de índice organizado para un acceso más rápido basado en llave para consultas. Sintonizar la librería caché es mucho menos importante para DSS. PCTFREE puede ponerse a 0, si vuelve a crear la tabla actualizada	Necesita dos archivos de parámetros separados para trabajos de día y de noche, cancelando y levantando el inicio y final del día de trabajo. ***Uso de Memoria*** Los siguientes parámetros tendrán valores mayores para el día (esto es, OLTP): • SHARED_POOL_SIZE • DB_BLOCK_BUFFERS • SORT_AREA_SIZE Los barridos de tabla completa (full table scan) de tablas grandes puede restringirse: usar diferentes perfiles en el día para limitar el LOGICAL_READS_PER_CALL o CPU_PER_CALL y asignar estos perfiles a los usuarios

Vistas de Desempeño dinámico

Puede consultar vistas de desempeño dinámico para derivar información sobre la sesión actual o la instancia.

- V$MYSTAT: Contiene estadísticas para la sesión actual.
- V$PARAMETER: Contiene los valores para los parámetros que pertenecen a la sesión actual que está consultando la vista.
- V$SYSTEM_PARAMETER: Contiene los valores para los parámetros que han sido configurados en el nivel de instancia.

2.6 Base de Datos en aplicaciones basadas en internet

Como se mencionó anteriormente desarrollamos una aplicación de base de datos, la cual se desarrollo en un 10% y se muestra a continuación.

En la Fig. 2.12 su muestra la pagina de bienvenida en donde se explican los objetivos de nuestra tesis. Esta conformada por las siguientes pestañas:

- Inicio: Página de inicio.
- Alta: Registro de datos de un infante.
- Actualización: Actualización de datos de un infante.
- Búsqueda: Búsqueda de un infante.
- Sitios de interés: Ligas que frecuentemente se utilizan.

El buscador de la página está ligado a la página de google.

Fig. 2.12 Página Principal

Aquí un ejemplo de búsqueda.

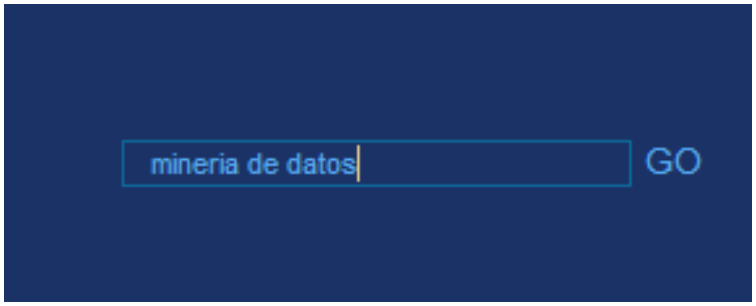


Fig. 2.12 Búsqueda

Resultado de la búsqueda.

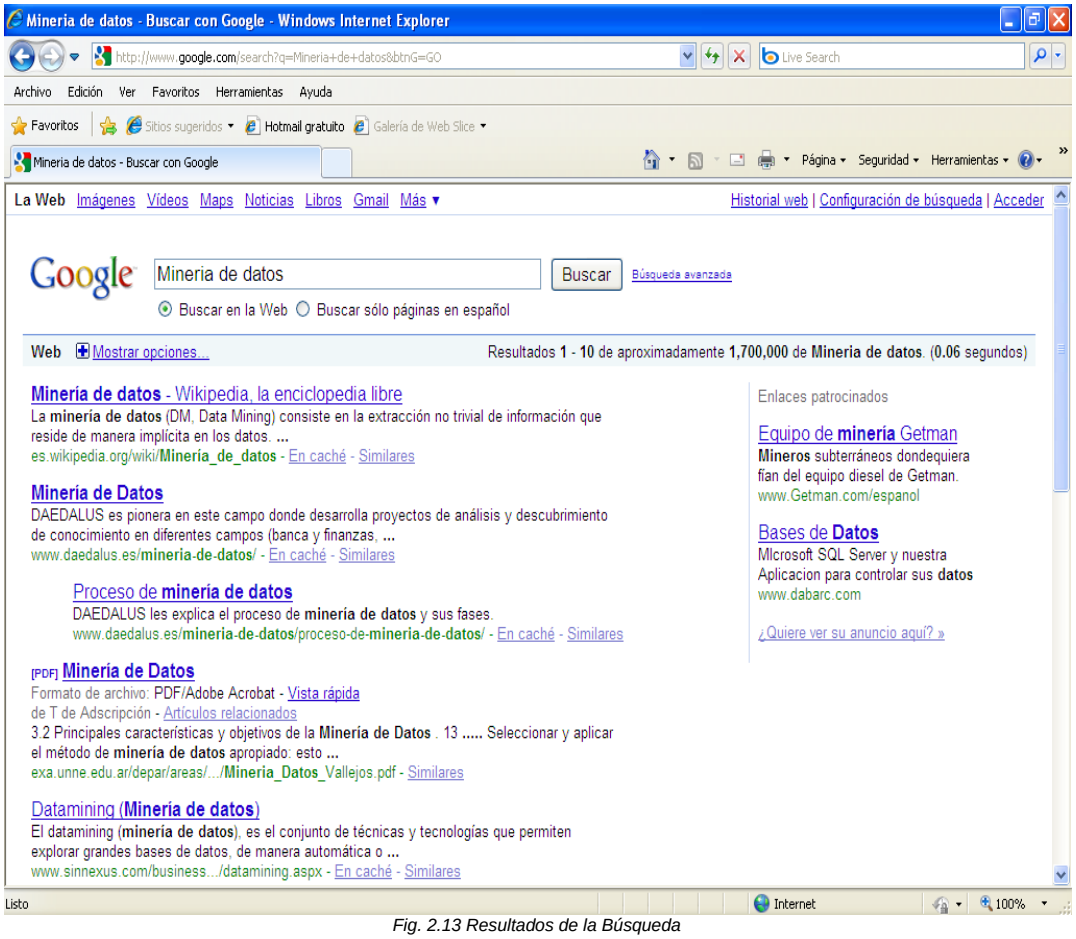


Fig. 2.13 Resultados de la Búsqueda

Al seleccionar “Alta” se desplegará la siguiente página, en donde se ingresarán los datos del infante (Correspondiente a la función 2.1 del DJF del Capítulo 1).

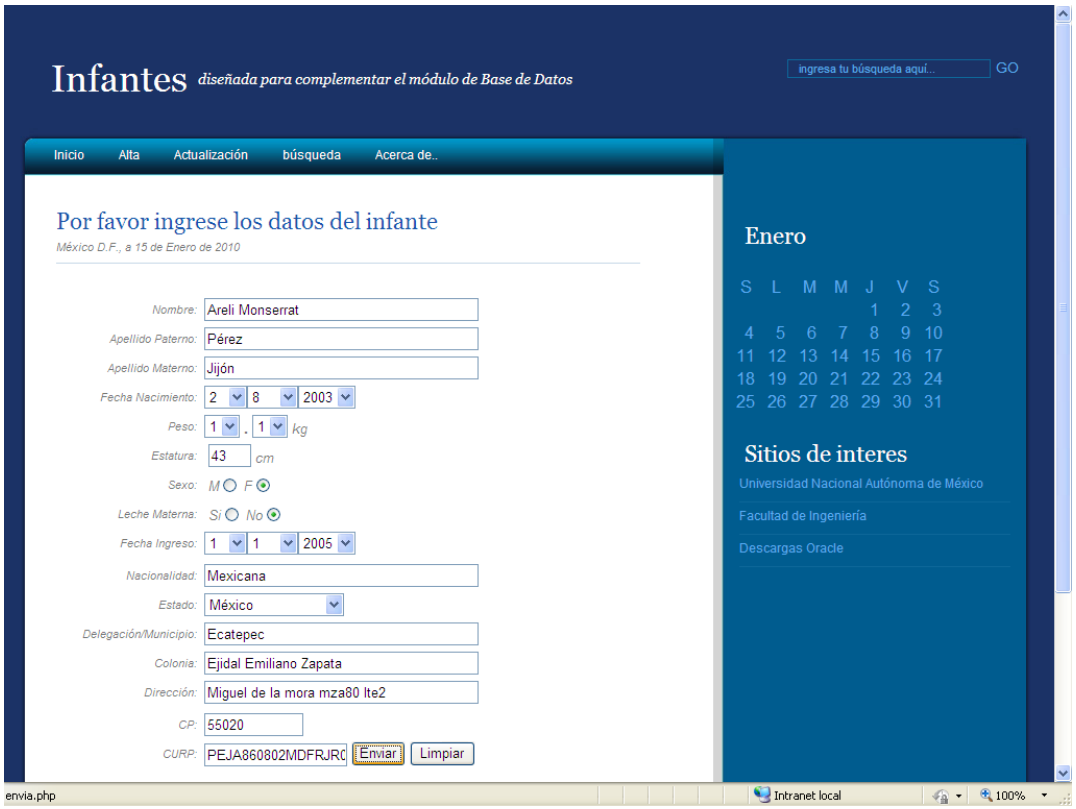


Fig. 2.14 Alta de infante

Resultado de un registro exitoso.

Fig. 2.15 Resultado exitoso

Si los datos están mal, esta la opción para limpiar los campos y volver a registrar los datos.

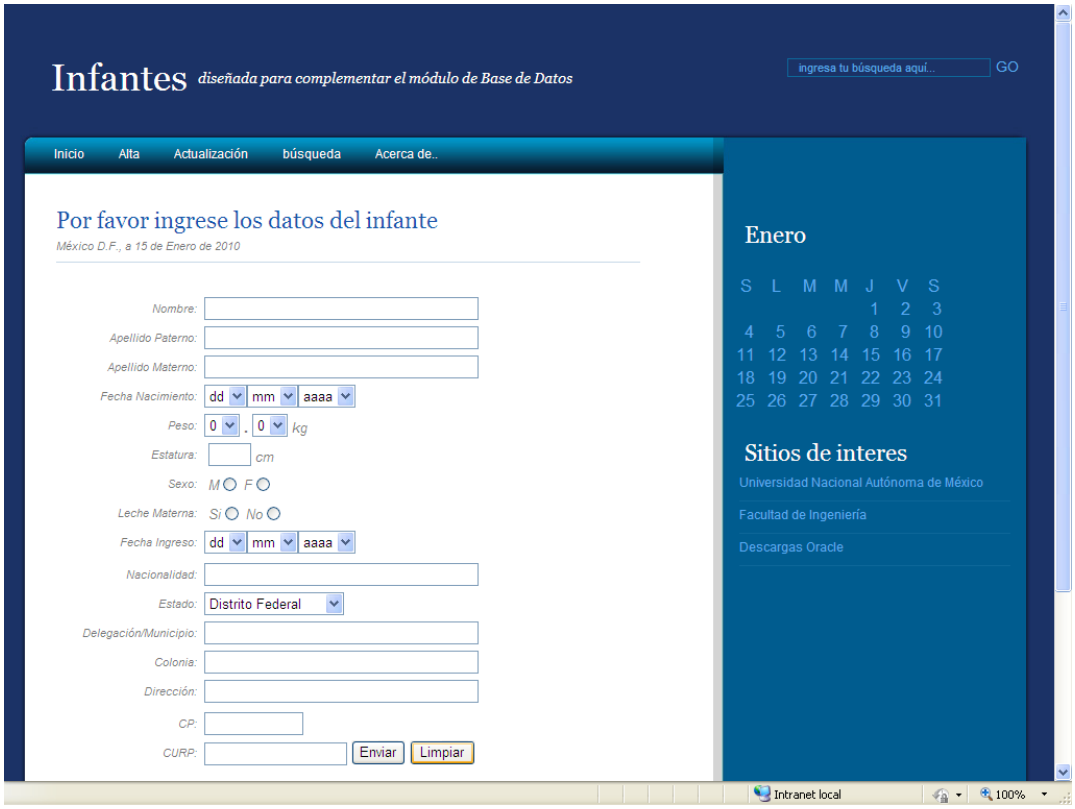


Fig. 2.16 Búsqueda de infantes

Al seleccionar “Búsqueda” se despliega la siguiente página para buscar los datos de un infante

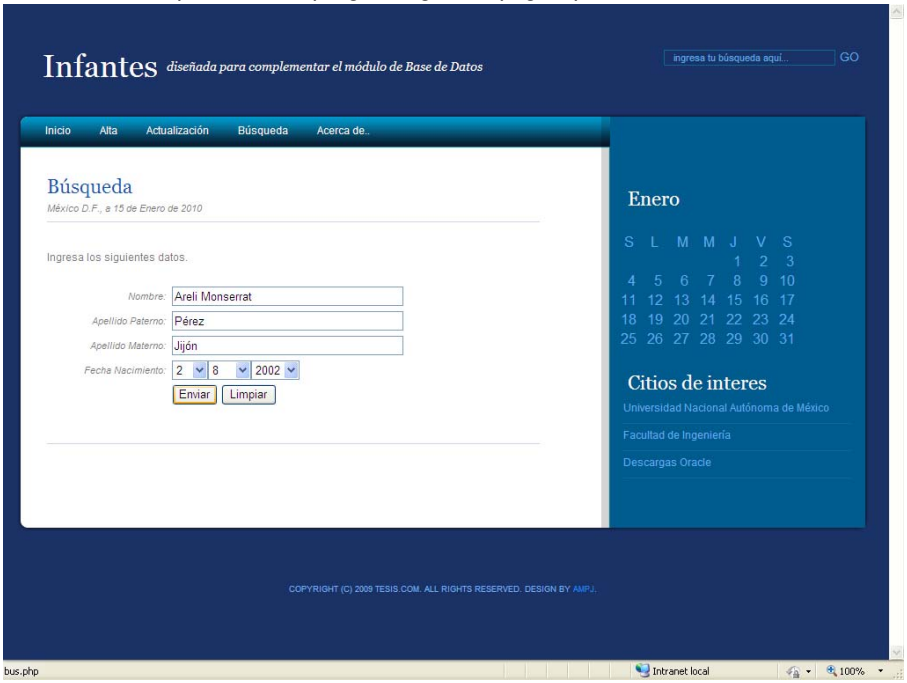


Fig. 2.17 Resultado de la Búsqueda de infantes

CONCLUSIÓN CAPÍTULO 2:

Aprendimos a administrar mejor los recursos de nuestras Bases de Datos, esto nos ayuda a tener un eficaz acceso a los datos lo cual nos permitirá tener una aplicación más rápida y segura, además reforzamos los conocimientos adquiridos en Base de Datos Avanzadas como lo son diferentes tipos de índices (bitmap y reverse), clusters, tablas de índice organizado(tablas e índice en un solo segmento), constraints diferidos, tipos de dato definidos por el usuario, usuarios permisos, roles, perfiles; para así tener más y mejores herramientas para el desarrollo de la tesis.

Además de optimizar los recursos de nuestra base de datos utilizando el archivo de configuración initEstancia.ora o los comandos alter system y alter session, al igual de optimizar el SQL , con sqltrace , tkproc ó explain plan haciendo a la base de datos más eficaz y segura

Capítulo 3
Depósitos de Datos.

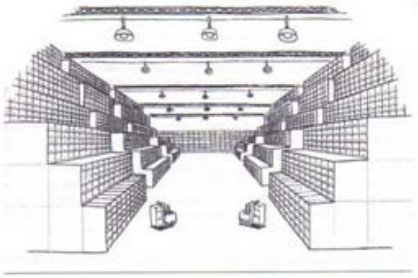


Fig 3.1 El antiguo data warehouse

Introducción

Un depósito de datos o “data warehouse” es el resultado de reunir los datos de la principal base de datos de una empresa y nutrirla con datos de otras fuentes (internas o externas) que ayuden a darle más valor a los mismos. Estos datos posteriormente se transformaran en grupos de información específicos (datamarts) con los cuales se realizarán nuevos análisis que servirán de soporte para futuras tomas de decisiones gerenciales.

La era de la computación comenzó hace algunos años y con ello muchas empresas sin darse cuenta comenzaron a acumular grandes cantidades de información. Actualmente, con la tendencia de la industria de moverse hacia configuraciones tanto de hardware como de software más potentes resulta más fácil contar con equipos de cómputo potentes, capaces de procesar y realizar análisis de estas grandes cantidades de información para obtener respuestas importantes las cuales ayudarán a realizar tomas de decisiones en las empresas.

El mundo actual está lleno de grandes cantidades de información sin depurar, es por eso que el acceso y la comprensión de dicha información significa poder y una ventaja competitiva que equivale a supervivencia.

Aun teniendo en la actualidad equipos de computo muy potentes resulta difícil determinar el patrón de uso del CPU por parte del data warehouse, e incluso si lo determináramos, éste sería inconsistente, lo cual nos lleva a una conclusión importante: no se debe mezclar los sistemas de data warehouse con los sistemas operacionales.

Como dice Bill Inmon, el padre de data warehouse: “Es imposible conocer por adelantado cuáles serán las consultas típicas”, ya que la gestión desplaza el foco de atención constantemente según las cuestiones relevantes que conciernen al negocio en cualquier punto del tiempo.

Datamart

Un datawarehouse está compuesto de datamarts. Un datamart es un subconjunto del data warehouse con un propósito específico. Por ejemplo, puede que tengamos un subconjunto financiero y un subconjunto de marketing, cada uno de ellos diseñado para surtir de información a una determinada parte de un negocio corporativo.

Sistema de ayuda a la decisión

Tenemos constancia de dos aproximaciones al proceso de construcción.

- Una aproximación consiste en gastar un tiempo extra en construir primero un núcleo de data warehouse y luego usarlo como base para realizar rápidamente muchos datamarts.

- La otra aproximación consiste en construir primero un datamart específico para trabajo en grupo. Esta aproximación pone rápidamente los datos en manos de los usuarios, pero el trabajo para poner la información en el datamart puede no ser rentalizable cuando se mueven dichos datos a un warehouse o cuando se intenta usar datos similares en un datamart diferente. Se gana velocidad, pero no portabilidad.

Retorno de las inversiones (ROI)
¿Por qué construir un datawarehouse?

Las ventajas que nos dará el uso de un datawarehouse se responde con un análisis de retorno de las inversiones, en el cual la empresa encontrara respuesta a preguntas claves sobre el datawarehouse como:
- ¿Nos dará una ventaja competitiva?
- ¿Mejorará la base del negocio?
- ¿Qué riesgo corremos si no lo hacemos?
- ¿Qué riesgo corremos si lo hacemos?

Existen muchas razones por las que un warehouse ayuda a las compañías a mejorar la base del negocio. La razón principal es: “Buena información provoca buenas decisiones”. En la actualidad, las compañías sufren constantes presiones del mercado y deben responder rápido o la competencia los arrollará.

Con un data warehouse efectivo, la gestión principal tiene en su mano toda la información pertinente para actuar rápido, de forma decisiva y de manera informada.

3.1 Necesidades

La información histórica de nuestra estancia infantil representa una gran bola de cristal a la cual podemos formularle algunas preguntas sobre el comportamiento de nuestra estancia o con la que se nos antojaría poder visualizar de una manera cómoda resúmenes de nuestra información para poder realizar futuras tomas de decisiones.
Para poder emplear de manera efectiva nuestra información, hace falta trabajar nuestra información para darle un formato que nos convenga para posteriormente agruparla en ciertas estructuras de información que nos faciliten su manipulación.

Nuestros datos históricos tienen que pasar por un proceso de “Extracción, Transformación y Carga”

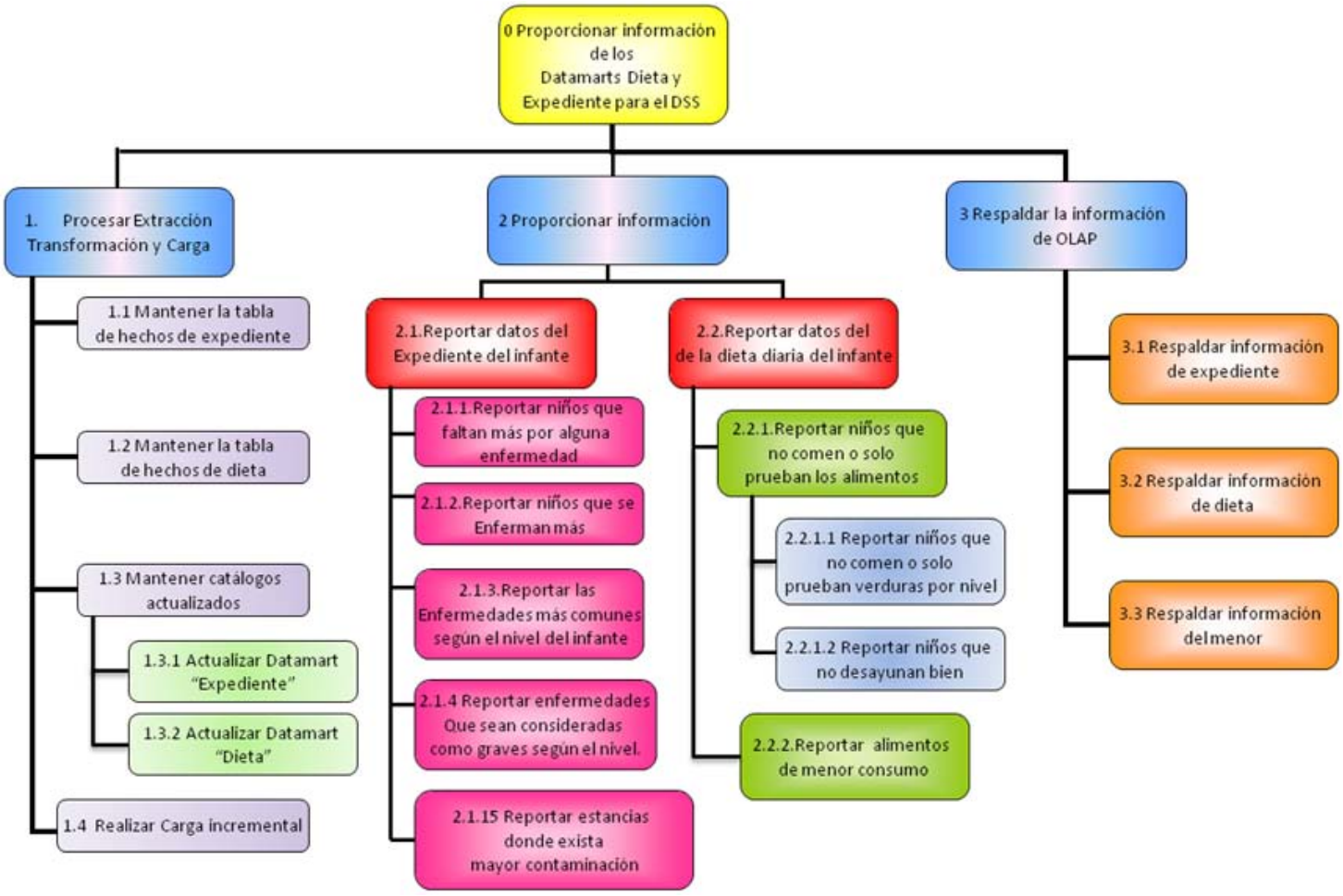
Dado que nuestra estancia tiene como elemento principal a los niños, podemos pensar rápidamente que las respuestas que más nos interesa conocer de nuestra estancia son las referentes a la alimentación de nuestros infantes (DIETA) y lo referente a la salud de nuestros infantes (EXPEDIENTE).

Es por eso que hemos decidido trabajar en estos dos puntos importantes de nuestra estancia, dicho de otra forma, realizaremos dos datamarts llamados DIETA y EXPEDIENTE.Para nuestra estancia y cualquier otra estancia sabemos que preguntas comunes y de gran interés para las personas que trabajan en las mismas son:

- ¿Qué niños que faltan más por alguna enfermedad?,
- ¿Cuáles son los niños que se enferman más?,
- ¿Cuáles son las enfermedades más comunes según el nivel del infante?,
- ¿Cuáles son las enfermedades consideradas como graves que aquejan a los infantes según su nivel?,
- ¿Cuáles son las estancias donde existe mayor nivel de contaminación?,
- ¿Cuáles son los niños que no comen o solo prueban alimentos?,
- ¿Qué niños no comen o sólo prueban verduras por nivel?,
- ¿Qué niños no desayunan bien? y
- ¿Cuáles son los alimentos de menor consumo?

Podemos ver de forma más clara lo que tenemos que hacer y las respuestas que queremos que nuestra información histórica nos conteste, en el siguiente Diagrama Jerárquico Funcional:

Diagrama Jerárquico Funcional (OLAP)



3.2 Análisis del Data Warehouse

Arquitectura de los almacenes de datos

El almacén de datos como integrador de diferentes fuentes de datos. Recoge, fundamentalmente, datos históricos, es decir, hechos, sobre el contexto en el que se desenvuelve la organización. Los hechos son, por tanto, el aspecto central de los almacenes de datos

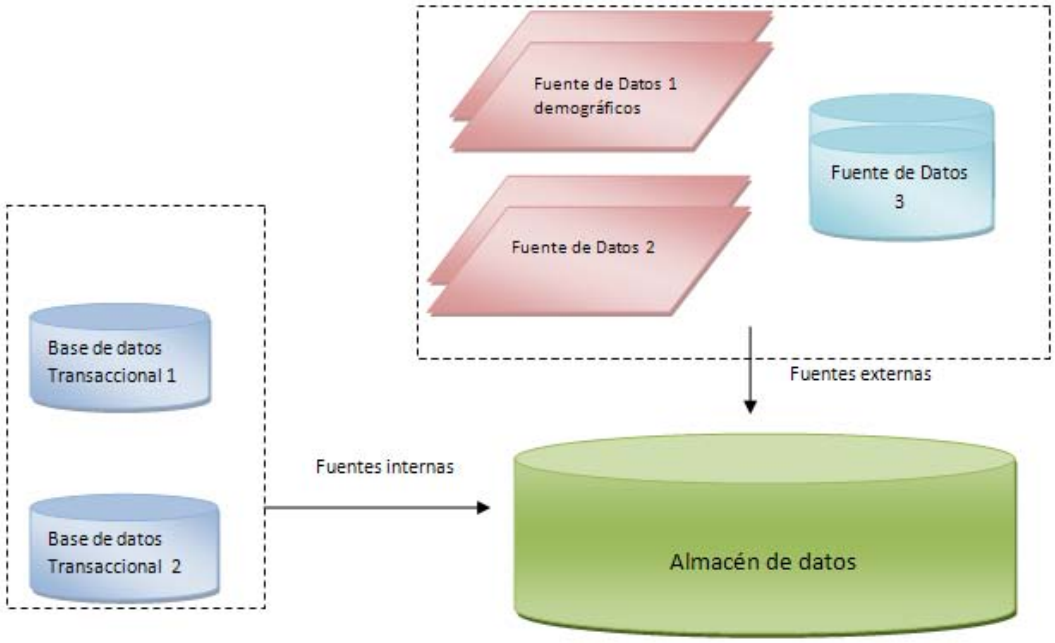


Fig 3.2. Almacenes de datos

Modelo Multidimensional ROLAP

El modelo conceptual de datos para los almacenes de datos es el modelo Multidimensional

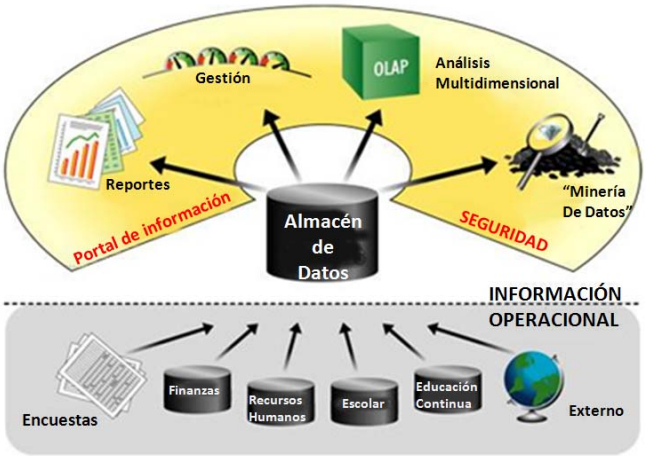


Fig 3.3. Modelo ROLAP

Los datos se organizan en torno a los hechos, que tienen unos atributos o medidas que pueden verse en mayor o menor detalle según ciertas dimensiones.
Las medidas responden generalmente a la pregunta “cuánto”, mientras que las dimensiones responderán al “cuándo”, “qué”, “dónde”, etc.

Aplicado a nuestro proyecto de estancias, en la tabla de hechos de **Expediente:**
Cada expediente tiene como **Medidas**: dosis, días permiso, y las **dimensiones**: menor, parentesco, estado salud, medicamento, día, enfermedad y estancia.
De igual forma, en la tabla de hechos de **Dieta:**
Cada expediente tiene como **Medida**: consumo y las **dimensiones**: menor, alimento y día.

La forma en que nuestra estancia tendría que organizar su almacén de datos para crear dos estructuras multidimensionales (una llamada Dieta y otra Expediente) ROLAP es la siguiente:

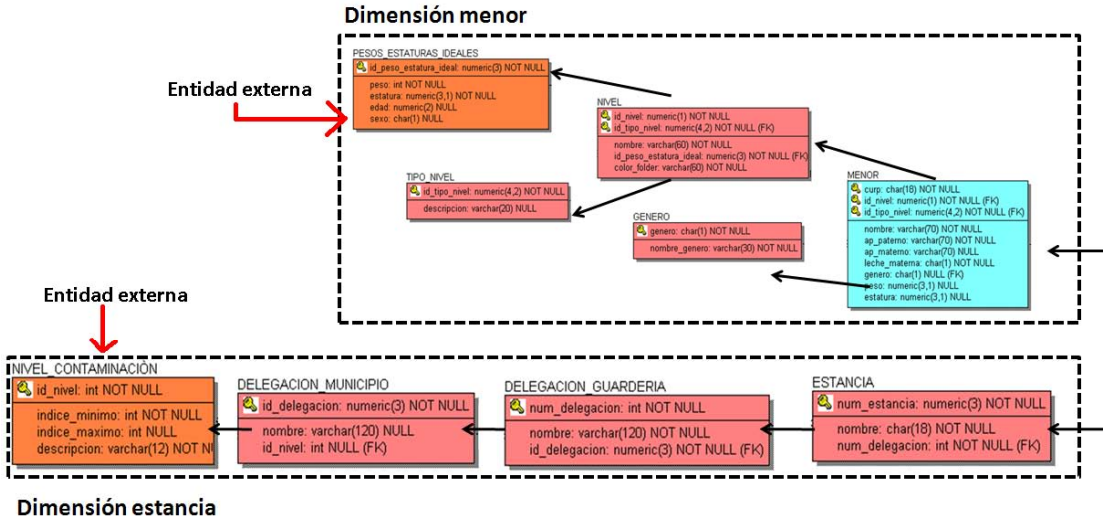


Fig 3.4 Datamart Expediente

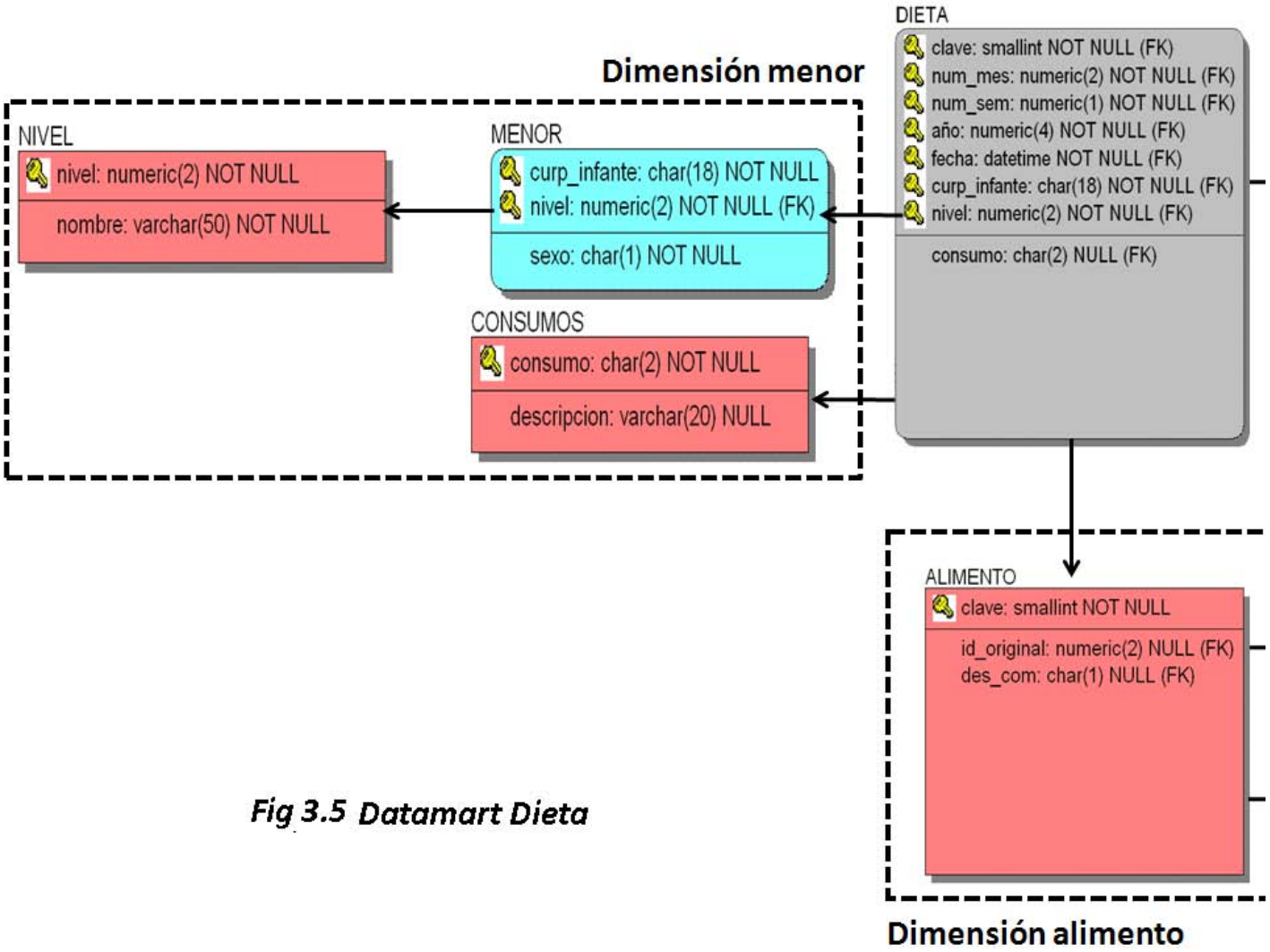


Fig 3.5 Datamart Dieta

Las flechas se pueden leer como “se agrega en”.

Cada dimensión tiene una estructura jerárquica pero no necesariamente lineal, por ejemplo en las dimensiones de tiempo y producto hay más de un camino posible de agregación (ruta de agregación). Esto permite la definición de hechos agregados con mucha facilidad.
La forma que tienen estos conjuntos de hechos y sus dimensiones hace que se llamen almacenes de datos

- “Estrella simple”, cuando no hay caminos alternativos en las dimensiones o
- “Estrella jerárquica” o “copo de nieve”, cuando sí hay caminos alternativos en las dimensiones, como el ejemplo anterior.

Cuando el número de dimensiones no excede de tres, podemos representar cada combinación de niveles de agregación como un cubo. El cubo está formado por casillas, cada casilla representa un hecho.

Ejemplo Cubo Expediente

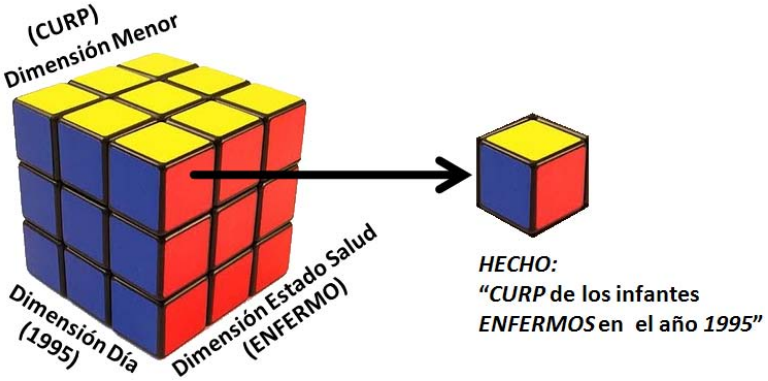


Fig 3.6. Visualización de un hecho en un modelo multidimensional para el Datamart Expediente

Ejemplo Cubo Dieta

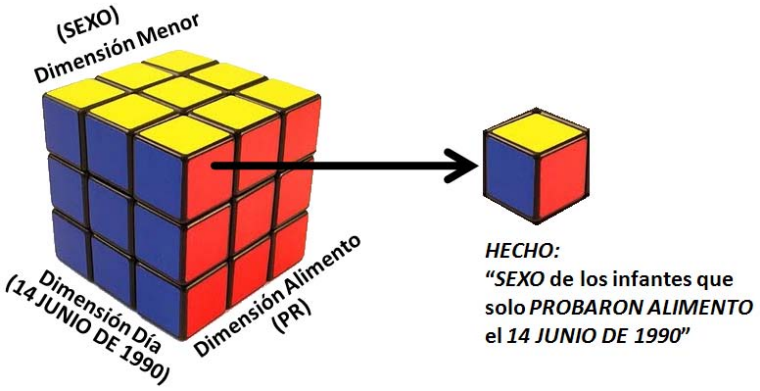


Fig 3.7.Visualización de un hecho en un modelo multidimensional para el Datamart Dieta

Esta visualización hace que, incluso cuando tengamos más de 3 dimensiones, se hable de un “cubo” (o mejor dicho hipercubo) como un conjunto de niveles de agregación para todas las dimensiones. Esta estructura permite ver de una manera intuitiva la sumarización/agregación (varias casillas se fusionan en casillas más grandes), la disgregación (las casillas se separan en casillas con mayor detalle).

Datamarts
La representación de todo el almacén de datos como una sola estrella, no es posible. Por ejemplo, la información de personal de una empresa (empleados, departamentos, proyectos, etc.) es difícilmente integrable en la misma estrella que las ventas. Incluso en ámbitos más relacionados por ejemplo, ventas y producción.

La idea general es que para cada subámbito de la organización se va a construir una estructura de estrella.
Por tanto el almacén de datos (data warehouse) estará formado por muchas estrellas formando una "constelación".

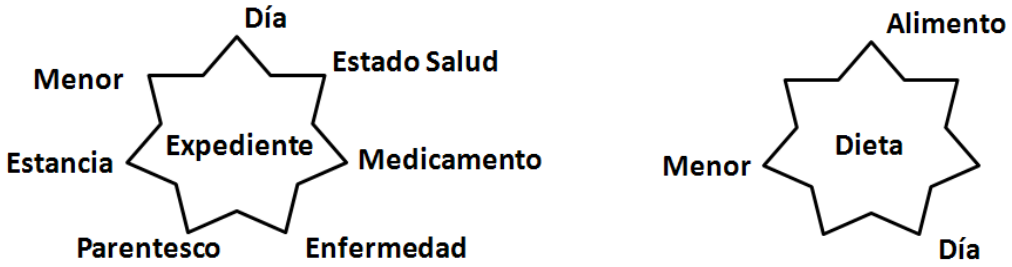


Fig 3.8. Estructura estrella para datamart Expediente y Dieta

Cada una de estas estrellas que representan un ámbito específico de la organización se denomina **"datamart" (mercado de datos).**

La única dimensión que suele aparecer en todos los datamarts es la dimensión tiempo (para nuestro caso, Día), ya que el almacén de datos representa información histórica.
Los almacenes de datos contienen información redundante, la estructura anterior es la estructura externa, visible o conceptual. Esta estructura no determina la manera de implementarlo ni lógica ni físicamente.

Para construir esta estructura se construyen 3 tipos de tablas:

1. Tablas copo de nieve (snowflake tables): para cada nivel de agregación de una dimensión se crea una tabla. Cada una de las tablas tiene una PK y tantas FK's como sean necesarias para conectar con los niveles de agregación superiores. Permiten realizar informes utilizando diferentes grados de detalle sobre varias dimensiones. Al estar normalizadas permiten seleccionar datos dimensionales de manera no redundante. Esto es útil para los operadores drill, slice & dice y pivot
2. Tabla de hechos (fact tables): se crea una única tabla de hechos por datamart. En esta tabla se incluye un atributo para cada dimensión, que será FK a cada una de las tablas copo de nieve de mayor detalle de cada dimensión. Además, todos estos atributos forman la clave primaria. Adicionalmente, pueden existir atributos que representen información de cada hecho, denominados generalmente medidas.
3. Tablas estrella (star tables): para cada dimensión se crea una tabla que tiene un atributo para cada nivel de agregación diferente en la dimensión. Cada uno de estos atributos es una FK que hace referencia a tablas copo de nieve. Todos los atributos de la tabla forman la PK. Las tablas estrella son, en realidad, tablas de apoyo, ya que no representan ninguna información que no esté en las demás. son tablas de apoyo, que representan "pre-concatenaciones", o "pre-joins" entre las tablas copo de nieve. Su propósito es evitar concatenaciones costosas cuando se realizan operaciones roll-up.

Podemos identificar 4 pasos principales a la hora de diseñar un almacén de datos (para cada datamart)

1. Elegir "dominio" de la organización sobre el que se deseen realizar informes complejos, por ejemplo, se puede hacer un datamart sobre pedidos, ventas, facturación, etc.
2. Decidir el hecho central y el "gránulo" (nivel de detalle) máximo que se va a necesitar sobre él. En general, siempre hay que considerar gránulos finos por si más adelante se fueran a necesitar, a no ser que haya restricciones de espacio.
3. Identificar las dimensiones que caracterizan el "dominio" y su jerarquía de agregación, así como los atributos básicos de cada nivel. No se debe incluir atributos descriptivos más que lo imprescindible para ayudar en la visualización. Nota: El tiempo siempre es una (o más de una) de las dimensiones presentes

4. Determinar y refinar las medidas y atributos necesarios para los hechos y las dimensiones. Generalmente las medidas de los hechos son valores numéricos agregables (totales, cuentas, medias…). Revisar si toda la información que se requiere sobre los hechos está representada en el almacén de datos. NOTA: *No se debe utilizar la misma codificación de PK's que en la BD transaccional.*

En general las **tablas de hechos** tienen muchas filas y relativamente pocas columnas. Las **tablas de dimensión** representan las diferentes perspectivas desde donde se ven y analizan los hechos de la tabla de hechos. A diferencia de las anteriores, su clave primaria está formada por un solo atributo

En general suelen tener muchas columnas pero pocas filas. Siempre que sea posible, es conveniente **compartir las tablas de dimensión entre distintas tablas de hechos.**
Una de las dimensiones más comunes es la que representa el tiempo, con atributos que describen periodos para años, cuatrimestres, periodos fiscales, y periodos contables. Otras dimensiones comunes son las de clientes, productos, representantes de ventas, regiones, sucursales.

En la siguiente figura vemos un ejemplo de **esquema Estrella**, donde la tabla de hechos es la tabla Dieta, y el resto son las tablas de dimensiones

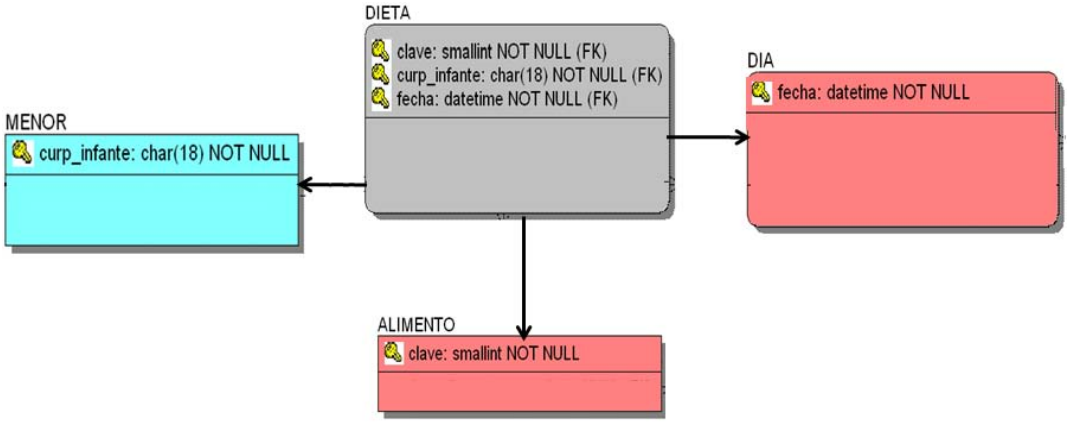


Fig 3.9. Esquema estrella para el datamart Dieta

Esquema Snowflake

Es una variante al esquema estrella en el cual las tablas de dimensión están normalizadas, es decir, pueden incluir claves que apuntan a otras tablas de dimensión

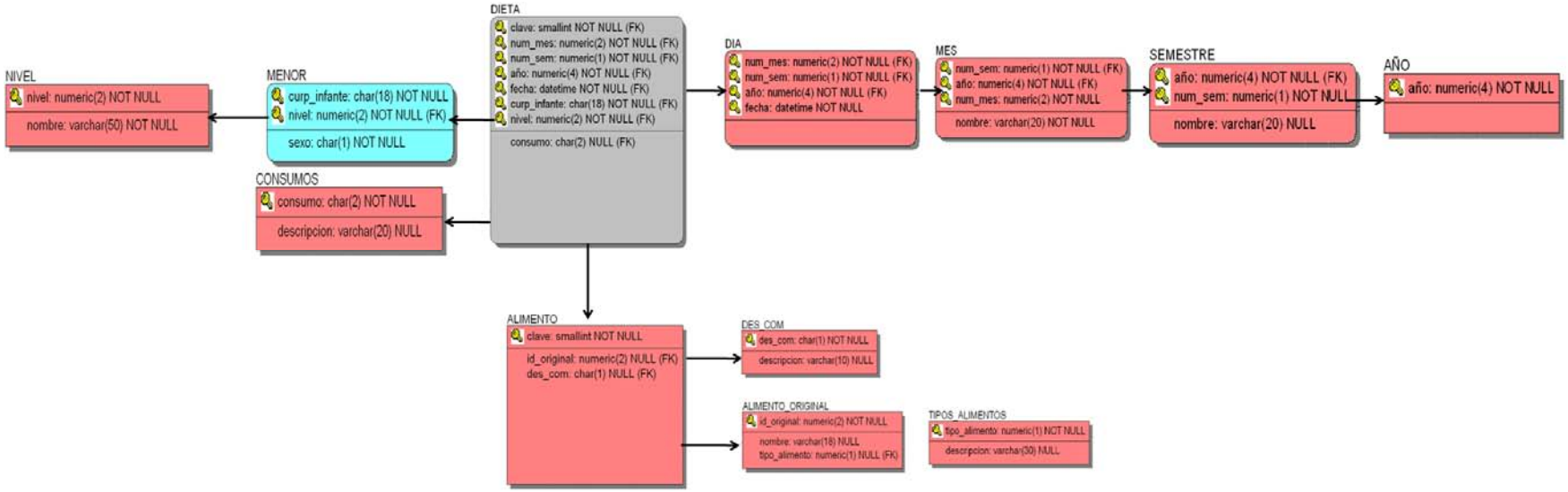


Fig 3.10. Esquema estrella para el datamart Dieta

Las ventajas de esta normalización son la reducción del tamaño y redundancia en las tablas de dimensión, y un aumento de flexibilidad en la definición de dimensiones.
Sin embargo, el incremento en la cantidad de tablas hace que se necesiten más operaciones de JOINs para responder a las consultas, lo que empeora el performance. Otra desventaja es el mantenimiento adicional que requieren las tablas adicionales.

3.3 Diseño

Los DataWarehouses frecuentemente almacenan algunos datos en forma redundante, el tamaño de estas Bases de Datos puede ser enorme. Las Bases de Datos que se acercan o exceden el terabyte son llamadas VLDBs (Very Large Databases).
A continuación se detallan algunos temas que impactan sobre el diseño físico del DataWarehouse.

Particionamiento

Dividir los datos de una tabla en viarias tablas es particionar una tabla.
La tabla que se particiona se denomina tabla particionada y las divisiones se llaman particiones, también puede especificar qué partición utilizar como parte de la cláusula **from** de las consultas.

Los criterios que sirven para determinar qué filas se almacenan en qué particiones se especifican como parte del comando **create table**.

En Oracle una tabla puede ser particionada en hasta 64.000 particiones separadas.
También se puede configurar atributos físicos tales como *pctfree* y *pctused*.
Se recomiendan cuando:

- La tabla tiene un tamaño superior a 2 GB.
- Tablas que mantienen históricos.

Creación de una tabla particionada en Oracle

1) Particionamiento por rangos.

Los rangos determinarán los valores almacenados en cada partición.
Se deben considerar las siguientes reglas:

- Cada partición se define con la cláusula VALUES LESS THAN, la cual especifica un límite superior. Cualquier valor de la clave de la partición igual o superior, es añadida a la próxima partición.
- Todas las particiones, excepto la primera, tienen un límite inferior implícito, especificado en la cláusula VALUES LESS THAN de la partición previa.
- Un literal MAXVALUE puede ser definido para la última partición; representa un valor virtual de infinito.

Ejemplo: (datamart dieta)

```
CREATE TABLE MENOR (
     curp_infante       CHAR(18) NOT NULL primary key,
     nivel              NUMBER(2) NOT NULL primary key,
     sexo               CHAR(1) NOT NULL,
Constraint FK_NIVEL_MENOR foreign key (nivel) references nivel(nivel)
) partition by range (nivel)
  (partition PART1          values less than (2)
    tablespace PART1_TS,
    partition PART2         values less than (3)
    tablespace PART2_TS,
partition PART3   values less than (4)
    tablespace PART3_TS,
partition PART4   values less than (5)
    tablespace PART4_TS,
partition PART5   values less than (6)
    tablespace PART5_TS,
partition PART6   values less than (7)
    tablespace PART6_TS,
partition PART7   values less than (8)
    tablespace PART7_TS,
partition PART8   values less than (9)
    tablespace PART8_TS)
```

La columna que se utiliza como base para la lógica de la partición no suele ser la clave primaria de la tabla. Suele ser más habitual una de las columnas de clave externa de la tabla (foránea).

Cuando se particiona una tabla, las particiones deberían almacenarse en tablespaces diferentes, permitiendo controlar su ubicación de almacenamiento físico.
No se pueden realizar particiones de objetos que utilicen tipos de datos LOB (como LONG o LONG RAW).

Cuándo usar particionamiento por rango

Típicamente las tablas más grandes de la base de datos son candidatas a ser particionadas usando rangos.
Ejemplos:

- Datos basados en tiempo (pedidos semanales)
- Datos ordenados (rangos de salario)

El particionamiento por rango permite archivar datos obsoletos. Ejemplo:
Eliminar extractos bancarios con más de 5 años
El particionamiento por rango permite adicionar nuevos rangos, sin la necesidad de reorganizar el resto de la tabla.

Consultas directas a las particiones.
Si sabe cuál es la partición desde la que se van a recuperar los datos, puede especificar su nombre como parte de la clausula from de la consulta.
Por ejemplo:

```
Select *
From MENOR partition (PART2)
Where nivel between 3 and 8;
```

Oracle conoce el valor más alto que puede haber en cada partición para determinar qué particiones deberían emplearse en la resolución de una consulta, las particiones se pueden almacenar en tablespace diferentes (y, por tanto, en dispositivos de disco distintos), lo que ayuda a reducir las posibles contiendas de E/S durante el procesamiento de la consulta.
Durante un proceso de inserción (insert) en la tabla particionada. Oracle utiliza las restricciones de rangos de las particiones para determinar en qué partición se debe insertar el registro.

Indexación de particiones.
Cuando se crea una tabla particionada, se debería crear un índice en la tabla. Este índice se podría particionar de acuerdo con el mismo rango de valores utilizado para particionar la tabla.

Ejemplo: (datamart dieta)

```
Create index MENOR_NIVEL_IDX
On MENOR(nivel)
Local
 (partition PART1
  tablespace PART1_NDX_TS,
partition PART2
  tablespace PART2_NDX_TS,
partition PART3
  tablespace PART3_NDX_TS,
partition PART4
  tablespace PART4_NDX_TS,
partition PART5
  tablespace PART5_NDX_TS,
partition PART6
  tablespace PART6_NDX_TS,
partition PART7
  tablespace PART7_NDX_TS)
partition PART8
  tablespace PART8_NDX_TS)
```

Fíjese en la palabra clave **local**. En este comando **create index**, no se especifica ningún rango. En su lugar, la palabra clave **local** le indica a Oracle que cree un índice diferente para cada partición de la tabla MENOR, son “locales” a las particiones.
También se pueden crear índices “globales”.
La cláusula **global** de este comando **create index** permite especificar rangos para los valores de índice que sean *diferentes* de los rangos de las particiones de tabla.

Gestión de particiones

- El comando ***alter table*** se puede utilizar para administrar los parámetros de storage de las particiones.
En general se permite modificar la estructura de particiones existente, lo cual puede tener que hacerse después de que se haya utilizado mucho una tabla particionada.

Otros Criterios de partición.

2) Partición por lista.

Una partición es asignada a una lista de valores. Si la llave de partición tiene uno de éstos valores, la partición se escoge. Por ejemplo, todos los niveles en una estancia con id 1,2 y 3 es decir lactante A, lactante B y lactante C podrían construir una partición para el nivel *lactante*.
Se especifica una lista de valores discretos para la clave de particionamiento. No se soportan claves de particionamiento formadas por varios atributos.

Ejemplo: (datamart dieta)

```
CREATE TABLE nivel_list
(nivel NUMBER(2) NOT NULL,
  nombre VARCHAR(50) NOT NULL,
)
PARTITION BY LIST(nivel)
(PARTITION lactante  VALUES('1','2','3'),
  PARTITION maternal  VALUES ('4', '5')
  PARTITION preescolar  VALUES ('6', '7','8') );
```

3) Partición Hash.

El valor de una función hash determina la partición. Asumiendo que hay cuatro particiones, la función hash podría regresar un valor de 0 a 3.
La correspondencia entre las filas y las particiones se realiza a través de una función de hash. Es una opción útil cuando:

- Se desconoce la correspondencia en función de los rangos y es difícil balancearla manualmente.

Ejemplo: (datamart dieta)

```
CREATE TABLE alimento_hash
(clave SMALLINT,
 tipo_alimento NUMBER(1),
 id_original NUMBER(2)),
 des_com CHAR(1)
PARTITION BY HASH(clave)
PARTITIONS 4
STORE IN (data1, data2, data3, data4);
```

La tabla alimento_hash se particiona en función de los valores de la columna clave_id. Las particiones se almacenan en los tablespaces: data1, data2, data3, y data4.

Cuándo usar particionamiento hash

En tablas grandes sin orden particular de valores sobre una llave que sea comúnmente usada por las aplicaciones para ordenamiento o agrupamiento.
Las particiones *hash* brindan balanceo de carga de los datos entre particiones.
La mayor parte de los tipos de datos pueden ser usados como columnas para la función *hash*.
Es fácil crear *n* particiones, ya que no se requiere especificar un rango o lista de valores para cada partición.
Es importante no elegir una columna con un alto grado de valores duplicados o en donde un valor sea mucho más frecuente que otros, ya que esto puede generar particiones no balanceadas (unas más grandes que otras).

4) Partición compuesta o Particionamiento por Composición.

La partición compuesta permite ciertas combinaciones de las particiones anteriores mencionadas; por ejemplo, en una partición que se aplique la partición por rangos y luego la partición hash.
Cada partición se particiona a su vez es sub-particiones. Las particiones más generales se hacen con el método de rango. Cada partición se sub-particiona con el método de hash o por lista.

(datamart expediente)

```
CREATE TABLE expediente
    (curp               CHAR(18) NOT NULL,
     num_estancia       NUMBER(3) NOT NULL,
     id_mes             NUMBER(2) NOT NULL,
     id_semestre        NUMBER(1) NOT NULL,
     id_año             NUMBER(4) NOT NULL,
     id_dia             NUMBER(2) NOT NULL,
     id_nivel           NUMBER(1) NOT NULL,
     dosis              NUMBER(3) NULL,
     dias_permiso       NUMBER(2) NULL,
     num_medicamento    NUMBER(2) NULL,
     id_parentesco      SMALLINT NULL,
     num_enfermedad_antecedente NUMBER(3) NULL,
     num_enfermedad     NUMBER(3) NULL,
     sano               CHAR(1) NULL,
     fecha          DATE
          )
          PARTITION BY RANGE(fecha)
          SUBPARTITION BY HASH(id_nivel)
          SUBPARTITION TEMPLATE(
          SUBPARTITION sp1 TABLESPACE data1,
          SUBPARTITION sp2 TABLESPACE data2,
          SUBPARTITION sp3 TABLESPACE data3,
          SUBPARTITION sp4 TABLESPACE data4)
          (PARTITION exp_jan2008 VALUES THAN (TO_DATE('01/02/2008','DD/MM/YYYY')
            PARTITION exp_fec2008 VALUES THAN (TO_DATE('01/03/2008','DD/MM/YYYY')
            PARTITION exp_mar2008 VALUES THAN (TO_DATE('01/04/2008','DD/MM/YYYY')
            PARTITION exp_apr2008 VALUES THAN (TO_DATE('01/05/2008','DD/MM/YYYY'))
```

Ejemplo de Particionamiento en SQL Server

Como hemos mencionado, en los ejemplos anteriores, las tablas deben particionarse y ubicarse en varios discos, pues el particionamiento permite que los datos de una tabla lógica se repartan en múltiples conjuntos de datos físicos.

Es sabido por todo el mundo de las capacidades de particionamiento que posee Oracle desde sus más antiguos releases. En Sql Server 2000 existía una especie de "chapuza" para poder hacerlo mediante restricciones "CHECK" en los campos y utilizando vistas mediante UNIONS.

El panorama en SQL Server 2005 a mejorado un poco respecto a eso, ahora veremos como particionar una tabla con dicho manejador.

Ejemplo:

Este Particionamiento se basa en una columna de la tabla de hechos Dieta, que es la que indica la fecha, en este caso de consumo.

Primero debemos de crear nuestra base de datos con diferentes filegroups definidos, que es donde vamos a colocar las particiones de la tabla.

```
ALTER DATABASE Dieta ADD FILEGROUP [FGR]
GO
```

Ahora le indicamos el espacio aproximado y la ruta donde se guardará dicha partición:

```
ALTER DATABASE Dieta
ADD FILE
 (NAME = N'fgr',
 FILENAME = N'C:\Documents and Settings\Tania RL\Mis documentos\Titulación\Nuevo\fgr.ndf',
```

```
  SIZE = 1MB,
  MAXSIZE = 5MB,
  FILEGROWTH = 5MB)
TO FILEGROUP [fgr]
GO
```

Después debemos de crear una función del tipo partición, que nos va a dar el rango para cada una de las particiones, en este caso será por décadas de año.

```
CREATE PARTITION FUNCTION
fechaPF(datetime)
AS RANGE RIGHT
FOR VALUES ('19700101','19800101','19900101','20000101','20081231')
GO
```

Ahora creemos un esquema para esta partición, para mapear las particiones a los filegroups.

```
CREATE PARTITION SCHEME fechaPS
AS PARTITION fechaPF ALL TO ([PRIMARY])
Go
```

Finalmente podemos crear una tabla particionada en el esquema que creamos para la partición, especificando una columna particionada apropiada, en este caso fecha.

```
CREATE TABLE DIETA (
      clave             int NOT NULL,
      curp_infante      char(18) NOT NULL,
      num_mes           numeric(2) NOT NULL,
      num_sem           numeric(1) NOT NULL,
      año               numeric(4) NOT NULL,
      fecha             datetime NOT NULL,
      nivel             numeric(2) NOT NULL,
      consumo           char(2)  NULL
)ON fechaPS(fecha)
GO
```

Listo podemos crear los constraints necesarios y seguir con la creación de objetos de nuestro datarmart.

Indexación de particiones.

Además de la partición del conjunto de datos de una tabla, puede dividir los índices particionando la tabla y sus índices de uso de la misma función que a menudo optimiza el rendimiento. Cuando los índices de la tabla y usar la función de partición misma y las columnas en el mismo orden, se alinean la tabla y el índice. Si se crea un índice en una tabla ya con particiones, SQL Server ajusta automáticamente el nuevo índice con el esquema de partición de la tabla a menos que el índice se expresa con particiones de manera diferente. Cuando se alinean una tabla y sus índices, a continuación, SQL Server puede mover particiones dentro y fuera de las tablas con particiones más eficaces, porque todos los datos e índices relacionados están divididos, con el mismo algoritmo.

Cuando las tablas e índices se definen con la función no sólo la misma partición, sino también el mismo esquema de partición, se considera que el almacenamiento alineados. Uno de los beneficios a la alineación de almacenamiento es que todos los datos dentro del mismo ámbito se encuentra en el mismo disco físico (s). En este caso, las copias de seguridad se puede aislar a un período de tiempo determinado y sus estrategias pueden variar, en términos de frecuencia y el tipo de copia de seguridad, basado en la volatilidad de los datos. Beneficios adicionales pueden ser vistos cuando las tablas e índices en el mismo archivo o grupo de archivos se unen o agregados. Los beneficios de SQL Server de paralelización de una operación a través de particiones. En el caso de la alineación de almacenamiento y CPU múltiples, cada procesador puede trabajar directamente en un archivo específico o grupo de

archivos, sin conflictos de acceso a los datos ya que todos los datos necesarios en el mismo disco. Esto permite ejecutar varios procesos en paralelo, sin interrupción.

Ejemplo: (datamart dieta)

```
ALTER DATABASE Dieta ADD FILEGROUP [FGR]
GO
ALTER DATABASE Dieta
ADD FILE
 (NAME = N'fgr',
 FILENAME = N'C:\Documents and Settings\Tania RL\Mis documentos\Titulación\Nuevo\fgr.ndf',
 SIZE = 1MB,
 MAXSIZE = 5MB,
 FILEGROWTH = 5MB)
TO FILEGROUP [fgr]
GO
CREATE PARTITION FUNCTION
nivelPF(numeric)
AS RANGE --RIGHT
FOR VALUES (1,2,3,4,5,6,7,8)
GO
CREATE PARTITION SCHEME nivelPS
AS PARTITION nivelPF ALL TO ([PRIMARY])
Go
CREATE TABLE MENOR (
    curp_infante      char(18) NOT NULL,
    nivel             numeric NOT NULL,
    sexo              char(1)  NULL,
    estatura                                  numeric(3,1) NULL
)ON nivelPS(nivel)
GO
```

Gestión de particiones

Las tablas e índices con particiones pueden modificarse de las maneras que se indican a continuación:

- Modificar una función de partición para que vuelva a crear las particiones de las tablas e índices a las que haga referencia.

ALTER PARTITION FUNCTION sólo se puede utilizar para dividir en dos una partición o para mezclar dos particiones en una sola. Para cambiar el modo en que se crean las particiones de una tabla o un índice (de 10 a 5 particiones, por ejemplo) puede recurrir a cualquiera de las opciones que se indican a continuación. El consumo de recursos de cada opción varía en función de la configuración del sistema.

```
ALTER PARTITION FUNCTION partition_function_name()

{

   SPLIT RANGE ( boundary_value )

 | MERGE RANGE ( boundary_value ) }

[ ; ]
```

Ejemplo: (datamart dieta)

```
ALTER PARTITION FUNCTION nivelPF

(MERGE RANGE ( 3 )
```

- Modificar un esquema de partición para diseñar un grupo de archivos que albergue una partición recién agregada.

Para modificar un esquema de partición puede diseñar un grupo de archivos que contenga la siguiente partición que se agregará a la tabla con particiones. Para hacerlo debe asignar la propiedad **NEXT USED** a un grupo de archivos. Puede asignar la propiedad NEXT USED a un grupo de archivos vacío o a uno que ya contenga una partición. Es decir, un grupo de archivos puede tener más de una partición.

```
ALTER PARTITION SCHEME partition_scheme_name

    NEXT USED [ filegroup_name ] [ ; ]
```

Ejemplo: (datamart dieta)

```
ALTER PARTITION SCHEME nivelPS

NEXT USED FGR
```

- Convertir una tabla sin particiones en una tabla con particiones.

Existen dos maneras de convertir una tabla sin particiones en una tabla con particiones.

Una consiste en crear un índice agrupado con particiones en la tabla mediante la **instrucción CREATE INDEX.** Esta acción es similar a crear un índice agrupado en cualquier tabla, puesto que lo que hace SQL Server es quitar la tabla y volver a crearla en formato de índice agrupado. Si la tabla ya tiene algún índice agrupado con particiones, puede quitar el índice y volver a generarlo en un esquema de partición mediante **CREATE INDEX** y la cláusula **DROP EXISTING = ON.**

La otra manera consiste en utilizar la instrucción **ALTER TABLE SWITCH** de Transact-SQL para modificar los datos de la tabla por los de una tabla con particiones por intervalos que sólo tenga una partición. Esta tabla con particiones ya debe existir antes de que se lleve a cabo la conversión. También es preciso que la única partición que contenga esté vacía. Para obtener más información acerca de cómo dividir particiones

- Convertir una tabla con particiones en una tabla sin particiones.

Para cambiar una tabla con particiones en una tabla sin particiones, basta con modificar la función de partición de la tabla con particiones para que la convierta en una tabla con una sola partición. Aunque técnicamente siga siendo una tabla con particiones, este estado no es relevante para las operaciones posteriores que se vayan a efectuar en la tabla.

Si la tabla tiene aplicado un índice agrupado con particiones, simplemente debe quitar el índice y volver a generarlo como si se tratara de un índice sin particiones. Para hacerlo puede utilizar el comando **CREATE INDEX** de Transact-SQL con la cláusula **DROP EXISTING = ON**.

- Transferir datos agregando, moviendo o eliminando particiones.

La instrucción **ALTER TABLE...SWITCH** de Transact-SQL permite transferir rápida y eficazmente bloques de datos entre tablas con particiones.

Consultar particiones

Para consultar particiones individuales de una tabla o índice con particiones

```
[ database_name. ] $PARTITION.partition_function_name(expression)
```

Argumentos

database_name

Es el nombre de la base de datos que contiene la función de partición.

partition_function_name

Es el nombre de cualquier función de partición existente con la que se está aplicando un conjunto de valores de columnas de partición.

expression

Es una expresión cuyo tipo de datos debe coincidir con el tipo de datos de su columna de partición correspondiente o ser convertible en éste de forma implícita. *expression* también puede ser el nombre de una columna de partición que participa en ese momento en *partition_function_name*.

Ejemplo:

```
        (datamart dieta)
SELECT $PARTITION.NivelPF (2) ;
GO
```

Reorganizaciones en Oracle

Las cargas incrementales de las Bases de Datos irán *fragmentando* las tablas, y esta fragmentación puede resultar en un *decaimiento del performance*. La mayoría de los DBMSs proveen rutinas de ***reorganización*** para reclamar el espacio fragmentado y mover registros.

Las actividades básicas involucradas en la reorganización de una base de datos implican:

-copiar la base de datos vieja en otro dispositivo,

-rebloquear las filas y recargarlas.

Estas tareas no son triviales en un DataWarehouse, pero todos los DBMSs permiten reorganizar particiones, lo cual es otra buena razón para particionar las tablas.

Periódicamente DEALLOCATE en tablas

Asignando y reasignando el espacio de tablas

Utilizar TRUNCATE (y revisar opciones)

TRUNCATE TABLE [schema.] tabla
[{DROP|REUSE} STORAGE
Si se trunca la tabla reconstruya el índice.

Cabe mencionar que se utiliza menos espacio del registro de transacciones. La instrucción DELETE quita una a una las filas y graba una entrada en el registro de transacciones por cada fila eliminada. TRUNCATE TABLE quita los datos al cancelar la asignación de las páginas de datos utilizadas para almacenar los datos de la tabla y sólo graba en el registro de transacciones las cancelaciones de asignación de páginas.

Eliminación de tablas en Oracle con:

DROP TABLE … CASCADE CONSTRAINTS
Para después volver a crearlas con un nuevo STORAGE

INDEXADO en Oracle

Las columnas que se elijan para indexar deben ser las que se usan más frecuentemente para recuperar las filas.

Una vez que se determinan las columnas a indexar, hay que determinar la estrategia de índice. La mayoría de los DBMSs proveen varios algoritmos, entre ellos B-tree, Hash, llave Invertida y Binario.

Minimizar el número de índices en tablas volátiles

Considerar NOLOGGING para índices grandes.

Tipos de índices en SQL Server

Para mejorar el desempeño de las consultas se utilizan índices, los más utilizados en SQLServer son los **Clustered** y **Non-Clustered**.

Clustering

Esta técnica es importante porque mejora drásticamente el performance del acceso secuencial, y este tipo de acceso es el más usado en el procesamiento OLAP.

Cuando las filas de la tabla no permanezcan almacenadas en el orden correspondiente a su índice clustering, situación conocida como fragmentación, el performance bajará y habrá que reorganizar la tabla.

La creación de un índice clúster en una tabla (montón) o la eliminación y nueva creación de un índice clúster existente requiere área de espacio adicional disponible en la base de datos para acomodar la ordenación de datos y una copia temporal de la tabla original o datos del índice clúster existente.

Existen otros 3 tipos de índices que se utilizan para mejorar los tiempos de acceso a datos XML, a búsquedas de texto y de datos espaciales. A continuación se muestra la disponibilidad según la versión de SQL Server:

	Clustered Index	Non-Clustered Index	Full Text Index	Xml Index	Spatial Index
SQL Server 2000	si	Si	-	-	-
SQL Server 2005	si	si (1)	si	si	-
SQL Server 2008	si	si (1 y 2)	si	si	si

Los Clustered Indexes son índices que controlan el orden físico de las filas en la tabla, por lo cual solo puede existir uno para cada tabla.

Los Non-Clustered indexes son índices que mantienen un sub conjunto de las columnas de la tabla en orden. Estos indices no modifican el orden de las filas de la tabla, en lugar de esto mantienen una lista ordenada de referencias a filas de la tabla original.

Para ilustrar la diferencia entre estos 2 tipos de índices podemos decir que las páginas blancas de la guía telefónica tienen un clustered index por Apellido(s) y Nombres, con lo cual puedo buscar de forma muy eficiente el número de teléfono de una persona si conozco sus apellidos y su nombre, una vez que lo encuentro obtendré su número de teléfono en forma inmediata pues el numero está al lado del nombre.

En el caso de las páginas amarillas de la guía telefónica la forma de buscar es un poco distinta, en este caso busco por rubro. Primero busco en un índice, el cual me indica en qué página se encuentra la lista de empresas que satisfacen la condición que busco. Esto mismo es lo que pasa cuando utilizo un índice Non-Clustered index una vez que encuentro lo que quiero en el índice debo ir a leer la fila específica para obtener el resto de los datos.

Veamos qué pasa cuando agregamos un índice a la columna Username de una tabla:

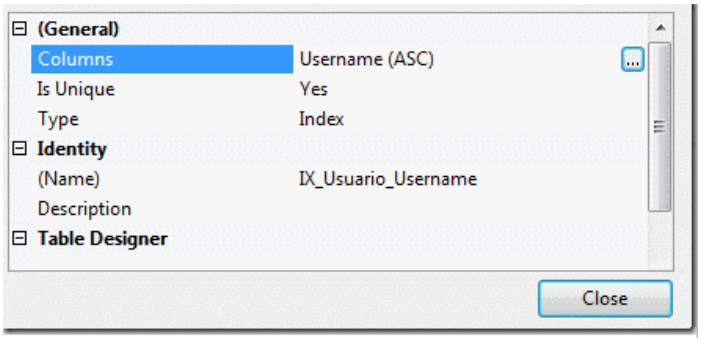


Fig 3.11.Agregando un índice a la columna Username

El índice será Non-Clustered y Único puesto que no podemos tener más de un usuario con el mismo nombre.

Al volver a ejecutar la misma consulta, obtenemos lo siguiente:

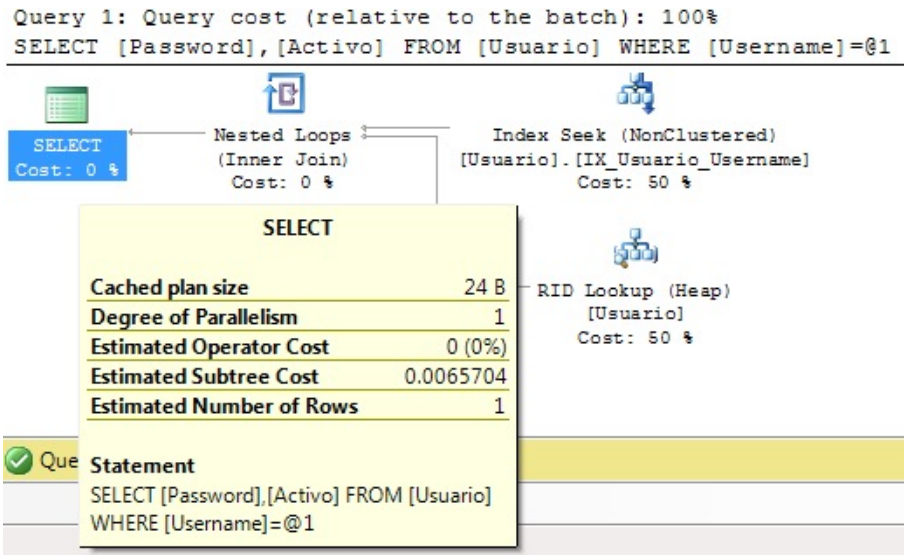


Fig 3.12 Ejecutando de nueva cuenta la consulta

El plan de ejecución es un poco más complicado pero esta consulta es 100 veces menos costosa que la anterior. Ahora la consulta utiliza el índice para encontrar el RID, que es el identificador de la fila, con este RID hace una búsqueda en la tabla de tipo heap (RID Lookup).
Esta reducción de costo se explica por la cantidad de accesos a datos que esta consulta requiere, puesto que estos bajan desde 774 a solo 4 páginas lo que equivale 32 kb de datos contra 6192 kb, que eran necesarios antes de la creación del índice.
Table 'Usuario'. Scan count 0, logical reads 4, physical reads 1, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.
Veamos otra alternativa, si utilizamos un clustered index en lugar de un non-cluster index no será necesario el lookup, por lo que el acceso será más eficiente.

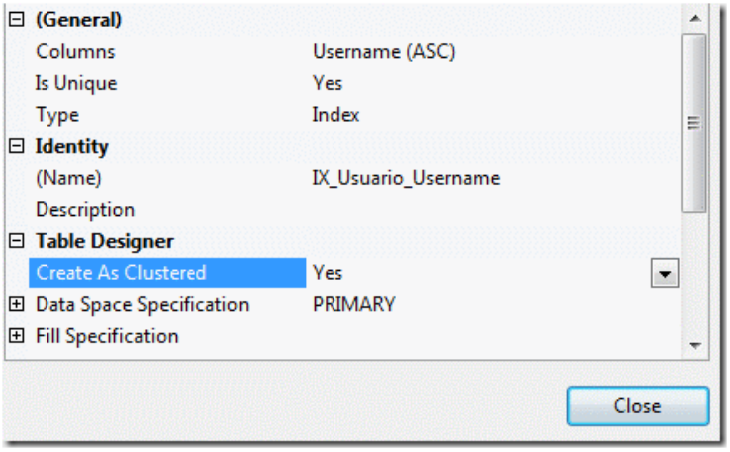


Fig 3.13 Empleando clustered index en lugar de un non-cluster index

Ahora el plan de ejecución se ve así:

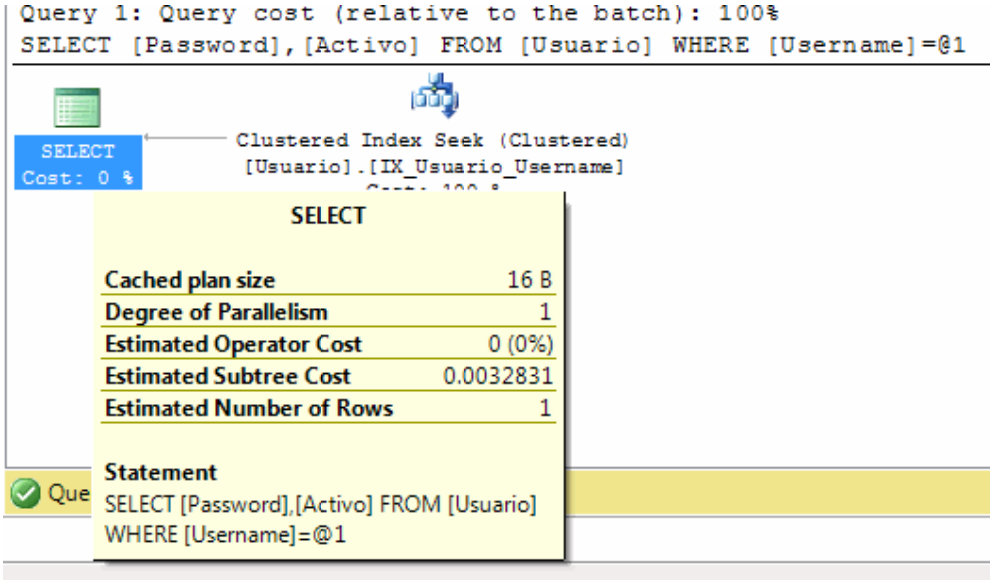


Fig 3.14. Nueva vista al plan de acción

El acceso se ve así:

Table 'Usuario'. Scan count 0, logical reads 3, physical reads 2, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

Como se puede ver el costo de esta consulta ahora es la mitad que con el índice non-clustered y en lugar de 4 lecturas lógicas tenemos sólo 2.

El clustered index controla el orden físico de las filas en la tabla, a diferencia de los índices Non-Clustered que funcionan como una lista ordenada de identificadores de fila.

Todas las tablas que tienen un clustered index tienen un nodo raíz y muchos nodos en los niveles intermedios, estos a su vez pueden apuntar a nodos hojas o a otros nodos intermedios. Esta estructura forma un árbol (B-Tree) que permite encontrar cualquier fila en forma eficiente.

La búsqueda parte desde el nodo raíz, este nodo tiene una lista de llaves, se comparan estas llaves para encontrar el nodo de nivel intermedio que contenga un rango de llaves que cubra la llave que se está buscando. Luego se repite el proceso en los nodos intermedios hasta que se encuentre la página de datos que contenga la fila específica.

Para ilustrar este proceso, realicemos una búsqueda sobre un clustered index de la llave 123.

Contenido del Nodo Raíz (ID 0)

Llave	ID Nodo Intermedio
1	1
100	**2**
160	3
300	4
500	5
1000	6

Contenido del Nodo Intermedio (ID 2)

Llave	ID Nodo Intermedio
100	10
110	11
120	**12**
130	13
140	14
150	15

Contenido del Nodo Hoja (ID 12)

Llave	*Columna 1*	Columna 2	…	Columna N
120	AX06	10000		XXX
121	AX02	10004		XXX
122	AX07	10000		XXX
123	**AX04**	**20000**		**XXX**
124	AX08	9000		XXX
125	AX01	1		XXX

Para obtener el resultado se lee el nodo raíz, se busca el nodo intermedio que contiene a la llave 123, en este caso el nodo 2 contiene filas con llave entre 100 y 160. Luego se repite el proceso, el nodo 12 contiene filas con llaves entre 120 y 130. El nodo 12 es un página de datos por lo cual no necesitamos seguir buscando.

Cada vez que se agrega una fila a la tabla, SQL Server debe insertar la nueva fila en la posición correcta dentro del índice, esto puede ser una operación simple y eficiente si es que la página es la última del índice o si la página tiene espacio disponible. Si la página no tiene espacio es necesario dividir la página en 2, algo conocido como Page Split, que deja 2 páginas con un 50% de utilización.

Cada vez que se elimina una fila de la tabla, SQL Server eliminará la fila de la página, sin modificar ninguna otra página, lo cual limita el impacto de la operación a 1 sola página pero causa que se desperdicie mayor porcentaje de las página, proceso conocido como fragmentación.

Algo aun más costoso ocurre cuando se actualiza el valor de la llave del índice clustered, en este caso SQL Server debe copiar la fila desde la pagina original, aplicar los cambios indicados en el update, insertar la fila en la nueva página y finalmente eliminar la fila desde la página original.

Ejemplo de Cluster Índice en SQL Server

En el ejemplo siguiente se crea un índice en una tabla con particiones (Menor) utilizando la compresión de fila en todas las particiones del índice.

```
CREATE CLUSTERED INDEX IDX_NivelPF_MENOR_NIVEL
ON MENOR (nivel)
WITH ( DATA_COMPRESSION = ROW ) ;
GO
```

En el ejemplo siguiente se crea un índice en una tabla con particiones (menor) utilizando la compresión de página en la partición 1 del índice y la compresión de fila en las particiones 2 a 4 del índice.

```
CREATE CLUSTERED INDEX IDX_NivelPF_MENOR_NIVEL
ON MENOR (nivel)
WITH (DATA_COMPRESSION = PAGE ON PARTITIONS(1),
   DATA_COMPRESSION = ROW ON PARTITIONS (2 TO 4 ) ) ;
GO
```

Ejemplo para modificar un Índice en SQL Server

Para modificar un índice existente de una tabla o una vista (relacional o XML) mediante su deshabilitación, regeneración o reorganización, o mediante el establecimiento de opciones en él.

Parámetros generales para Índices en SQL Server:

```
ALTER INDEX { index_name | ALL }
   ON <object>
   { REBUILD
      [ [ WITH ( <rebuild_index_option> [ ,...n ] ) ]
        | [ PARTITION = partition_number
            [ WITH ( <single_partition_rebuild_index_option>
                  [ ,...n ] )
            ]
         ]
      ]
   | DISABLE
   | REORGANIZE
      [ PARTITION = partition_number ]
      [ WITH ( LOB_COMPACTION = { ON | OFF } ) ]
   | SET ( <set_index_option> [ ,...n ] )
   }
[ ; ]

<object> ::=
{
   [ database_name. [ schema_name ] . | schema_name. ]
      table_or_view_name
}

<rebuild_index_option > ::=
{
   PAD_INDEX  = { ON | OFF }
 | FILLFACTOR = fillfactor
```

```
  | SORT_IN_TEMPDB = { ON | OFF }
  | IGNORE_DUP_KEY = { ON | OFF }
  | STATISTICS_NORECOMPUTE = { ON | OFF }
  | ONLINE = { ON | OFF }
  | ALLOW_ROW_LOCKS = { ON | OFF }
  | ALLOW_PAGE_LOCKS = { ON | OFF }
  | MAXDOP = max_degree_of_parallelism
}

<single_partition_rebuild_index_option> ::=
{
   SORT_IN_TEMPDB = { ON | OFF }
  | MAXDOP = max_degree_of_parallelism
}

<set_index_option>::=
{
   ALLOW_ROW_LOCKS= { ON | OFF }
  | ALLOW_PAGE_LOCKS = { ON | OFF }
  | IGNORE_DUP_KEY = { ON | OFF }
  | STATISTICS_NORECOMPUTE = { ON | OFF }
}
```

Ejemplo:

(datamart dieta)

```
CREATE INDEX nivel_idx
ON MENORR (nivel)
```

Command(s) completed successfully.

Reconstrucción de un índice en SQL Server

```
ALTER INDEX nivel_idx
ON MENOR
REBUILD
PARTITION NivelPF (2) ;
```

Command(s) completed successfully.

No es posible utilizar ALTER INDEX para volver a crear particiones en un índice o moverlo a un grupo de archivos distinto. No es posible utilizar esta instrucción para modificar la definición de índice, como por ejemplo para agregar o eliminar columnas o cambiar su orden. Utilice CREATE INDEX con la cláusula DROP_EXISTING para realizar estas operaciones.

3.4 Construcción de DOLAP

A continuación vamos a proceder a construir nuestro Datamart Dieta en base al diagrama que diseñamos anteriormente.

Para esto lo primero que tenemos que hacer es construir nuestra base de datos en **SQLServer 2005** como se muestra a continuación.

Una vez abierto nuestro entorno de SQL Server Management Studio.

Creamos una nueva base hacemos clic derecho sobre la carpeta **databases** en **New Databases**:

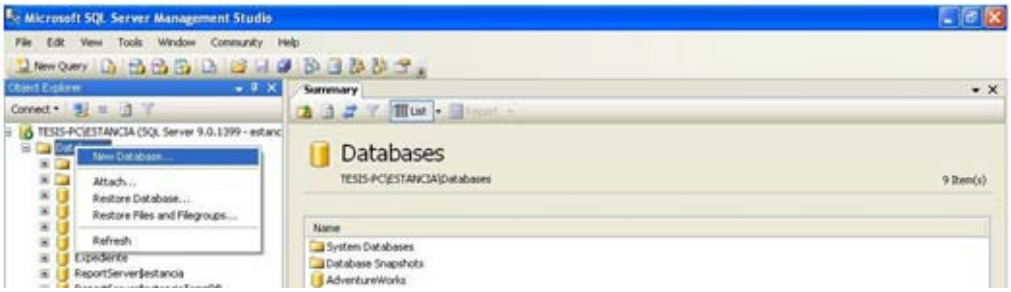


Fig 3.15 Pantalla "New Database"

Asignamos el nombre a la base de datos y OK.

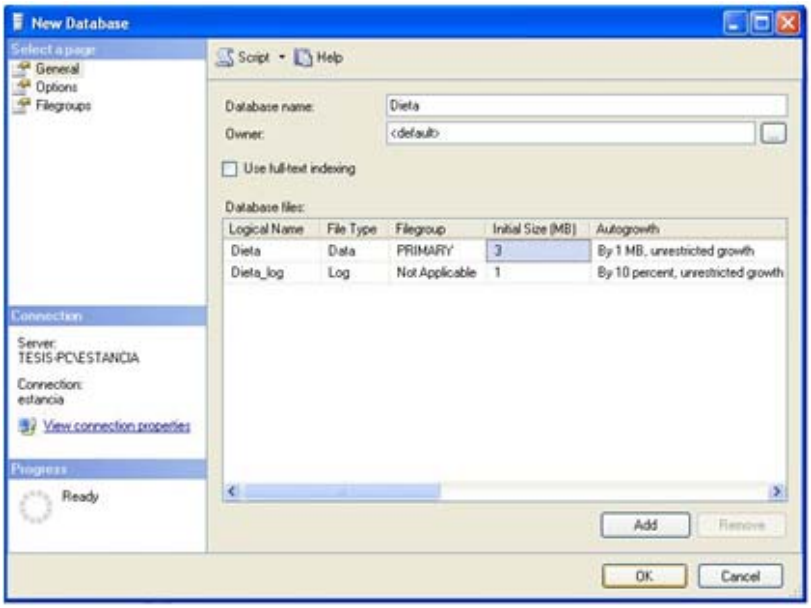


Fig 3.16. Pantalla "Asignar nombre"

Procedemos a la construcción de la base de datos de acuerdo al diagrama siguiente:

Una vez construido en Erwin procedemos a obtener su script y ejecutarlo en nuestra base de datos:

SQL Server
Script de creación de objetos para la base de datos Dieta

```
CREATE TABLE ALIMENTOS (
     clave              int IDENTITY (1,1) NOT FOR REPLICATION,
     tipo_alimento      numeric(1) NULL,
     id_original        numeric(1) NULL,
     des_com            char(1) NULL
)
go
ALTER TABLE ALIMENTOS
     ADD CONSTRAINT PK_ALIMENTO PRIMARY KEY (clave)
go
CREATE TABLE ALIMENTOS_ORIGINAL (
     id_original        numeric(2) NOT NULL,
     tipo_alimento      numeric(1) NOT NULL,
     nombre             varchar(18) NULL
)
go
ALTER TABLE ALIMENTOS_ORIGINAL
     ADD CONSTRAINT PK_ALIMENTO_ORIGINAL PRIMARY KEY (id_original,
          tipo_alimento)
go
CREATE TABLE AÑOS (
     año                numeric(4) NOT NULL
)
go
ALTER TABLE AÑOS
     ADD CONSTRAINT PK_AÑO PRIMARY KEY (año)
go
CREATE TABLE DES_COM (
     des_com            char(1) NOT NULL,
     descripcion        varchar(10) NULL
)
go
ALTER TABLE DES_COM
     ADD CONSTRAINT PK_DES_COM PRIMARY KEY (des_com)
go
CREATE TABLE DIAS (
     num_mes            numeric(2) NOT NULL,
     num_sem            numeric(1) NOT NULL,
     año                numeric(4) NOT NULL,
     fecha              datetime NOT NULL
)
go
ALTER TABLE DIAS
     ADD CONSTRAINT PK_DIA PRIMARY KEY (num_mes, num_sem, año,
          fecha)
go
CREATE TABLE DIETAS (
     clave              int NOT NULL,
     curp_infante       char(18) NOT NULL,
     nivel              numeric(2) NOT NULL,
     num_mes            numeric(2) NOT NULL,
     num_sem            numeric(1) NOT NULL,
     año                numeric(4) NOT NULL,
     fecha              datetime NOT NULL,
     consumo            char(2)  NULL
)
go

ALTER TABLE DIETAS
     ADD CONSTRAINT PK_DIETA PRIMARY KEY (clave, curp_infante, num_mes,
```

```
            num_sem, año, fecha,nivel)
go

CREATE TABLE MENORES (
    curp_infante        char(18) NOT NULL,
    nivel               numeric(2) NOT NULL,
    sexo                char(1) NOT NULL
)
go
ALTER TABLE MENORES
    ADD CONSTRAINT PK_MENOR PRIMARY KEY (curp_infante, nivel)
go
CREATE TABLE MESES (
    num_sem             numeric(1) NOT NULL,
    año                 numeric(4) NOT NULL,
    num_mes             numeric(2) NOT NULL,
    nombre              varchar(20) NOT NULL
)
go
ALTER TABLE MESES
    ADD CONSTRAINT PK_MES PRIMARY KEY (num_sem, año, num_mes)
go
CREATE TABLE NIVELES (
    nivel               numeric(2) NOT NULL,
    nombre              varchar(50) NOT NULL
)
go
ALTER TABLE NIVELES
    ADD CONSTRAINT PK_NIVEL PRIMARY KEY (nivel)
go
CREATE TABLE SEMESTRES (
    año                 numeric(4) NOT NULL,
    num_sem             numeric(1) NOT NULL,
    nombre              varchar(20) NULL
)
go
ALTER TABLE SEMESTRES
    ADD CONSTRAINT PK_SEMESTRE PRIMARY KEY (año, num_sem)
go
CREATE TABLE TIPOS_ALIMENTOS (
    tipo_alimento       numeric(1) NOT NULL,
    descripcion         varchar(30) NULL
)
go
ALTER TABLE TIPOS_ALIMENTO
    ADD CONSTRAINT PK_TIPO_ALIMENTO PRIMARY KEY (tipo_alimento)
go
ALTER TABLE ALIMENTOS
    ADD CONSTRAINT FK_ALIMENTO
        FOREIGN KEY (des_com)
                REFERENCES DES_COM  (des_com)
go
ALTER TABLE ALIMENTOS
    ADD CONSTRAINT FK_ALIMENTO_ALIMENTO_ORIGINAL
        FOREIGN KEY (id_original, tipo_alimento)
                REFERENCES ALIMENTOS_ORIGINAL  (
        id_original, tipo_alimento)
go
ALTER TABLE ALIMENTOS_ORIGINAL
    ADD CONSTRAINT ALIMENTO_ORIGINAL_TIPO_ALIMENTO
        FOREIGN KEY (tipo_alimento)
                REFERENCES TIPOS_ALIMENTO  (tipo_alimento)
go
ALTER TABLE DIAS
    ADD CONSTRAINT FK_MES_DIA
        FOREIGN KEY (num_sem, año, num_mes)
```

```
                    REFERENCES MESES  (num_sem, año, num_mes)
go
ALTER TABLE DIETAS
    ADD CONSTRAINT FK_DIETA_DIA
        FOREIGN KEY (num_mes, num_sem, año, fecha)
                    REFERENCES DIAS  (num_mes, num_sem, año,
        fecha)
go
ALTER TABLE DIETAS
    ADD CONSTRAINT FK_ALIMENTO_DIETA
        FOREIGN KEY (clave)
                    REFERENCES ALIMENTOS  (clave)
go
ALTER TABLE DIETAS
    ADD CONSTRAINT FK_MENOR_DIETA
        FOREIGN KEY (curp_infante)
                    REFERENCES MENORES  (curp_infante, nivel)
go
ALTER TABLE MENORES
    ADD CONSTRAINT FK_NIVEL_MENOR
        FOREIGN KEY (nivel)
                    REFERENCES NIVELES  (nivel)
go
ALTER TABLE MESES
    ADD CONSTRAINT FK_SEMESTRE_MES
        FOREIGN KEY (año, num_sem)
                    REFERENCES SEMESTRES  (año, num_sem)
go
ALTER TABLE SEMESTRES
    ADD CONSTRAINT FK_AÑO_SEMESTRE
        FOREIGN KEY (año)
                   REFERENCES AÑOS  (año)
go
CREATE TABLE CONSUMOS (
    consumo       char(2) NOT NULL,
    descripcion    varchar(20) NULL
)
go
ALTER TABLE CONSUMOS
  ADD CONSTRAINT PK_CONSUMOS PRIMARY KEY (consumo)
go
ALTER TABLE DIETAS
    ADD CONSTRAINT FK_CONSUMO
        FOREIGN KEY (consumo)
                    REFERENCES CONSUMOS  (consumo)
go
```

Un paso previo para la carga del datamart Dieta, fueron necesarias crear algunos objetos auxiliares al diseño original:

```
CREATE TABLE INTERSECCION_ALIMENTO (
    id_alimento        SMALLINT NOT NULL,
    tipo_alimento      NUMBER(1) NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_INTERSECCION_ALIMENTOS ON INTERSECCION_ALIMENTO
(
    id_alimento                ASC
);
ALTER TABLE INTERSECCION_ALIMENTO
    ADD  ( CONSTRAINT PK_INTERSECCION_ALIMENTO PRIMARY KEY (id_alimento) ) ;
```

```
CREATE TABLE INTERSECCION_CEREAL (
    id_cereal          SMALLINT NOT NULL,
    tipo_alimento      NUMBER(1) NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_INTERSECCION_CEREAL ON INTERSECCION_CEREAL
(
    id_cereal                 ASC
);

ALTER TABLE INTERSECCION_CEREAL
    ADD  ( CONSTRAINT PK_INTERSECCION_CEREAL PRIMARY KEY (id_cereal) ) ;

CREATE TABLE INTERSECCION_FRUTA (
    id_fruta           SMALLINT NOT NULL,
    tipo_alimento      NUMBER(1) NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_NTERSECCION_FRUTA ON INTERSECCION_FRUTA
(
    id_fruta                  ASC
);

ALTER TABLE INTERSECCION_FRUTA
    ADD  ( CONSTRAINT PK_NTERSECCION_FRUTA PRIMARY KEY (id_fruta) ) ;

CREATE TABLE INTERSECCION_JUGO (
    id_jugo            SMALLINT NOT NULL,
    tipo_alimento      NUMBER(1) NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_INTERSECCION_JUGO ON INTERSECCION_JUGO
(
    id_jugo                   ASC
);

ALTER TABLE INTERSECCION_JUGO
    ADD  ( CONSTRAINT PK_INTERSECCION_JUGO PRIMARY KEY (id_jugo) ) ;
CREATE TABLE INTERSECCION_LECHE (
    id_leche           SMALLINT NOT NULL,
    tipo_alimento      NUMBER(1) NOT NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_INTERSECCION_LECHE ON INTERSECCION_LECHE
(
    id_leche                  ASC
);

ALTER TABLE INTERSECCION_LECHE
    ADD  ( CONSTRAINT PK_INTERSECCION_LECHE PRIMARY KEY (id_leche) ) ;
CREATE TABLE INTERSECCION_SOPA (
    id_sopa            SMALLINT NOT NULL,
    tipo_alimento      NUMBER(1) NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_INTERSECCION_SOPA ON INTERSECCION_SOPA
(
    id_sopa                   ASC
);
ALTER TABLE INTERSECCION_SOPA
```

```
    ADD  ( CONSTRAINT PK_INTERSECCION_SOPA PRIMARY KEY (id_sopa) ) ;

CREATE TABLE INTERSECCION_VERDURA (
    id_verdura_ensalada  SMALLINT NOT NULL,
    tipo_alimento       NUMBER(1) NULL
)
            TABLESPACE TS_CATALOGOS
 ;
CREATE UNIQUE INDEX PK_INTERSECCION_VERDURA ON INTERSECCION_VERDURA
(
    id_verdura_ensalada          ASC
);


ALTER TABLE INTERSECCION_VERDURA
    ADD  ( CONSTRAINT PK_INTERSECCION_VERDURA PRIMARY KEY (id_verdura_ensalada) ) ;
ALTER TABLE INTERSECCION_ALIMENTO
    ADD  ( CONSTRAINT FK_INTERSECCION_ALIMENTO_TIPOS
        FOREIGN KEY (tipo_alimento)
                REFERENCES TIPOS_ALIMENTOS
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_ALIMENTO
    ADD  ( CONSTRAINT FK_ALIMENTOS_INTERSECCION_ALIM
        FOREIGN KEY (id_alimento)
                REFERENCES ALIMENTOS_FUERTES
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_CEREAL
    ADD  ( CONSTRAINT FK_INTERSECCION_CEREAL_TIPOS
        FOREIGN KEY (tipo_alimento)
                REFERENCES TIPOS_ALIMENTOS
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_CEREAL
    ADD  ( CONSTRAINT FK_CEREALES_INTERECCION_CEREAL
        FOREIGN KEY (id_cereal)
                REFERENCES CEREALES_LEGUMINOSAS
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_FRUTA
    ADD  ( CONSTRAINT FK_INTERSECCION_FRUTA_TIPOS
        FOREIGN KEY (tipo_alimento)
                REFERENCES TIPOS_ALIMENTOS
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_FRUTA
    ADD  ( CONSTRAINT FK_FRUTAS_INTERSECCION_FRUTA
        FOREIGN KEY (id_fruta)
                REFERENCES FRUTAS_YOGOURTH_POSTRES
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_JUGO
    ADD  ( CONSTRAINT FK_INTERSECCION_JUGO_TIPOS
        FOREIGN KEY (tipo_alimento)
                REFERENCES TIPOS_ALIMENTOS
                ON DELETE SET NULL ) ;


ALTER TABLE INTERSECCION_JUGO
    ADD  ( CONSTRAINT FK_JUGOS_INTERSECCION_JUGO
        FOREIGN KEY (id_jugo)
                REFERENCES JUGOS_AGUAS
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_LECHE
    ADD  ( CONSTRAINT FK_INTERSECCION_LECHE_TIPOS
        FOREIGN KEY (tipo_alimento)
                REFERENCES TIPOS_ALIMENTOS
                ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_LECHE
    ADD  ( CONSTRAINT FK_VARIACIONES_INTERSECCION_LE
        FOREIGN KEY (id_leche)
```

```
                    REFERENCES VARIACIONES_LECHE
                    ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_SOPA
    ADD  ( CONSTRAINT FK_INTERSECCION_SOPA_TIPOS
        FOREIGN KEY (tipo_alimento)
                    REFERENCES TIPOS_ALIMENTOS
                    ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_SOPA
    ADD  ( CONSTRAINT FK_SOPAS_INTERSECCION_SOPA
        FOREIGN KEY (id_sopa)
                    REFERENCES SOPAS
                    ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_VERDURA
    ADD  ( CONSTRAINT FK_INTERSECCION_VERDURA_TIPOS
        FOREIGN KEY (tipo_alimento)
                    REFERENCES TIPOS_ALIMENTOS
                    ON DELETE SET NULL ) ;
ALTER TABLE INTERSECCION_VERDURA
    ADD  ( CONSTRAINT FK_VERDURAS_INTERSECCION_VERDU
        FOREIGN KEY (id_verdura_ensalada)
                    REFERENCES VERDURAS_ENSALADAS
                    ON DELETE SET NULL ) ;
```

La siguiente tabla nos servirá como tabla auxiliar para poder cargar la tabla de hechos en nuestro Datamart Dieta.

```
CREATE TABLE ALIMENTO (
    clave             NUMBER(1) NOT NULL,
    tipo_alimento      NUMBER(1) NULL,
    id_original       NUMBER(2) NULL,
    des_com            CHAR(1) NULL
)
          TABLESPACE TS_CATALOGOS
 ;
ALTER TABLE ALIMENTO
    ADD  ( CONSTRAINT PK_ALIMENTO PRIMARY KEY (clave) ) ;
```

3.5 ETL (Extract, Transform, Load)

La migración de los datos desde las fuentes operacionales al DataWarehouse requiere la necesidad de procesos para extraer, transformar y cargar los datos, actividad que se conoce como ETL.
Estos procesos se originan debido a la necesidad de reformatear, conciliar y limpiar los datos de origen.
La mayoría de los datos de origen son los datos operacionales actuales, aunque parte de ellos pueden ser datos históricos archivados.

Carga inicial de Datos para el Datamart Dieta
Una vez creados todos nuestros objetos de nuestra base de datos **Dieta**, procedemos a cargar los datos con ayuda de Pentaho.

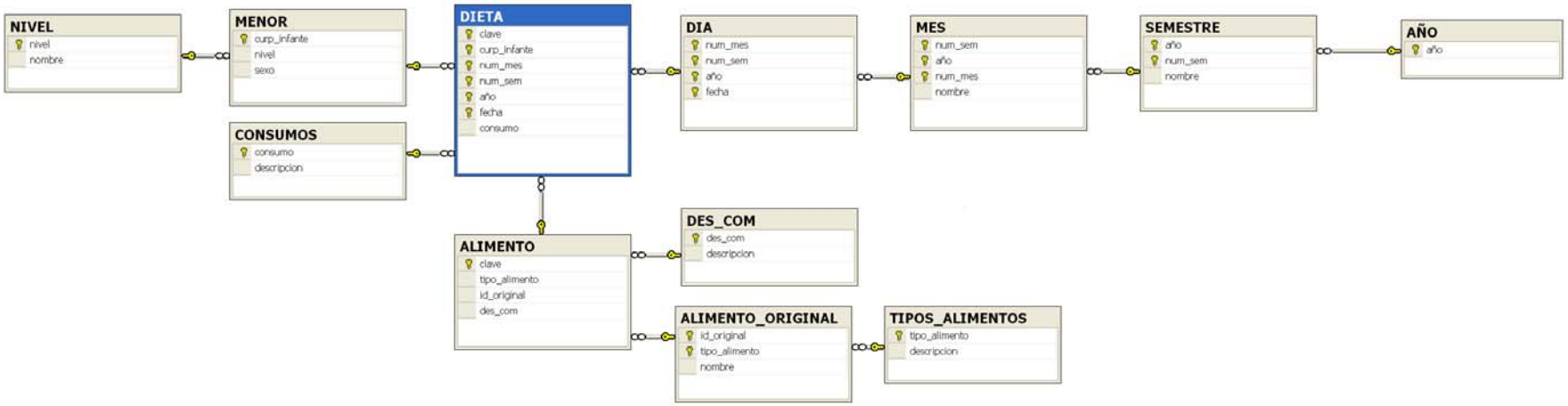


Fig 3.18. Estructura del Datamart Dieta en SQL Server

Para esto hay que definir las conexiones necesarias, en este caso una para Oracle, como lo hemos hecho anteriormente, y otra para SQLServer.

Conexión con Oracle

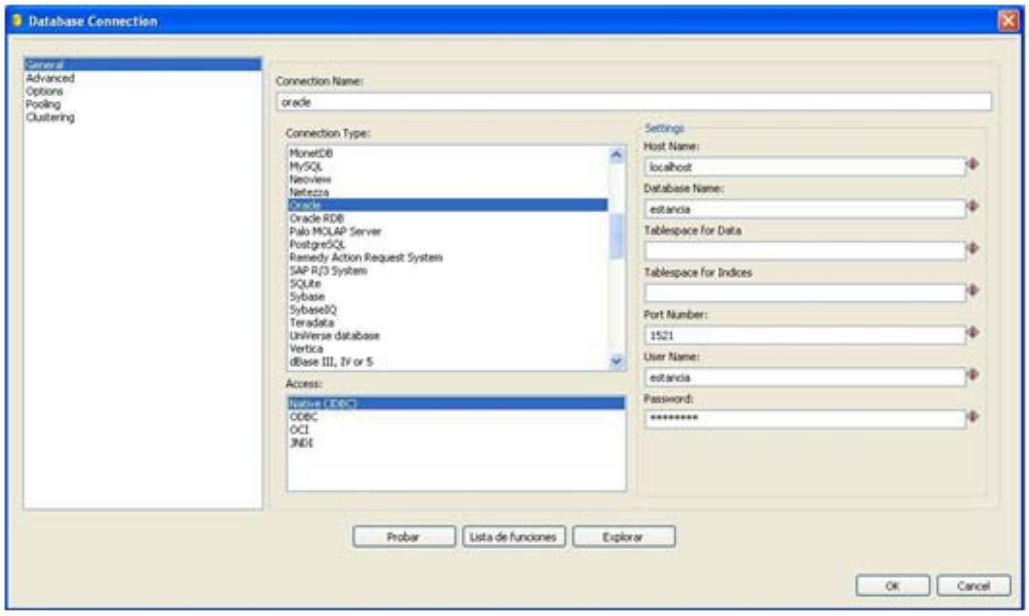


Fig 3.19. Configuración de la conexión en Oracle

Conexión con SqlServer

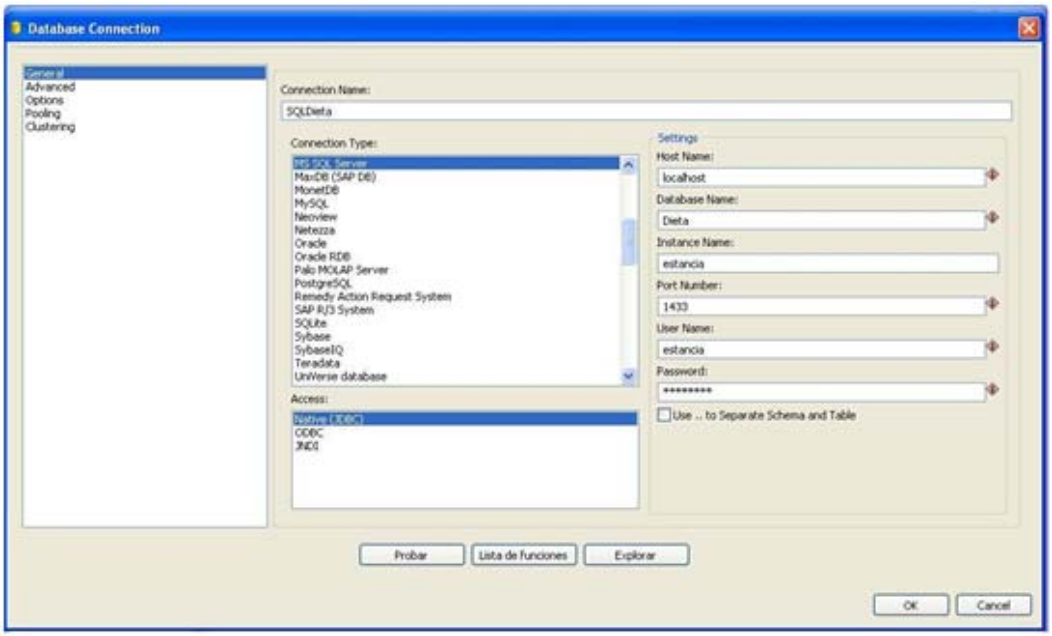


Fig 3.20 Configuración de la conexión en SQL Server

Ahora definimos una nueva transformación, en la que la entrada será una tabla (proveniente de la Base de Datos OLTP llamada ESTANCIA en Oracle) y la salida otra tabla (en la Base de Datos OLAP llamada DIETA o EXPEDIENTE en SQLServer).

Transformación para la carga de datos:

Fig 3.21 Visualización de una transformación en Pentaho

Aquí, la tabla de entrada tendrá asignada la conexión Oracle, y en el recuadro escribimos la sentencia que obtendrá los datos requeridos de nuestro OLTP.

Fig 3.22 Configuración de la tabla entrada en Pentaho

En la salida asignamos la conexión correspondiente a nuestra base de datos OLAP en SQLServer, y el nombre de la Tabla destino.

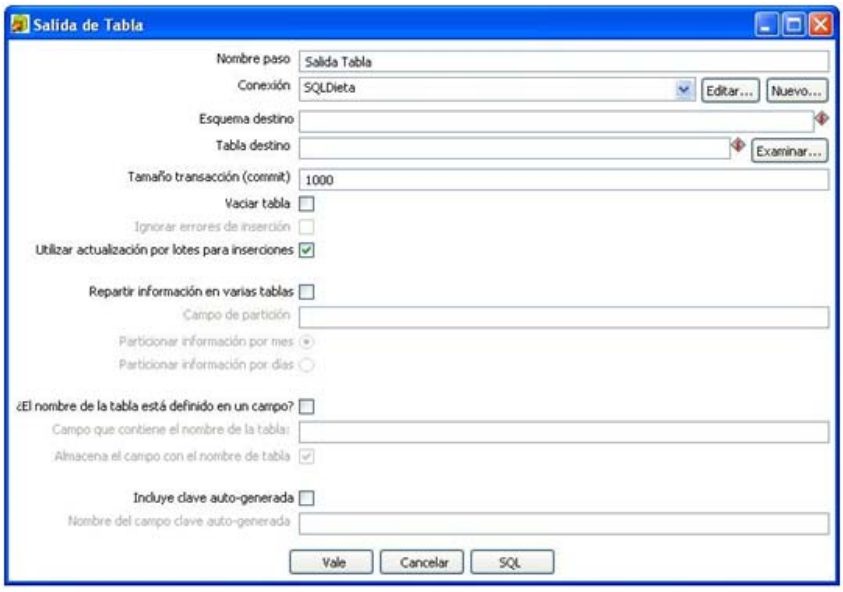


Fig 3.23. Configuración de la tabla salida en Pentaho

A continuación se muestran las sentencias que utilizamos para cargar cada tabla de nuestro datamart DIETA.

Para la Dimensión INFANTE:

NIVEL

```
SELECT nivel,nombre FROM niveles;
```

MENOR

```
SELECT Ifn.curp_infante,Ifn.nivel,I.sexo FROM infantes_nivel Ifn, infantes I where
Ifn.curp_infante=I.curp_infante;
```

*Para la Dimensión **TIEMPO**:*

AÑO

```
SELECT distinct to_char(FECHA, 'yyyy') as año from consumo_comidas
```

SEMESTRE

```
SELECT distinct to_char(FECHA, 'yyyy') as "año",
case
when to_char(FECHA, 'MM') <= 6 then '1'
when to_char(FECHA, 'MM') > 6 then '2'
else ''
end "num_sem",
case
when to_char(FECHA, 'MM') <= 6 then 'Primer'
when to_char(FECHA, 'MM') > 6 then 'Segundo'
else ''
end "nombre"
from consumo_comidas order by "año","num_sem"
```

MES

```
SELECT distinct to_char(FECHA, 'yyyy') as "año",
case
when to_char(FECHA, 'MM') <= 6 then '1'
when to_char(FECHA, 'MM') > 6 then '2'
else ''
end "num_sem",
to_char(FECHA, 'MM') as "num_mes",
case
when to_char(FECHA, 'MM') = 1 then 'enero'
when to_char(FECHA, 'MM') = 2 then 'febrero'
when to_char(FECHA, 'MM') = 3 then 'marzo'
when to_char(FECHA, 'MM') = 4 then 'abril'
when to_char(FECHA, 'MM') = 5 then 'mayo'
when to_char(FECHA, 'MM') = 6 then 'junio'
when to_char(FECHA, 'MM') = 7 then 'julio'
when to_char(FECHA, 'MM') = 8 then 'agosto'
when to_char(FECHA, 'MM') = 9 then 'septiembre'
when to_char(FECHA, 'MM') = 10 then 'octubre'
when to_char(FECHA, 'MM') = 11 then 'noviembre'
when to_char(FECHA, 'MM') = 12 then 'diciembre'
else ''
end "nombre"
from consumo_comidas order by "año","num_sem","num_mes"
```

DIA

```
SELECT distinct to_char(FECHA, 'yyyy') as "año",
case
when to_char(FECHA, 'MM') <= 6 then '1'
when to_char(FECHA, 'MM') > 6 then '2'
else ''
end "num_sem",
```

```
to_char(FECHA, 'MM') as "num_mes",
fecha
from consumo_comidas order by "año","num_sem","num_mes"
```

Para la Dimensión CONSUMO:

CONSUMO

Esta tabla también es una fuente externa, compuesta por la siguente tabla:

consumo	descripcion
NC	No Comio
PR	Probo
CM	Comio Mitad
CT	Comio Todo

Dicha tabla fue cargada con Pentaho pero esta vez tomando como entrada una tabla de Excel;

Para la Dimensión ALIMENTO:

DES_COM

Este catalogo se trata de una fuente externa, compuesta por la siguiente tabla:

des_com	descripción
d	desayuno
c	comida

Las siguientes tablas requieren de un cambio que no existe en nuestro **OLTP,** así que agregamos las tablas necesarias para relacionar cada alimento con su tipo de alimento, el script y diagrama se muestran más adelante.

TIPOS_ALIMENTOS

```
SELECT tipo_alimento, descripcion from tipos_alimentos
```

ALIMENTOS_ORIGINAL

```
SELECT ij.id_jugo as id_original,ij.tipo_alimento, ja.nombre
FROM tipos_alimentos ta, jugos_aguas ja, interseccion_jugo ij
WHERE  ta.tipo_alimento = ij.tipo_alimento AND ja.id_jugo = ij.id_jugo
UNION
SELECT ift.id_fruta as id_original,ift.tipo_alimento, f.nombre
FROM tipos_alimentos ta, frutas_yogourth_postres  f, interseccion_fruta ift
WHERE  ta.tipo_alimento = ift.tipo_alimento AND f.id_fruta = ift.id_fruta
UNION
SELECT ia.id_alimento as id_original,ia.tipo_alimento, a.nombre
FROM tipos_alimentos ta, alimentos_fuertes   a, INTERSECCION_ALIMENTO ia
WHERE  ta.tipo_alimento = ia.tipo_alimento AND a.id_alimento = ia.id_alimento
UNION
SELECT ic.id_cereal as id_original,ic.tipo_alimento, c.nombre
FROM tipos_alimentos ta, cereales_leguminosas c, INTERSECCION_CEREAL ic
WHERE  ta.tipo_alimento = ic.tipo_alimento AND c.id_cereal = ic.id_cereal
UNION
SELECT iso.id_sopa as id_original,iso.tipo_alimento, s.descripcion as nombre
FROM tipos_alimentos ta, sopas s, INTERSECCION_SOPA iso
WHERE  ta.tipo_alimento = iso.tipo_alimento AND s.id_sopa = iso.id_sopa
```

```
UNION
SELECT iv.id_verdura_ensalada as id_original,iv.tipo_alimento, v.nombre
FROM tipos_alimentos ta, verduras_ensaladas v, INTERSECCION_VERDURA iv
WHERE  ta.tipo_alimento = iv.tipo_alimento AND v.id_verdura_ensalada =
iv.id_verdura_ensalada
UNION
SELECT il.id_leche as id_original,il.tipo_alimento, l.descripcion as nombre
FROM tipos_alimentos ta, variaciones_leche l, INTERSECCION_LECHE il
WHERE  ta.tipo_alimento = il.tipo_alimento AND l.id_leche = il.id_leche
```

ALIMENTO

Esta tabla no necesita información del OLTP así que, la cargamos directamente con sentencias SQL dentro del manejador SQLServer.

```
INSERT INTO ALIMENTO
SELECT ao.tipo_alimento,ao.id_original, dc.des_com
FROM ALIMENTO_ORIGINAL ao,DES_COM dc
WHERE ao.tipo_alimento<5 AND DC.DES_COM='d';

INSERT INTO ALIMENTO
SELECT ao.tipo_alimento,ao.id_original, dc.des_com
FROM ALIMENTO_ORIGINAL ao,DES_COM dc
WHERE ao.tipo_alimento=7 AND DC.DES_COM='d';

INSERT INTO ALIMENTO
SELECT ao.tipo_alimento,ao.id_original, dc.des_com
FROM ALIMENTO_ORIGINAL ao,DES_COM dc
WHERE ao.tipo_alimento<7 AND DC.DES_COM='c';
```

TABLA DE HECHOS:

DIETA

```
SELECT a.clave, cd.curp_infante, cd.nivel, to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_jugo as consumo
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 1 AND A.id_original = cd.id_jugo AND A.des_com = 'd'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_fruta as consumo
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 2 AND A.id_original = cd.id_fruta AND A.des_com = 'd'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
```

```
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_alimento as consumo
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 3 AND A.id_original = cd.id_alimento AND A.des_com = 'd'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_cereal as consumo
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 4 AND A.id_original = cd.id_cereal AND A.des_com = 'd'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_leche as consumo
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 7 AND A.id_original = cd.id_leche AND A.des_com = 'd'

UNION

SELECT a.clave, cd.curp_infante, cd.nivel, to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_jugo as consumo
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 1 AND A.id_original = cd.id_jugo AND A.des_com = 'c'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_fruta as consumo
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 2 AND A.id_original = cd.id_fruta AND A.des_com = 'c'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_alimento as consumo
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 3 AND A.id_original = cd.id_alimento AND A.des_com = 'c'
```

```
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_cereal as consumo
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 4 AND A.id_original = cd.id_cereal AND A.des_com = 'c'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_sopa as consumo
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 5 AND A.id_original = cd.id_sopa AND A.des_com = 'c'
UNION

SELECT a.clave, cd.curp_infante, cd.nivel,to_char(cd.FECHA, 'MM') as "num_mes",
case
when to_char(cd.FECHA, 'MM') <= 6 then '1'
when to_char(cd.FECHA, 'MM') > 6 then '2'
else ''
end "num_sem", to_char(cd.FECHA, 'yyyy') as "año",cd.FECHA,cd.status_verdura_ensalada as
consumo
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 6 AND A.id_original = cd.id_verdura_ensalada AND A.des_com =
'c'
```

Transformaciones de Infantes para el Datamart

En ocasiones los datos con los que necesitamos trabajar, no están de la forma en que nosotros desearíamos que estuvieran, es por eso que dichos datos deben someterse a un proceso de transformación.

El software "Pentaho" cuenta con una serie de utilidades con las cueles podemos realizar estas transformaciones en nuestros datos:

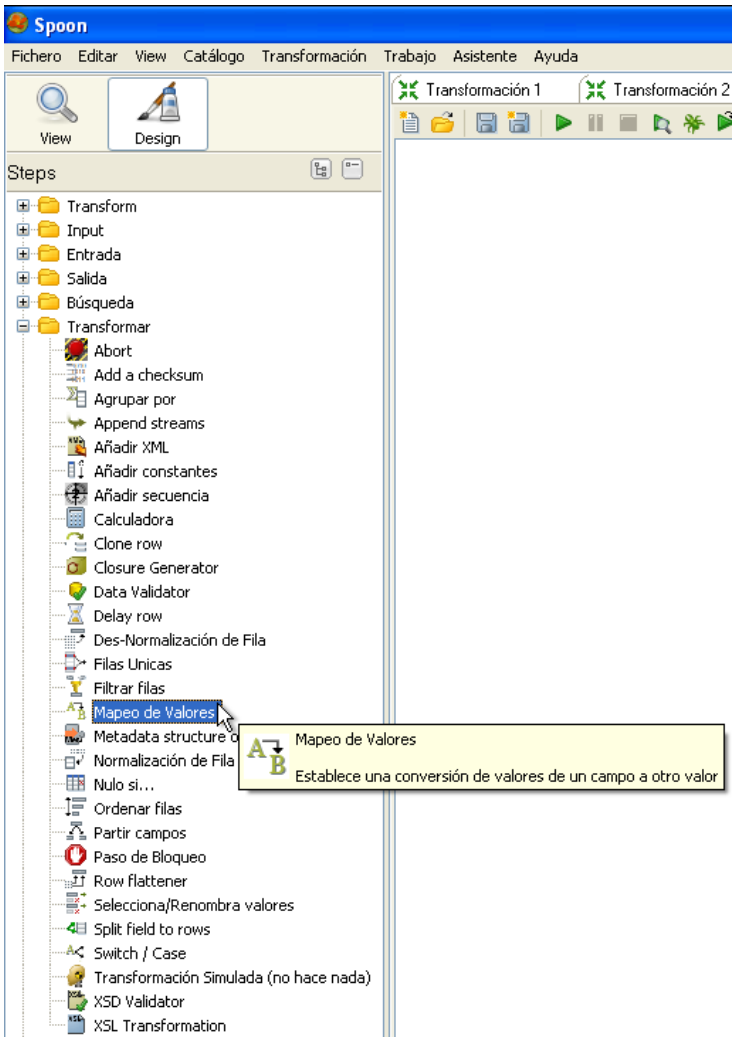


Fig 3.24. Diversas opciones de transformaciones en Pentaho

Para ejemplificar concretamente una de las transformaciones que se utilizan con mayor frecuencia, realizaremos un "*mapeo de valores*", empleando la función de Pentaho con el mismo nombre:

Fig 3.25. Icono de Mapeo de Valores

Lo que se pretende hacer es mapear los valores de la tabla "infantes" que está en Oracle para posteriormente pasar dichos datos a otra tabla del manejador SQL Server.
La tabla origen contiene una columna llamada "género" la cual se llenó erróneamente de datos en diferentes formatos para su representación (m, f, h, A, B, M, F) así que para pasarlos a la base de datos

en SQL server y mantener un solo formato era necesario tenerlos con el formato F de femenino y M de masculino como se muestra a continuación

m transformar a **F**
h transformar a **M**
A transformar a **M**
B transformar a **F**

Para realizar el mapeo, comenzaremos con "jalar" el ícono "Mapeo de valores" al área de trabajo y enlazarlo entre la entrada y salida de datos:

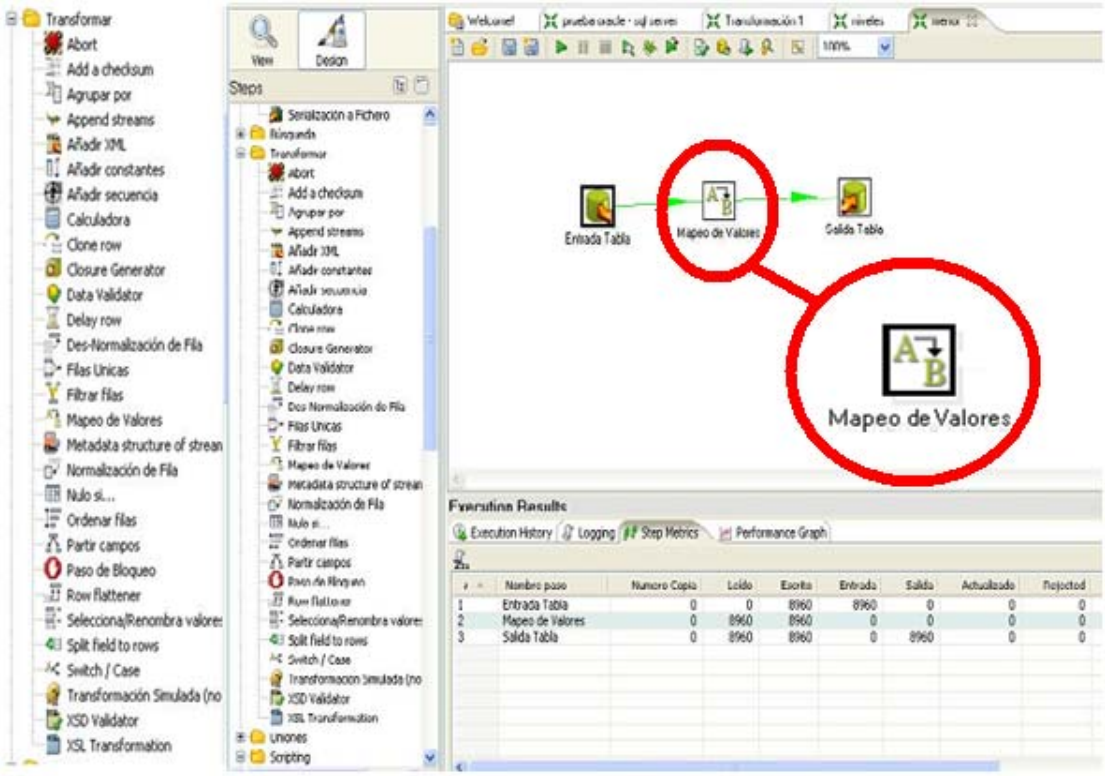


Fig 3.26. Visualización de un mapeo en Pentaho

Posteriormente daremos doble clic sobre dicho ícono para configurar las características del mapeo:

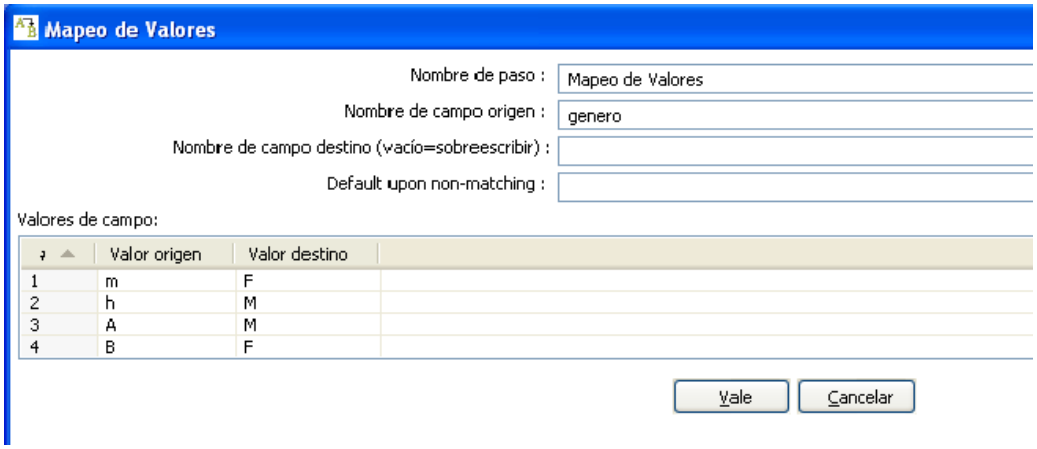


Fig 3.27 Configurando las características del mapeo de valores

Finalmente realizaremos la carga de los datos:

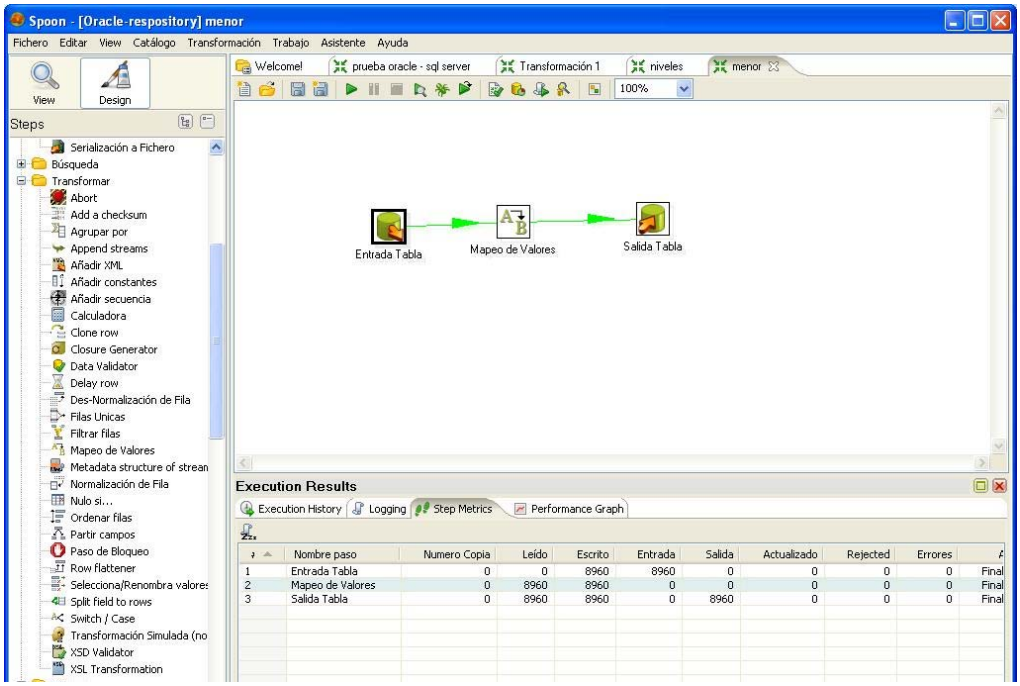


Fig 3.28 Ejecución de la trasformación de valores

Carga incremental

Una vez que el Data Warehouse está cargado, hay que desarrollar otro proceso para la carga incremental, que se ejecutará mensual, semanal o diariamente.
Extraer Deltas permite extraer registros nuevos o registros que contengan valores que cambiaron desde la última carga realizada.

Diseñar programas ETL para extracciones delta es más fácil cuando las fuentes consisten en Bases de Datos relacionales y contamos con una columna ***timestamp*** para determinar los deltas.
Si los datos residen en archivos planos sin timestamp el proceso se vuelve más complejo y puede requerir la lectura de archivos de auditoría para determinar los registros que cambiaron.
Otra alternativa es sugerir al equipo de desarrollo de los sistemas operacionales que agregue un timestamp, aunque en la mayoría de los casos los administradores no estarán de acuerdo porque cualquier cambio en la estructura de la base operacional requerirá cambios en los programas.

Para el caso específico de nuestra estancia, podemos ver que nuestras tablas fueron diseñadas con un campo llamado “timestamp”, en el cual se guarda la fecha en que los datos fueron ingresados o modificados, lo que hace que las tareas de carga incremental sean relativamente fáciles. Dicha facilidad radica en que solo necesitaremos modificar los queries que empleamos para cargar los datos para que estos contengan una restricción de fecha, dicho de otra forma, pediremos que los queries inserten datos que se generaron a partir de la fecha en que se realizo la última carga.

Podemos ver este tipo de campo “*timestamp*” en tablas como ESTANCIAS, ANOMALIAS, MEDICAMENTOS, CONSUMO_COMIDAS, CONSUMO_DESAYUNOS, ETC.

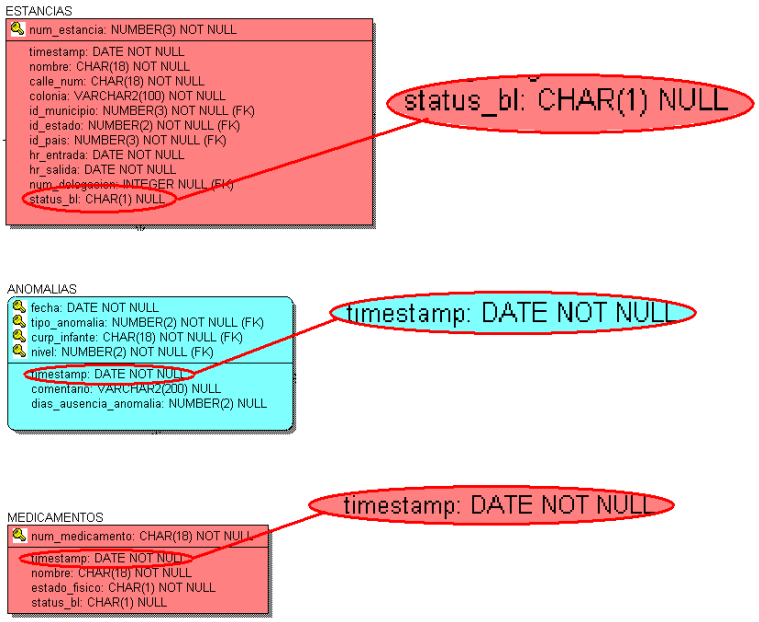


Fig 3.29 Tablas con campo timestamp

Conexión con Oracle

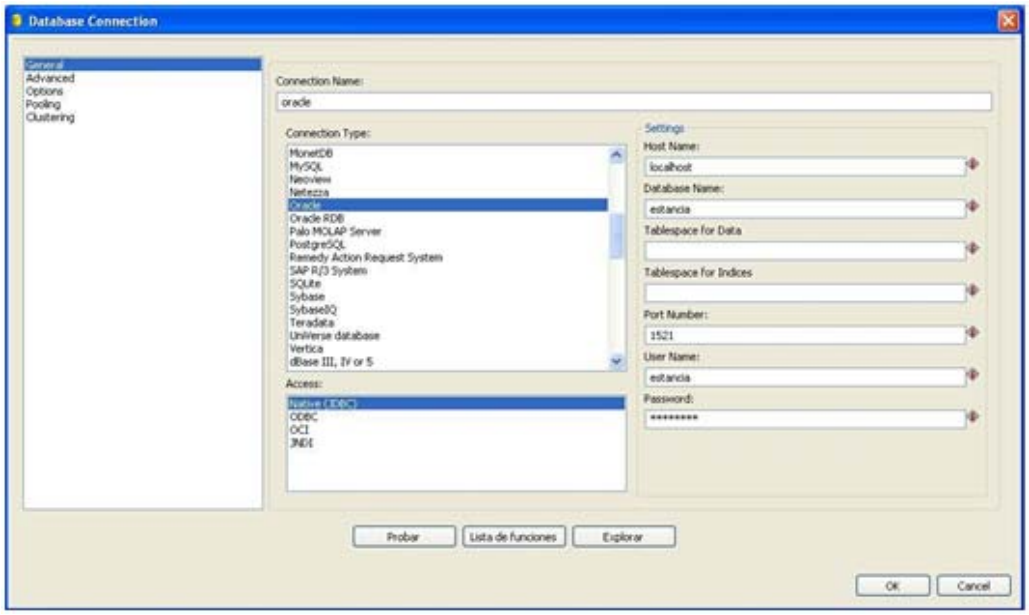


Fig 3.30. Configuración de la conexión en Oracle

Conexión con SqlServer

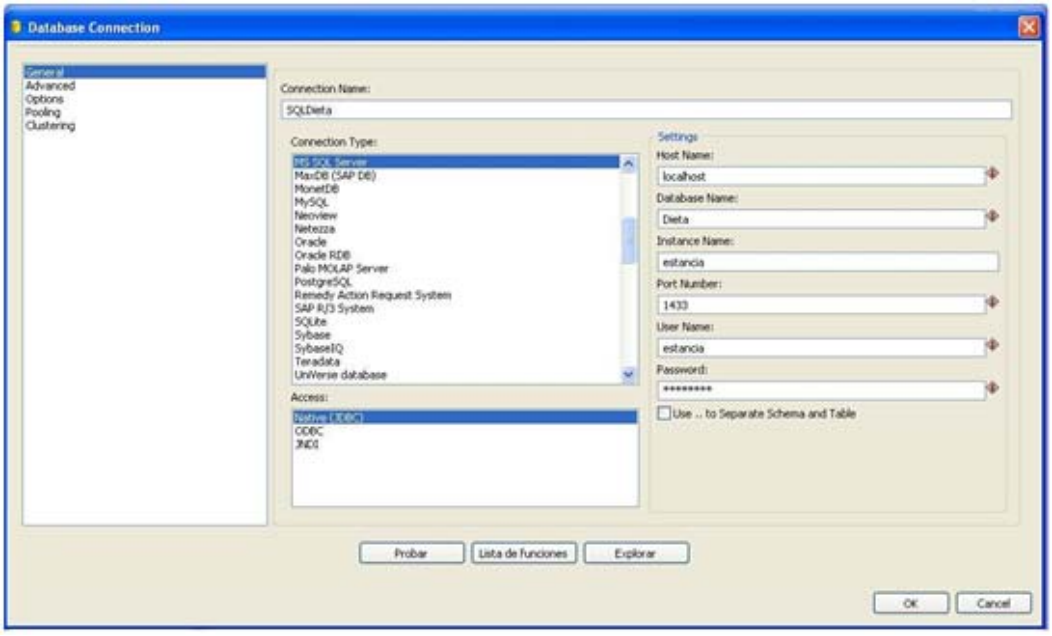


Fig 3.31. Configuración de la conexión en SQL Server

Ahora definimos una nueva transformación, en la que la entrada será una tabla del OLTP ESTANCIA (Oracle) y la salida la Base de Datos OLAP DIETA (SQLServer).

Transformación para la carga de datos:

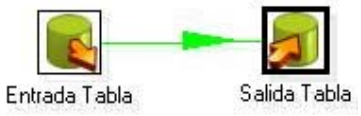


Fig 3.32. Visualización de una transformación en Pentaho

En la configuración de “Entrada tabla” ingresaremos el código SQL necesario para realizar la carga incremental deseada

Fig 3.33 Configuración de la tabla entrada en Pentaho

En la salida asignamos la conexión correspondiente a nuestra base de datos OLAP en SQLServer, y el nombre de la Tabla destino.

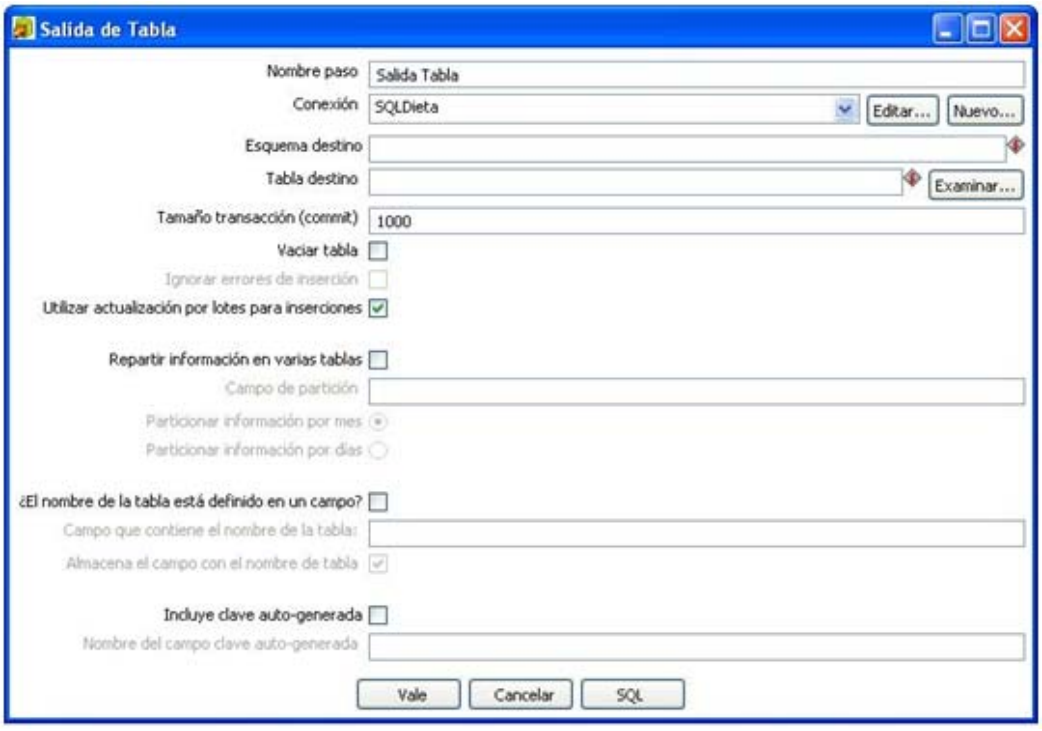


Fig 3.34. Configuración de la tabla salida en Pentaho

Los queries empleados para realizar la carga incremental son muy parecidos a los empleados para hacer la primera carga, solo que estos tienen una nueva condición que es que los datos ingresados deben de ser posteriores al año 2009.

Los queries empleados fueron:

Para la Dimensión INFANTE:

MENOR

```
SELECT Ifn.curp_infante,Ifn.nivel,I.sexo FROM infantes_nivel Ifn, infantes I,CONSUMO_COMIDAS CM
where Ifn.curp_infante=I.curp_infante AND I.curp_infante=CM.CURP_INFANTE AND
to_char(CM.timestamp, 'yyyy') >= 2009;
```

Para la Dimensión TIEMPO:

AÑO

```
        SELECT distinct to_char(FECHA, 'yyyy') as "año"from consumo_comidas WHERE
to_char(timestamp, 'yyyy') >= 2009
 WHERE TO_CHAR(TIMESTAMP,'YYYY') >= 2009;
```

SEMESTRE

```
SELECT distinct to_char(FECHA, 'yyyy') as "año",
case
when to_char(FECHA, 'MM') <= 6 then '1'
when to_char(FECHA, 'MM') > 6 then '2'
else ''
end "num_sem",
case
when to_char(FECHA, 'MM') <= 6 then 'Primer'
when to_char(FECHA, 'MM') > 6 then 'Segundo'
else ''
end "nombre"
from consumo_comidas
WHERE to_char(timestamp, 'yyyy') >= 2009
order by "año","num_sem";
```

MES

```
SELECT distinct to_char(FECHA, 'yyyy') as "año",
case
when to_char(FECHA, 'MM') <= 6 then '1'
when to_char(FECHA, 'MM') > 6 then '2'
else ''
end "num_sem",
to_char(FECHA, 'MM') as "num_mes",
case
when to_char(FECHA, 'MM') = 1 then 'enero'
when to_char(FECHA, 'MM') = 2 then 'febrero'
when to_char(FECHA, 'MM') = 3 then 'marzo'
when to_char(FECHA, 'MM') = 4 then 'abril'
when to_char(FECHA, 'MM') = 5 then 'mayo'
when to_char(FECHA, 'MM') = 6 then 'junio'
when to_char(FECHA, 'MM') = 7 then 'julio'
when to_char(FECHA, 'MM') = 8 then 'agosto'
when to_char(FECHA, 'MM') = 9 then 'septiembre'
when to_char(FECHA, 'MM') = 10 then 'octubre'
when to_char(FECHA, 'MM') = 11 then 'noviembre'
when to_char(FECHA, 'MM') = 12 then 'diciembre'
else ''
end "nombre"
from consumo_comidas
WHERE to_char(timestamp, 'yyyy') >= 2009
order by "año","num_sem","num_mes"
```

DIA

```
SELECT distinct to_char(FECHA, 'yyyy') as "año",
case
when to_char(FECHA, 'MM') <= 6 then '1'
when to_char(FECHA, 'MM') > 6 then '2'
else ''
end "num_sem",
to_char(FECHA, 'MM') as "num_mes",
fecha
from consumo_comidas
WHERE to_char(consumo_comidas.timestamp, 'yyyy') >= 2009
order by "año","num_sem","num_mes"
```

TABLA DE HECHOS:

```
SELECT a.clave, cd.curp_infante,cd.FECHA,cd.status_jugo as consumo, cd.nivel
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 1 AND A.id_original = cd.id_jugo AND A.des_com = 'd'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante,cd.FECHA,cd.status_fruta as consumo, cd.nivel
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 2 AND A.id_original = cd.id_fruta AND A.des_com = 'd'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante,cd.FECHA,cd.status_alimento as consumo, cd.nivel
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 3 AND A.id_original = cd.id_alimento AND A.des_com = 'd'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante,cd.FECHA,cd.status_cereal as consumo, cd.nivel
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 4 AND A.id_original = cd.id_cereal AND A.des_com = 'd'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante,cd.FECHA,cd.status_leche as consumo, cd.nivel
FROM consumo_desayunos cd, alimento a
WHERE a.TIPO_ALIMENTO = 7 AND A.id_original = cd.id_leche AND A.des_com = 'd'
AND  to_char(cd.timestamp, 'yyyy') >= 2009

UNION

SELECT a.clave, cd.curp_infante, cd.FECHA,cd.status_jugo as consumo, cd.nivel
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 1 AND A.id_original = cd.id_jugo AND A.des_com = 'c'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante, cd.FECHA,cd.status_fruta as consumo, cd.nivel
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 2 AND A.id_original = cd.id_fruta AND A.des_com = 'c'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION
```

```
SELECT a.clave, cd.curp_infante, cd.FECHA,cd.status_alimento as consumo, cd.nivel
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 3 AND A.id_original = cd.id_alimento AND A.des_com = 'c'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante, cd.FECHA,cd.status_cereal as consumo, cd.nivel
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 4 AND A.id_original = cd.id_cereal AND A.des_com = 'c'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante, cd.FECHA,cd.status_sopa as consumo, cd.nivel
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 5 AND A.id_original = cd.id_sopa AND A.des_com = 'c'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
UNION

SELECT a.clave, cd.curp_infante, cd.FECHA,cd.status_verdura_ensalada as consumo , cd.nivel
FROM consumo_comidas cd, alimento a
WHERE a.TIPO_ALIMENTO = 6 AND A.id_original = cd.id_verdura_ensalada AND A.des_com = 'c'
AND  to_char(cd.timestamp, 'yyyy') >= 2009
```

Antes de ejecutar la carga incremental haremos un select para ver los registros que actualmente contiene nuestra tabla de hechos “dieta”:

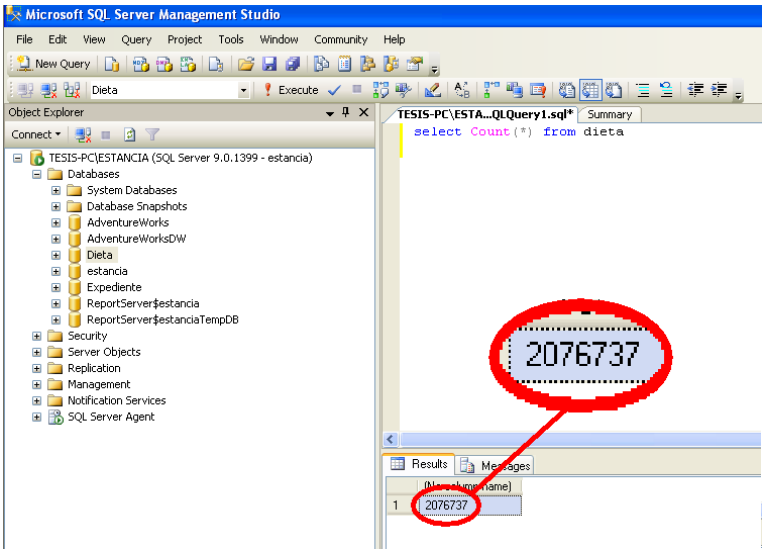


Fig 3.35 Comando select antes de realizar la carga incremental

Después de ejecutar la carga incremental podemos ver que el número de registros de la misma tabla “dieta” creció ya que se le añadieron 1270 registros nuevos correspondientes a los registros añadidos de las tablas consumo_comidas y consumo_desayunos que se generaron a partir del año 2009.

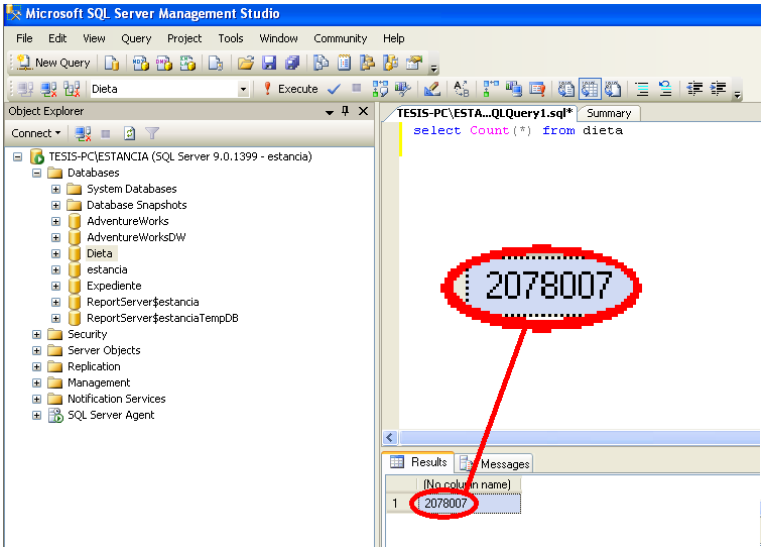


Fig 3.36 Comando select después de realizar la carga incremental

3.6 Implementación y manipulación de cubos

Creación del cubo de DIETA

Ahora procedemos a la creación del cubo en **SQL Server Business Inteliggence Development Studio.**

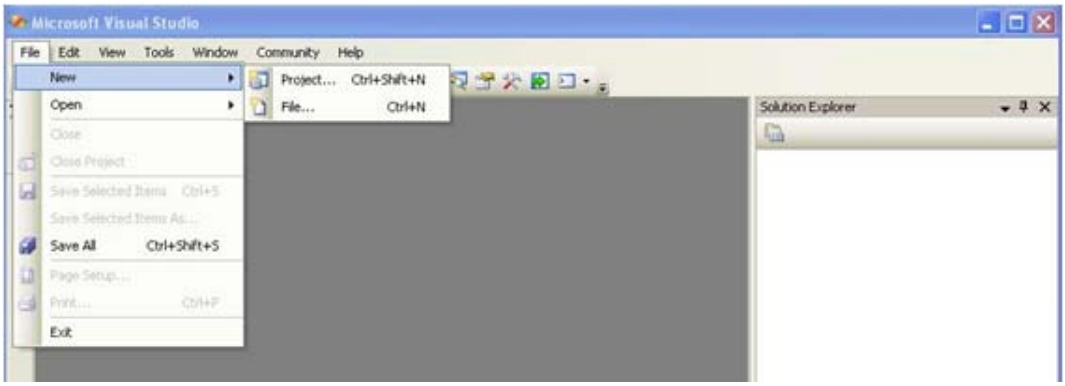


Fig 3.37 Pantalla principal de SQL Server Business Inteliggence Development Studio

Una vez abierto nuestro entorno de desarrollo de Microsoft Visual Studio 2005, hacemos clic en **Nuevo** y a continuación en **proyecto**:

En el cuadro de diálogo **Nuevo proyecto**, seleccionamos **Proyectos de Business Intelligence** en el panel **Tipos de proyecto**, y seleccionamos **Proyecto de Analysis Services** en el panel **Plantillas.**

Asignamos el nombre **Dieta** a nuestro proyecto y hacemos clic en **OK**.

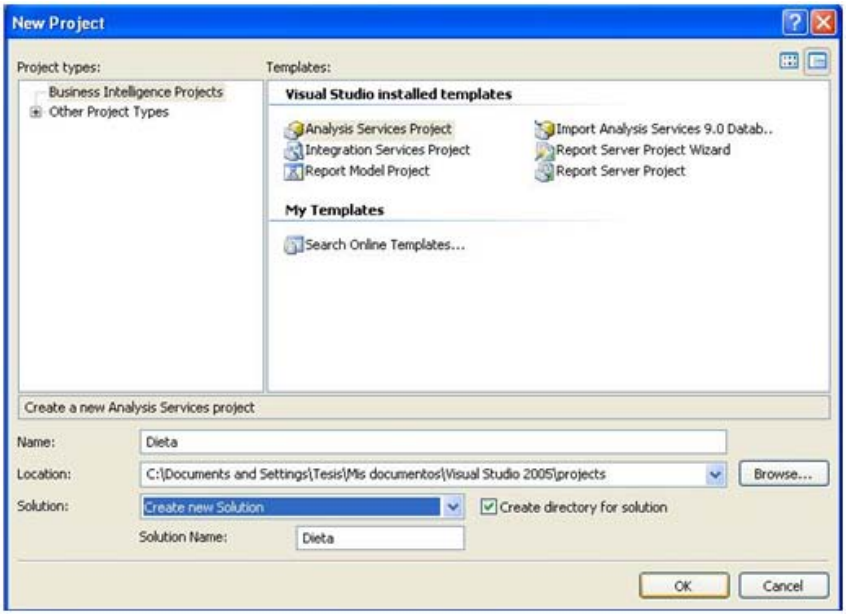


Fig 3.38 Asignando nombre al proyecto

Ahora vamos a ***definir un origen de datos nuevo.***

En el **Explorador de soluciones**, hacemos clic con el botón secundario en **Orígenes de datos** y, a continuación, hacemos clic en **Nuevo origen de datos**.

Se abre el Asistente para orígenes de datos. En la página de inicio del Asistente para orígenes de datos, hacemos clic en **Siguiente**.

Nos aseguramos de que la opción **Crear un origen de datos basado en una conexión nueva o existente** esté seleccionada y, a continuación, hacemos clic en **Nuevo**.

Fig 3.39 Seleccionando la conexión

En la lista **Proveedor**, comprobamos que la opción **Native OLE DB\SQL Native Client** está seleccionada. En el cuadro de texto **Nombre de servidor**, escribimos **localhost\estancia**.

Comprobamos que la opción **Utilizar autenticación de Windows** está seleccionada. En la lista **Seleccione o escriba un nombre de base de datos**, seleccionamos **Dieta** .

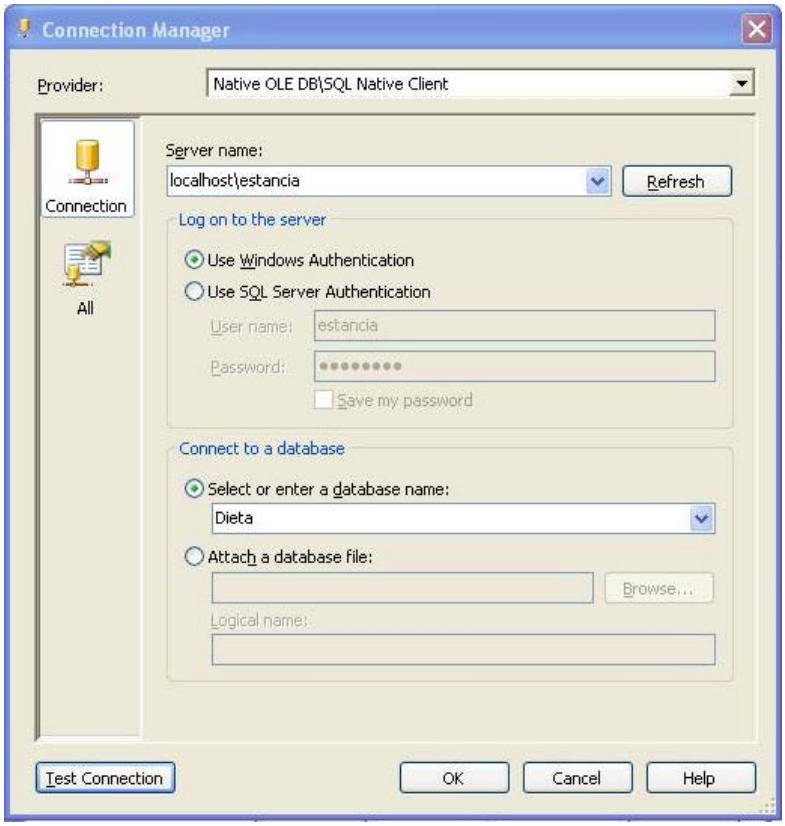


Fig 3.40 Seleccionando servidor

Comprobamos nuestra conexión, haciendo clic en **Prueba conexión:**

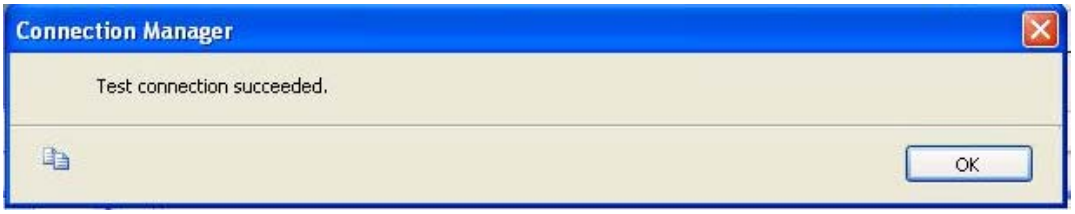


Fig 3.41 Probado la conexión

Ok, ahora seleccionamos **Utilizar cuenta de servicio** y hacemos clic en **Siguiente**.

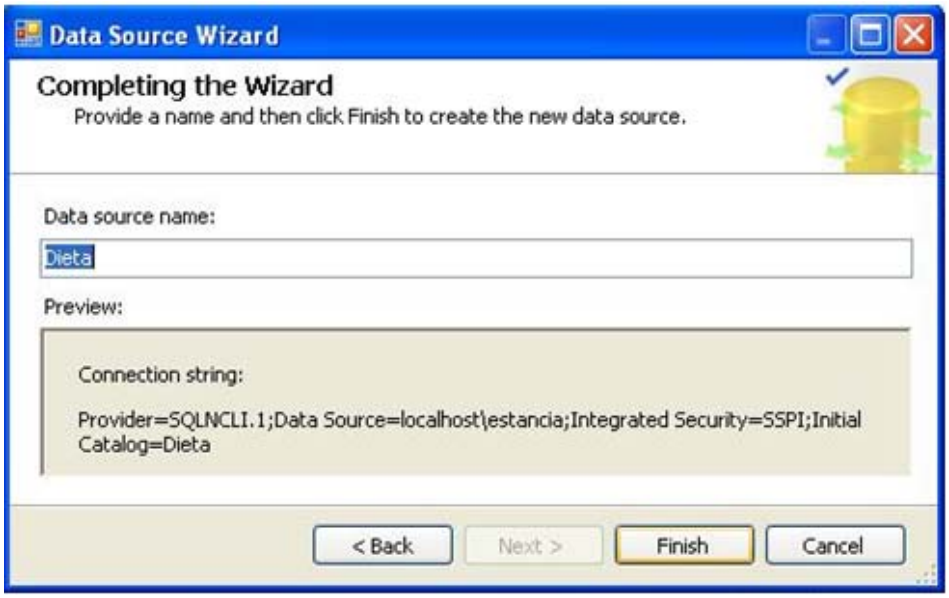


Fig 3.42 Seleccionando Cuenta de Servicio

La imagen siguiente muestra la página **Finalización del asistente**.

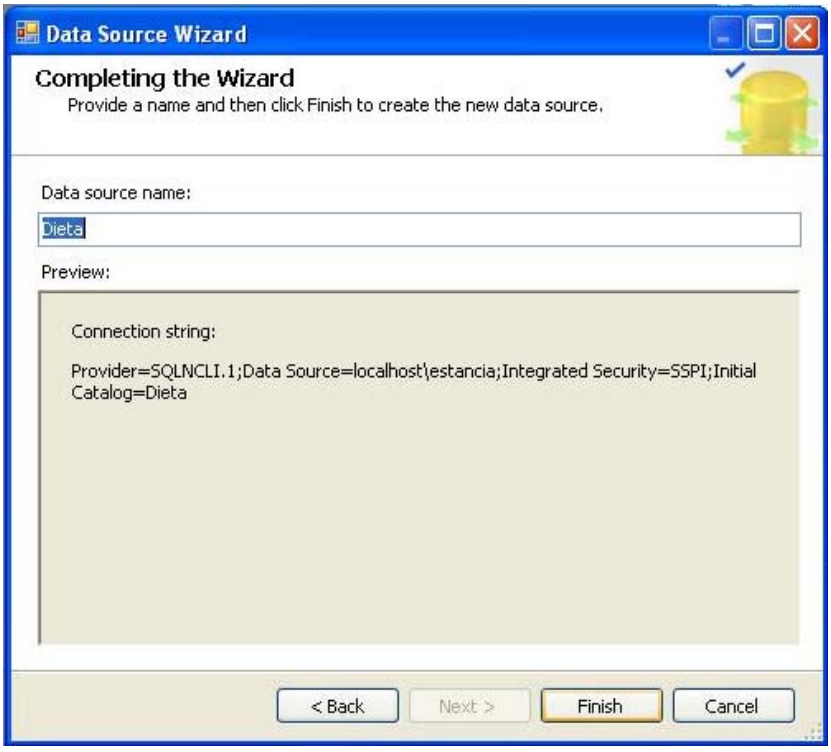


Fig 3.43 Finalización del asistente.

Hemos definido correctamente el origen de datos Dieta para el proyecto Dieta.

Ahora vamos a definir una ***vista de origen de datos nueva.***

En el Explorador de soluciones, hacemos clic con el botón secundario en **Vistas de origen de datos** y, a continuación, hacemos clic en **Nueva vista de origen de datos**.

Se abre el Asistente para vistas de origen de datos.

En la página **Asistente para vistas de origen de datos**, hacemos clic en **Siguiente**.

Aparece la página **Seleccionar un origen de datos**. En **Orígenes de datos relacionales**, el origen de datos **Dieta** aparece seleccionado. Hacemos clic en **Siguiente**.

Aparece la página **Seleccionar tablas y vistas**. En esta página seleccionamos todas las tablas pues son de las que estará compuesto nuestro datamart Dieta.

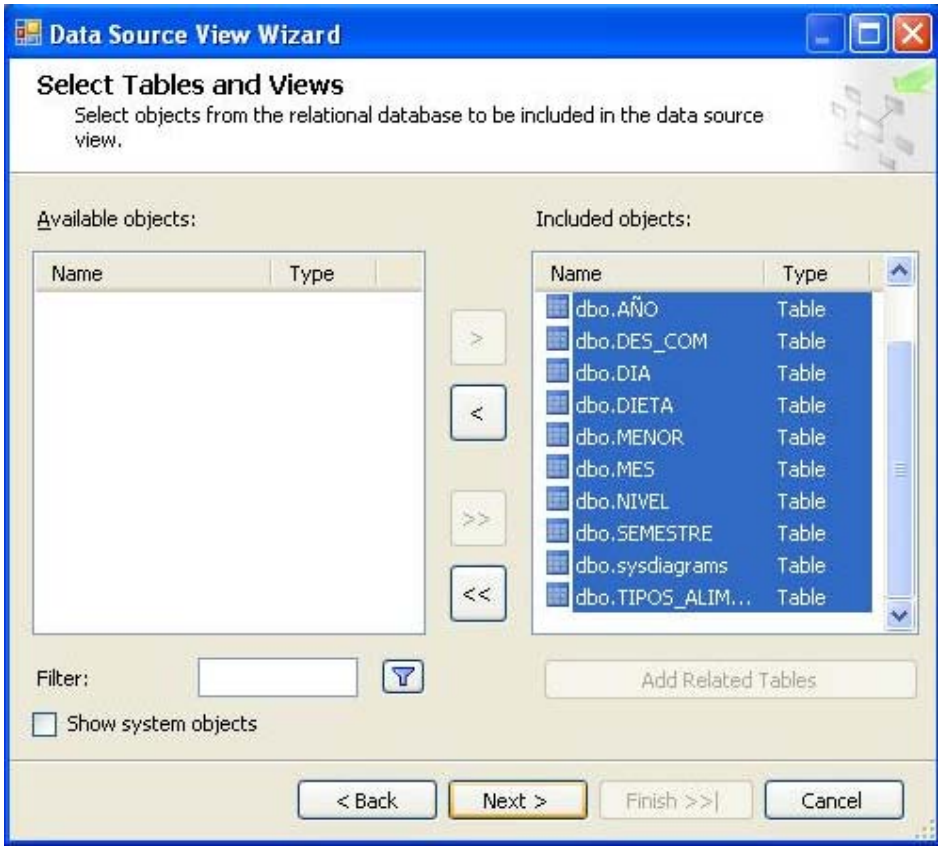


Fig 3.44 Selección de tablas y vistas

Hacemos clic en **Siguiente** y, a continuación, hacemos clic en **Finalizar** .

En la imagen siguiente se muestra el panel **Diagrama** del Diseñador de vistas de origen de datos.

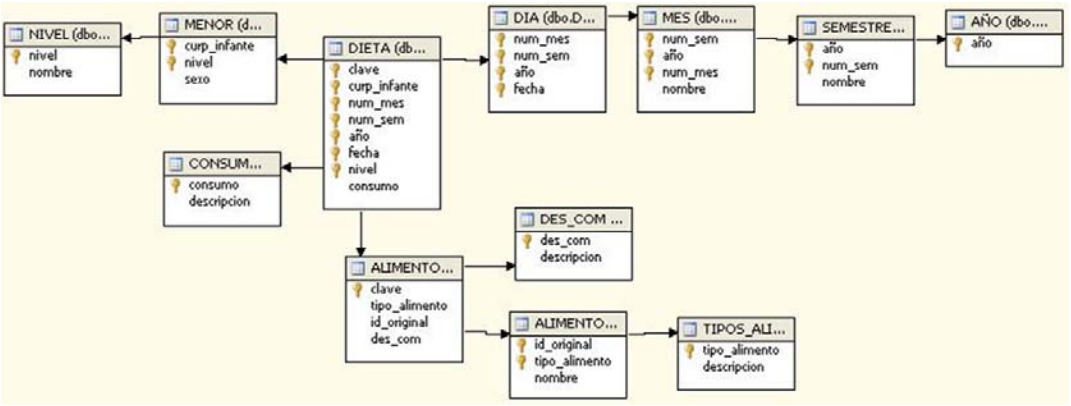


Fig 3.45 Diagrama del diseñador de vistas

Ahora vamos procedemos a ***definir nuestro cubo y sus propiedades***:

En el Explorador de soluciones, hacemos clic con el botón secundario en **Cubos** y, a continuación, hacemos clic en **Nuevo cubo**.

En la página **Asistente para cubos**, hacemos clic en **Siguiente**. En la página **Seleccionar método de generación**, comprobamos que las opciones **Generar el cubo con un origen de datos** y **Generación automática** están seleccionadas y hacemos clic en **Siguiente**.

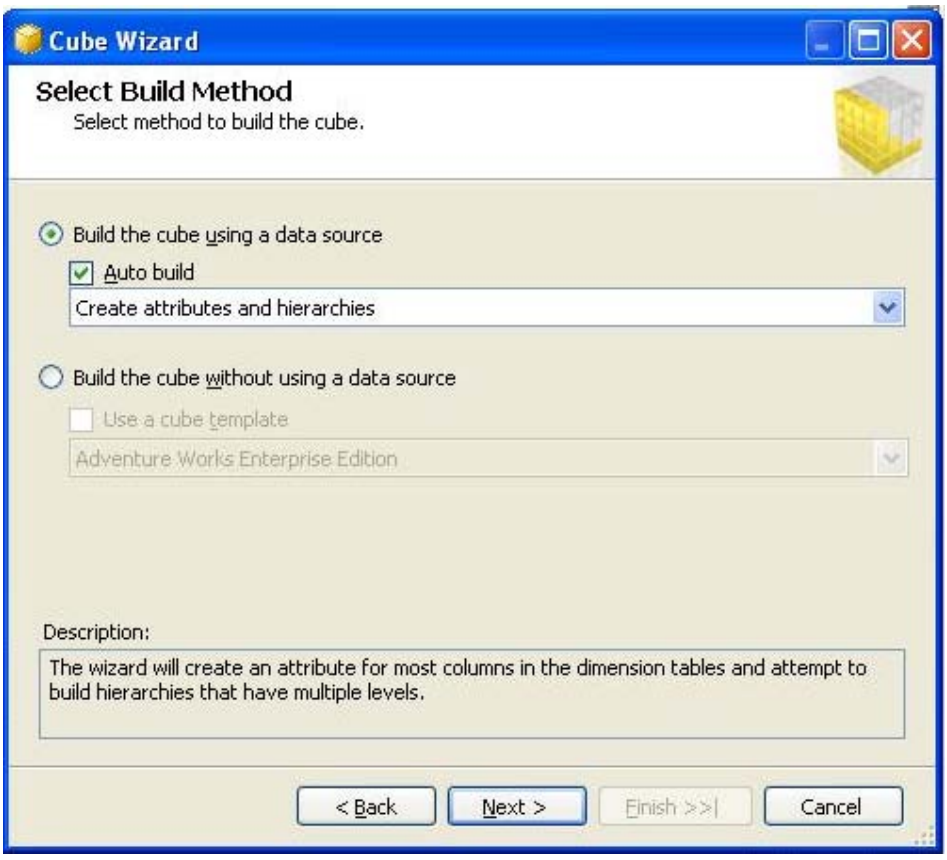


Fig 3.46 Pantalla principal del asistente para cubos

En la página **Seleccionar vista de origen de datos**, comprobamos que la vista de origen de datos **Dieta** está seleccionada. Hacemos clic en **Siguiente**.

En la página **Detectando tablas de hechos y de dimensiones**, hacemos clic en **Siguiente** cuando el asistente haya identificado las tablas de hechos y de dimensiones.

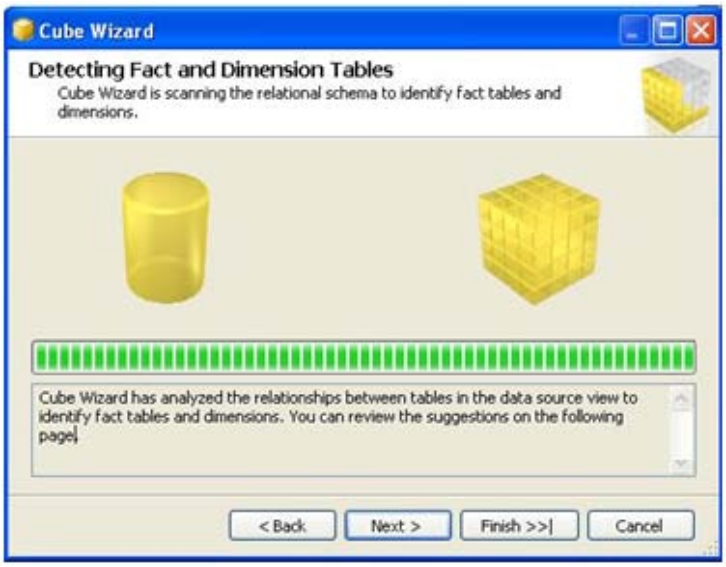


Fig 3.47 Detectando tablas de hechos y de dimensiones

En la página **Identificar tablas de hechos y de dimensiones** se muestran las tablas de hechos y de dimensiones identificadas por el asistente, verificamos que la tabla de hechos es **DIETA**.

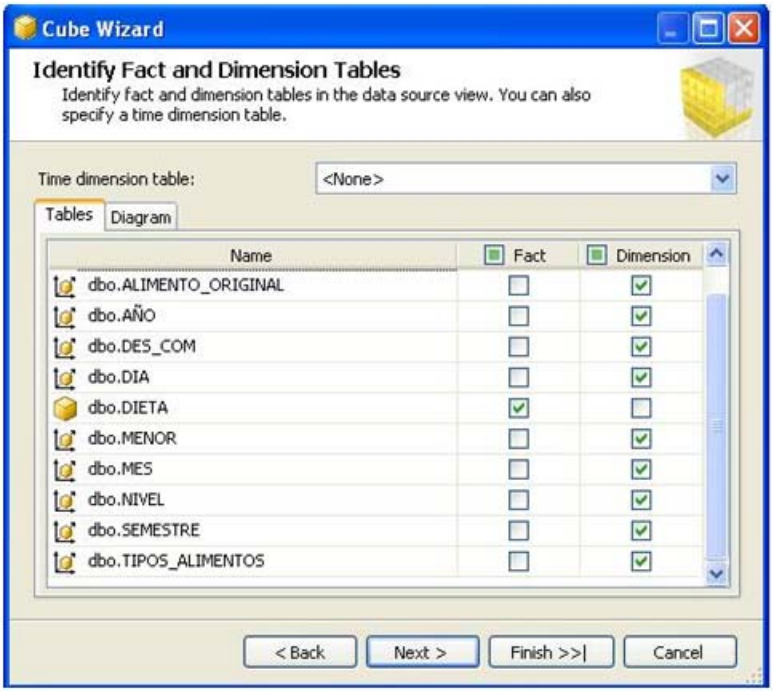


Fig 3.48 Identificar tablas de hechos y de dimensiones

En la página **Identificar tablas de hechos y de dimensiones**, seleccione **DIA** en la lista **Tabla de dimensiones de tiempo** y hacemos clic en **Siguiente**.

En la página **Seleccionar períodos de tiempo**

Fig 3.49 Seleccionar períodos de tiempo

- Asigne la propiedad **Year** a la columna a**ño**
- Asigne la propiedad **Half Year** a la columna **num_sem**.
- Asigne la propiedad **Month** a la columna **num_mes**.
- Asigne la propiedad **Date** a la columna **fecha**.
-

Hacemos clic en **siguiente.**

En la página **Seleccionar medidas**, revise las medidas seleccionadas en el grupo de medida **DIETA**.

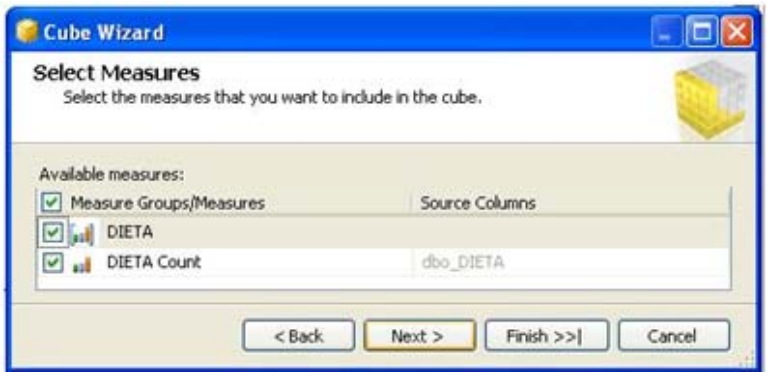


Fig 3.50 Seleccionar Medidas

Hacemos clic en siguiente, después en la página **Detectando jerarquías**, hacemos clic en **Siguiente**.

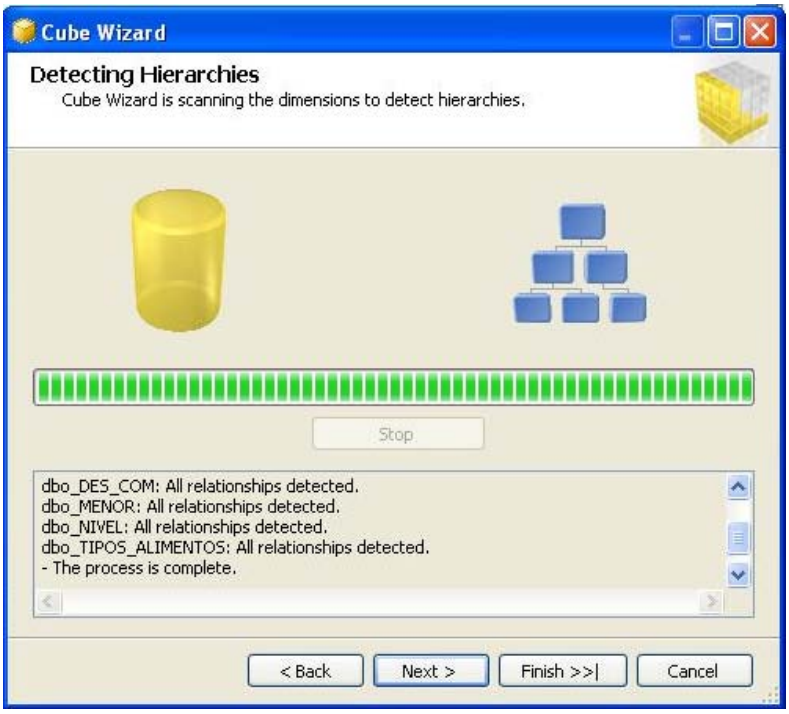


Fig 3.51 Detectando Jerarquías

En la página **Revisar las nuevas dimensiones**, revisamos la estructura de la jerarquía de dimensiones de las tres dimensiones expandiendo el control de árbol para ver las jerarquías y los atributos que el asistente ha detectado para cada dimensión.

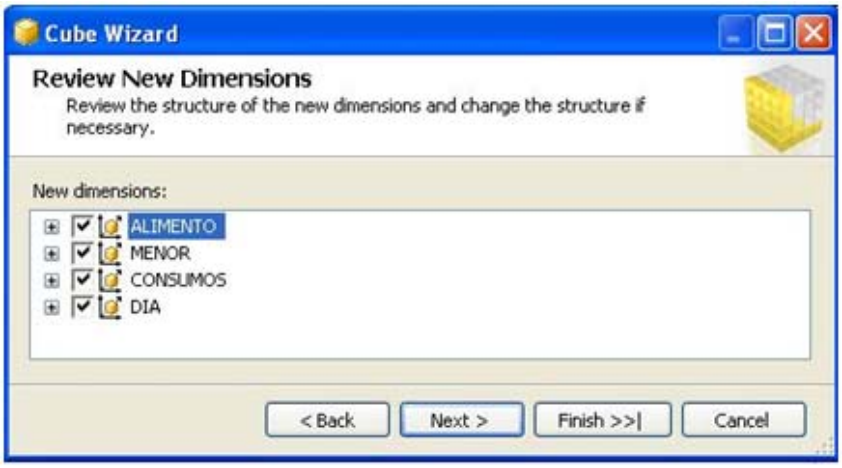


Fig 3.52 Revisar las nuevas dimensiones

Como podemos ver las dimensiones mostradas son correctas, entonces hacemos clic en **finalizar**

En la imagen siguiente, se muestran las tablas de dimensiones y de hechos en el diseñador. Observe que la tabla de hechos es amarilla y las tablas de dimensiones son azules.

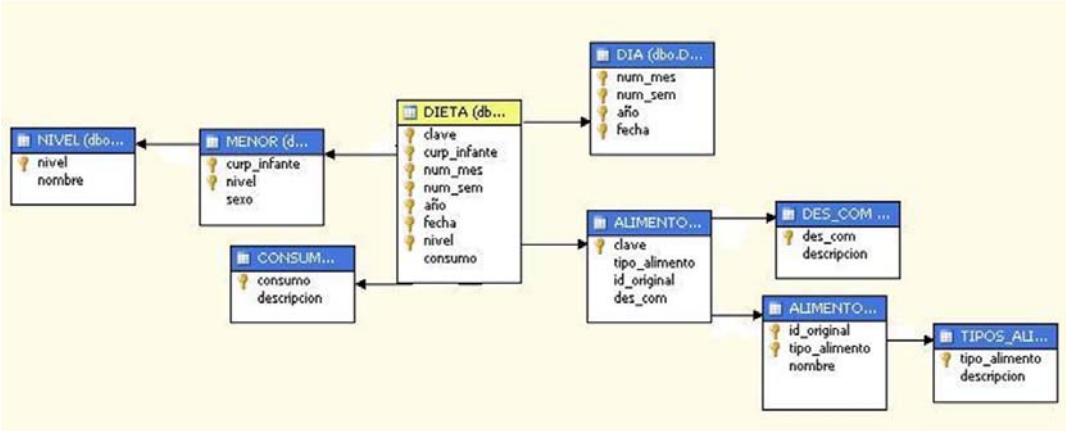


Fig 3.53 Tablas de dimensiones y de hechos en el diseñador

Una vez hecho esto **guardamos Todo**, tenemos nuestro cubo construido.

Después de revisar las propiedades de cubo y dimensiones.

Procedemos a implementar el proyecto para esto:

En el Explorador de soluciones, hacemos clic con el botón secundario en el proyecto **DIETA** y, a continuación, hacemos clic en **Propiedades**.

En el nodo **Propiedades de configuración** del panel de la izquierda, hacemos clic en **Implementación**. En la ficha destino cambiamos el nombre del servidor a **localhost\estancia**, como se muestra a continuación:

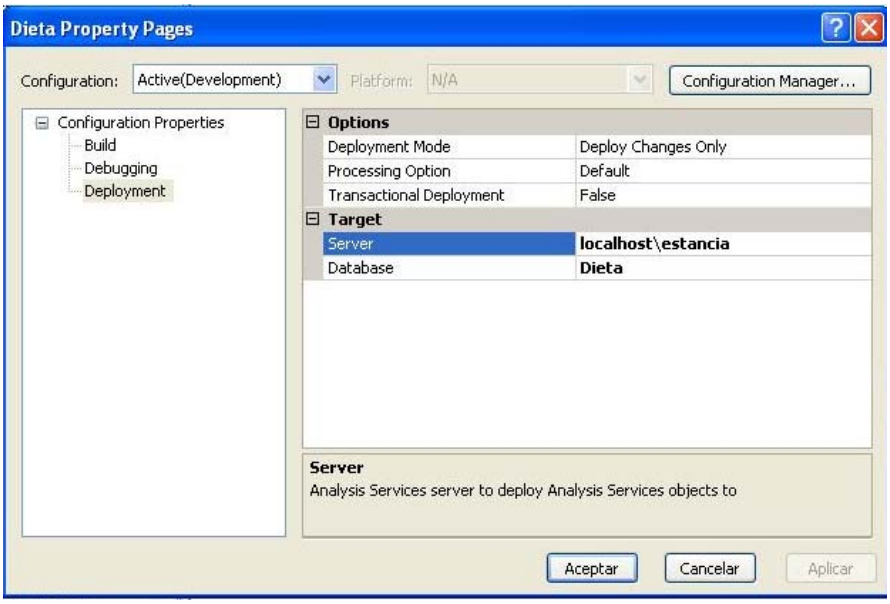


Fig 3.54 Propiedades de configuración

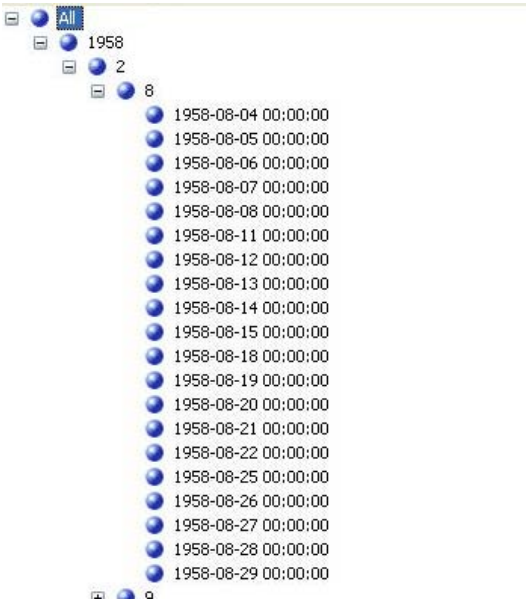
Ahora en el Explorador de soluciones, hacemos clic con el botón secundario en el proyecto **DIETA** y, a continuación, hacemos clic en **Implementar.**

Y esperamos que se implemente nuestro proyecto finalmente podremos observar que se completo exitosamente.

Fig 3.55 Mensaje de proceso exitoso

Ahora haremos algunas modificaciones para examinar nuestro cubo implementado.

Cambie al Diseñador de dimensiones para la dimensión DIA de Business Intelligence Development Studio y por default está en la ficha **Estructura de dimensión** cambiamos a **examinador** y hacemos clic en el ícono **reconectar**.



Como podemos ver al extender nuestro árbol de jerarquías, las fechas que nos muestra no son muy descriptivas.

Fig 3.56 Árbol de jerarquías

Modificamos la medida tiempo para que ésta sea más descriptiva para eso:

Cambiamos al Diseñador de dimensiones para la dimensión DIA de Business Intelligence Development Studio y hacemos clic en la ficha **Estructura de dimensión**.

Cambiamos al Diseñador de vistas de origen de datos de la vista de origen de datos **DIETA**, hacemos clic con el botón secundario en Dia **(dbo.dia)** en el panel **Tablas** y, a continuación, hacemos clic en **Nuevo cálculo con nombre**.

En el cuadro de diálogo **Crear cálculo con nombre**, escribimos **SimpleDate** en el cuadro **Nombre de columna** y, a continuación, escribimos la secuencia de comandos SQL siguiente en el cuadro **Expresión**:

1. DATENAME(mm, fecha) + ' ' +
2. DATENAME(dd, fecha) + ' ' +
3. DATENAME(yy, fecha)

Hacemos clic en **Aceptar** y luego cambiamos al Diseñador de dimensiones para la dimensión Dia.

En la jerarquía definida por el usuario **num_mes-num_sem-año-fecha** , cambiamos el valor de la propiedad **SourceAttribute** , expandimos la colección de propiedades **NameColumn** y, a continuación, expandimos la colección de propiedades **Source** de la ventana Propiedades. Cambiamos el valor de la propiedad **ColumnID** por **SimpleDate**.

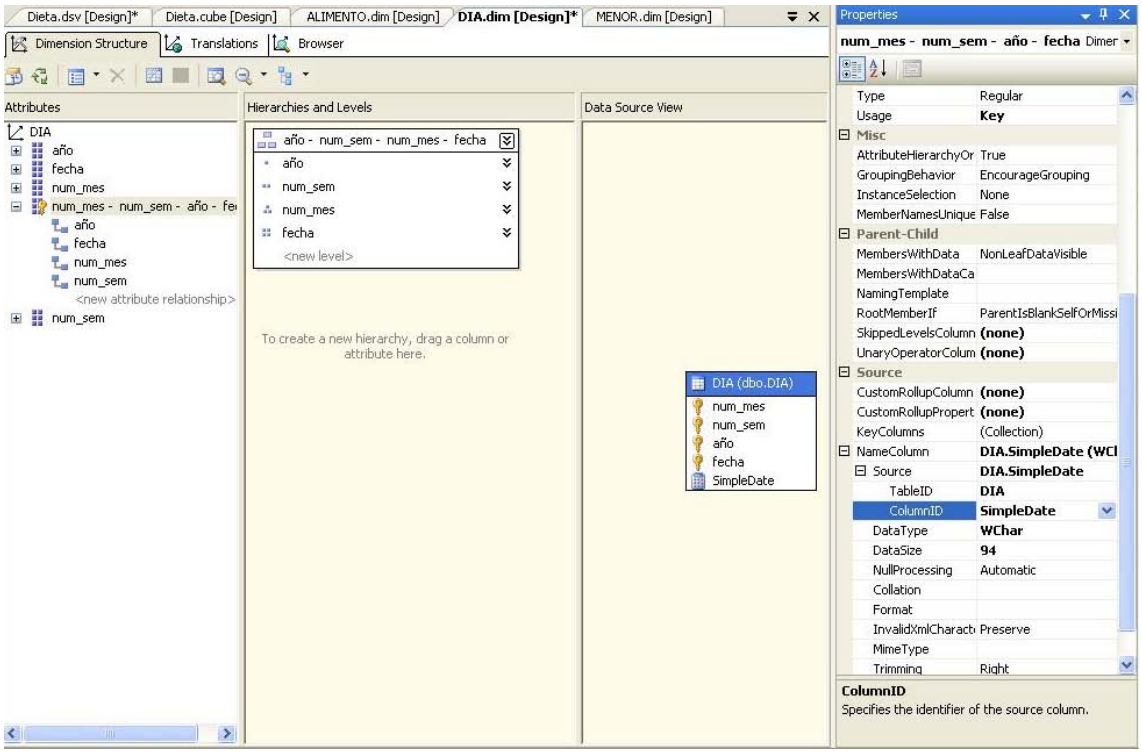


Fig 3.57 Ventana propiedades

En el menú **Generar** de BI Development Studio, hacemos clic en **Implementar DIETA**. Cuando la implementación finalice correctamente, hacemos clic en la ficha **Examinador** del Diseñador de dimensiones para la dimensión **Dia** y luego hacemos clic en **Volver a conectar** en la barra de herramientas. Podemos observar que hicimos más descriptivo el atributo **fecha**.

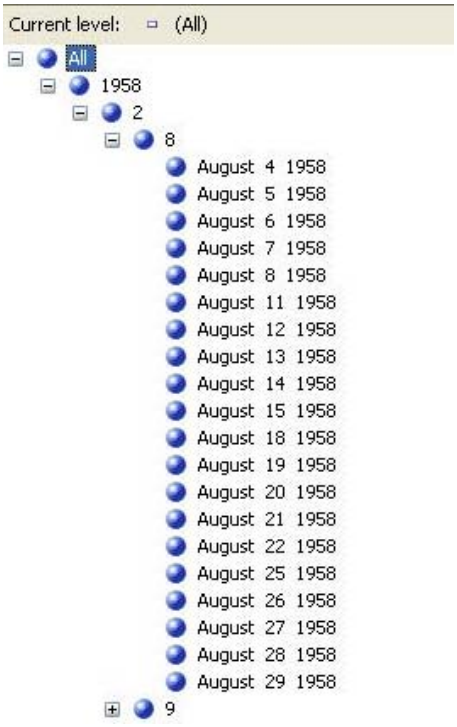


Fig 3.58 Árbol de jerarquías mejorado

Ahora Procedemos de la misma manera para describir **num_mes** para los meses y **num_sem** para los semestres.

Secuencia de comandos SQL **SimpleMonth**:

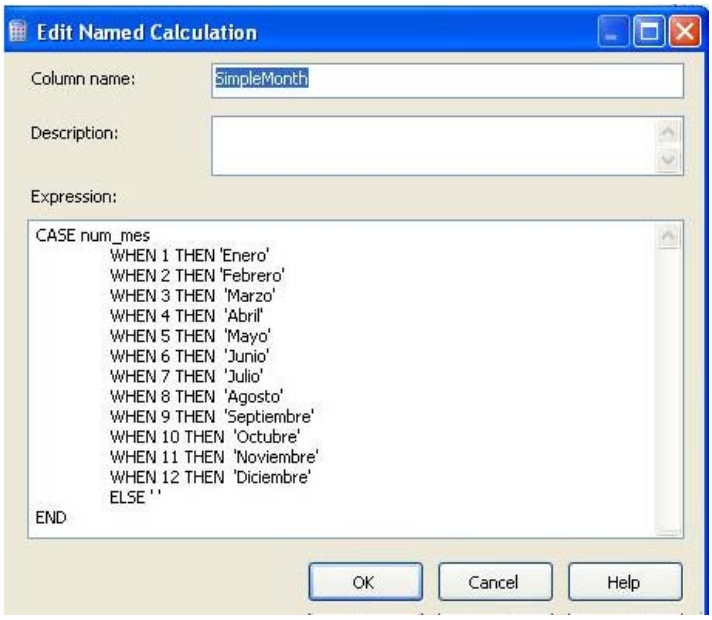


Fig 3.59 Secuencia de comandos SQL SimpleMonth

Secuencia de comandos SQL para **SimpleSemester**:

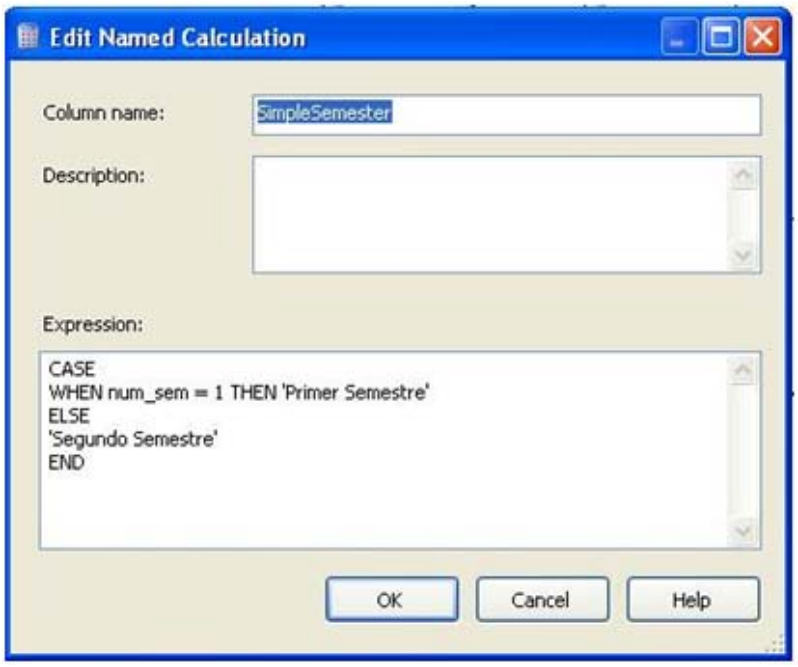


Fig 3.60 Secuencia de comandos SQL SimpleSemester

Ahora una vez implementado nuestro proyecto con los cambios que hicimos, así es como queda nuestra dimensión de tiempo más descriptiva:

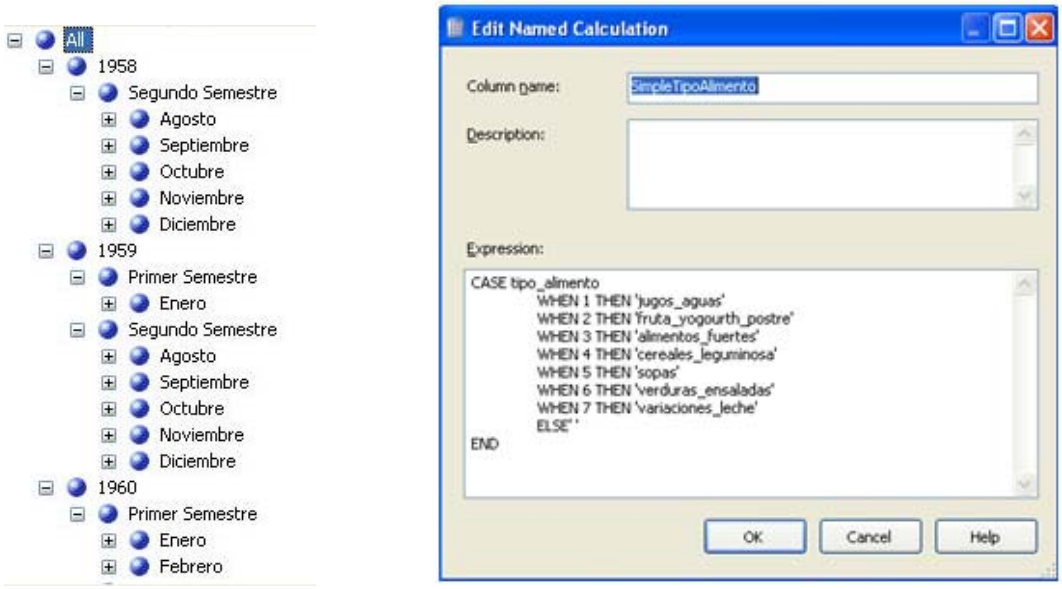


Fig 3.61 Vista de dimensión tiempo mejorada

También podemos hacer descriptivo un atributo que no es parte de la llave primaria de una dimensión, como tipo_alimento pues al examinar los datos resultaría muy útil que nos muestre el tipo alimento y no su id, a si que de la misma manera, creamos el cálculo.

Para ver los cambios en el diseñador de cubos:

Cambiamos al Diseñador de cubos de BI Development Studio haciendo clic en la ficha del diseñador del cubo Dieta. Seleccionamos la ficha **Examinador** y hacemos clic en **Volver a conectar** en la barra de herramientas del diseñador. Alternativamente, hacemos clic en el vínculo **Haga clic aquí para volver a intentar cargar el examinador** que aparece en el centro del panel del examinador.

En el panel izquierdo del diseñador se muestran los metadatos del cubo Dieta. Podemos ver que las opciones **Perspectiva** e **Idioma** están disponibles en la barra de herramientas de la ficha **Examinador**. También vemos que la ficha **Examinador** incluye dos paneles a la derecha del panel **Metadatos**: el superior es el panel **Filtro** y el inferior es el panel **Datos**.

En la imagen siguiente aparecen resaltados los paneles individuales en el Diseñador de cubos.

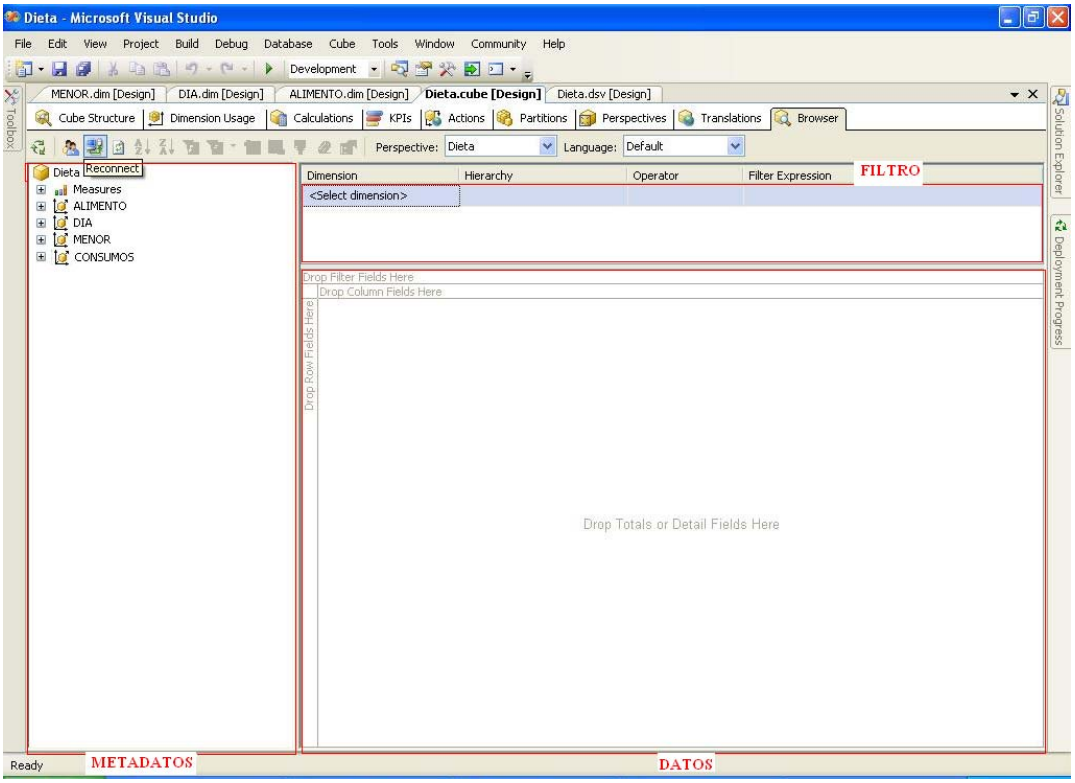


Fig 3.62 Paneles individuales del diseñador de cubos

Ahora procedemos a consultar la información más relevante de acuerdo a las necesidades presentadas inicialmente.

Primero expandimos **Measures,** expandimos **Dieta** y arrastramos la medida **DIETA Count** a el panel de datos en el área **Coloque Campos de totales o campos detallados aquí.**

Arrastramos la dimensión **MENOR** en el área **Coloque campos de fila aquí**, Arrastramos la dimensión **CONSUMO** en el área **Coloque campos de columnas aquí.**

Expandimos la dimensión **ALIMENTO**, arrastramos **DESCOM** en él área **Coloque campos de filtro aquí** y seleccionamos **c** (comida)**,** arrastramos también **Descripcion** en él área **Coloque campos de filtro aquí.** Y seleccionamos **verduras_ensaladas.**

Después arrastramos la dimensión **DIA** en el panel filtro y en operador seleccionamos **equal** y en expresión de filtro seleccionamos **2008.**

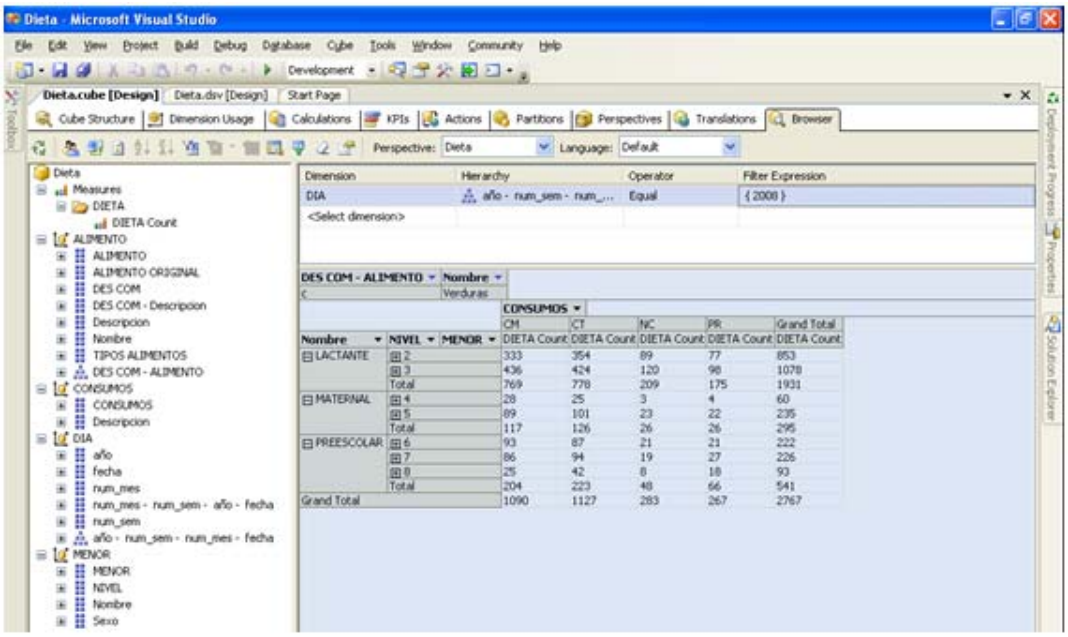


Fig 3.63 Consulta de información de los cubos

Esta es la vista que nos muestra dichos datos en particular.

Como podemos observar podemos comparar de acuerdo al consumo que nivel consume menos verduras, por ejemplo, a continuación veremos algunas de la necesidades que tiene las estancias infantiles, para poder solucionarlas con nuestro DWH.

Creación del cubo de Expediente

Ahora procedemos a la creación del cubo en **SQL Server Business Inteliggence Development Studio.** Una vez abierto nuestro entorno de desarrollo de Microsoft Visual Studio 2005, hacemos clic en **Nuevo** y a continuación en **proyecto**:

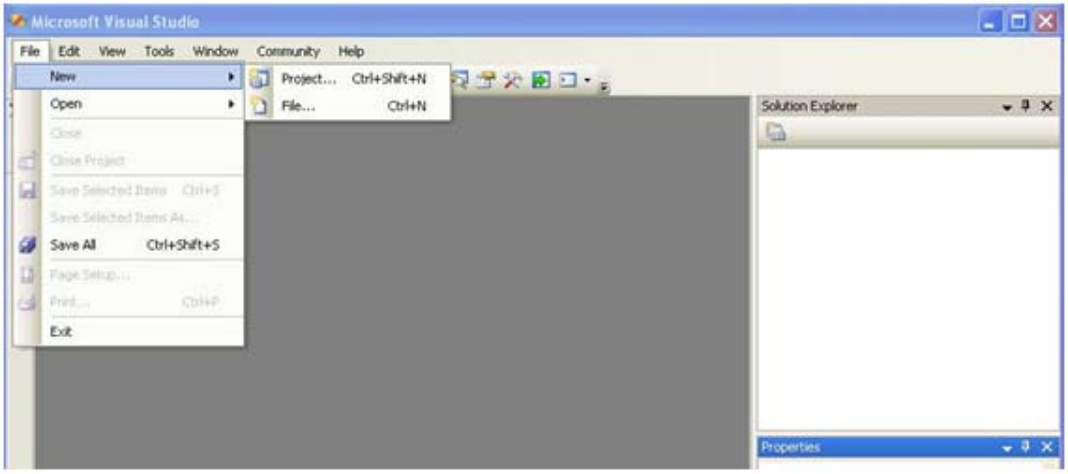


Fig 3.64 Pantalla indicial de SQL Server Business Inteliggence Development Studio

En el cuadro de diálogo **Nuevo proyecto**, seleccionamos **Proyectos de Business Intelligence** en el panel **Tipos de proyecto**, y seleccionamos **Proyecto de Analysis Services** en el panel **Plantillas.** Asignamos el nombre **clasedepositoDatos** a nuestro proyecto y hacemos clic en **OK**.

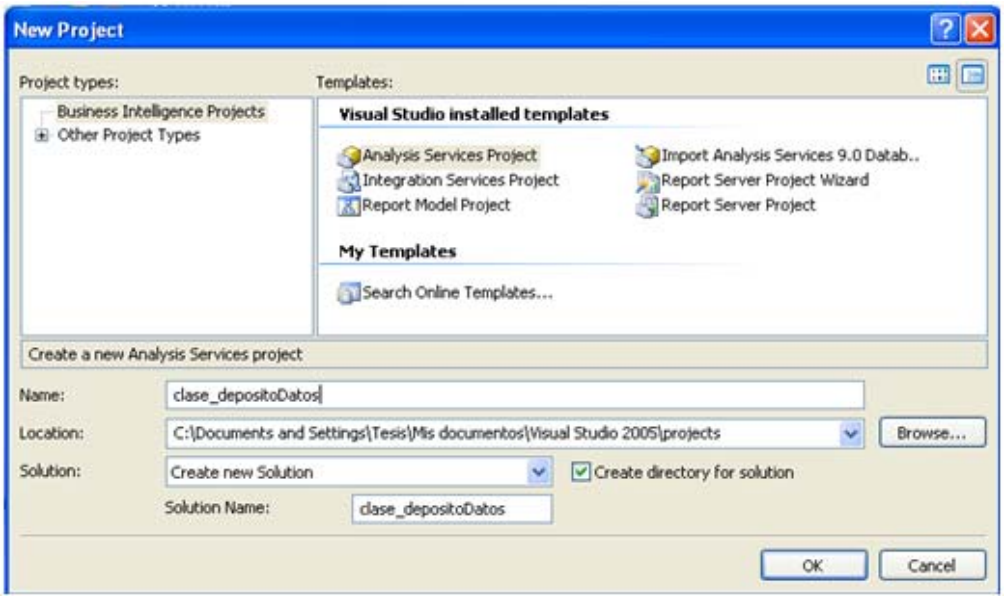


Fig 3.65 Asignando nombre al proyecto

Ahora vamos a ***definir un origen de datos nuevo.***

En el **Explorador de soluciones**, hacemos clic con el botón secundario en **Orígenes de datos** y, a continuación, hacemos clic en **Nuevo origen de datos**.

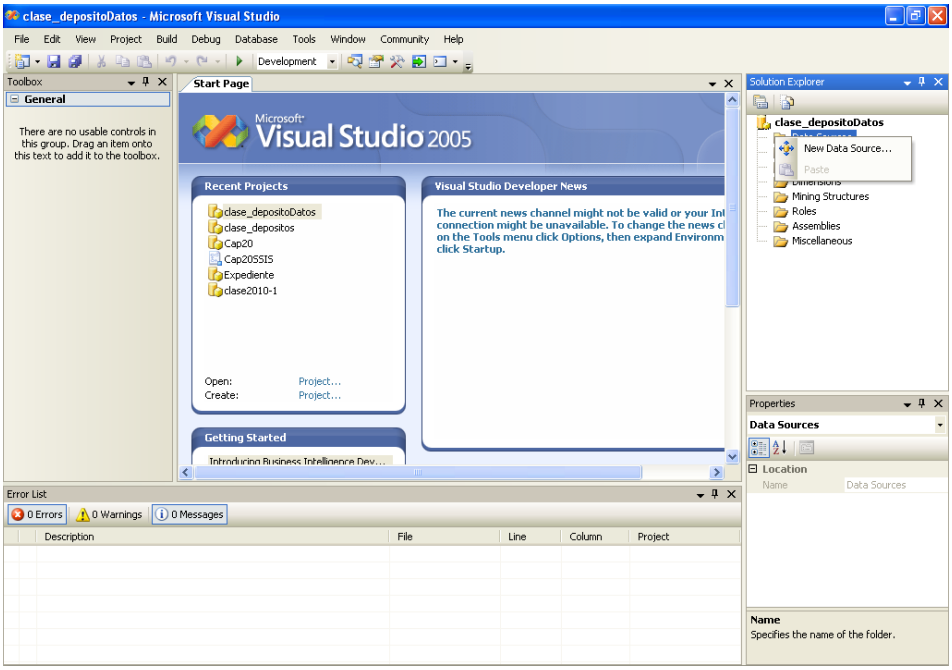


Fig 3.66 Panel explorador de soluciones

Se abre el Asistente para orígenes de datos. En la página de inicio del Asistente para orígenes de datos, hacemos clic en **Siguiente**.

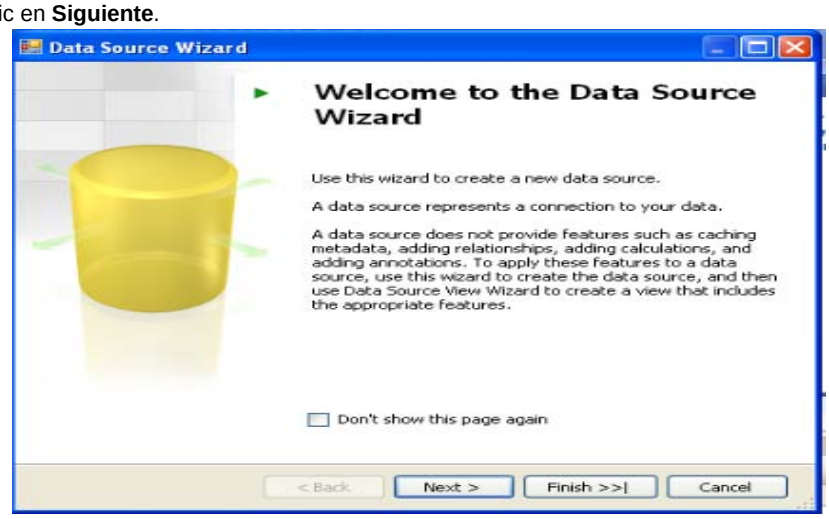


Fig 3.67. Pantalla de bienvenida del asistente para orígenes de datos

Nos aseguramos de que la opción **Crear un origen de datos basado en una conexión nueva o existente** esté seleccionada y, a continuación, hacemos clic en **Nuevo**.

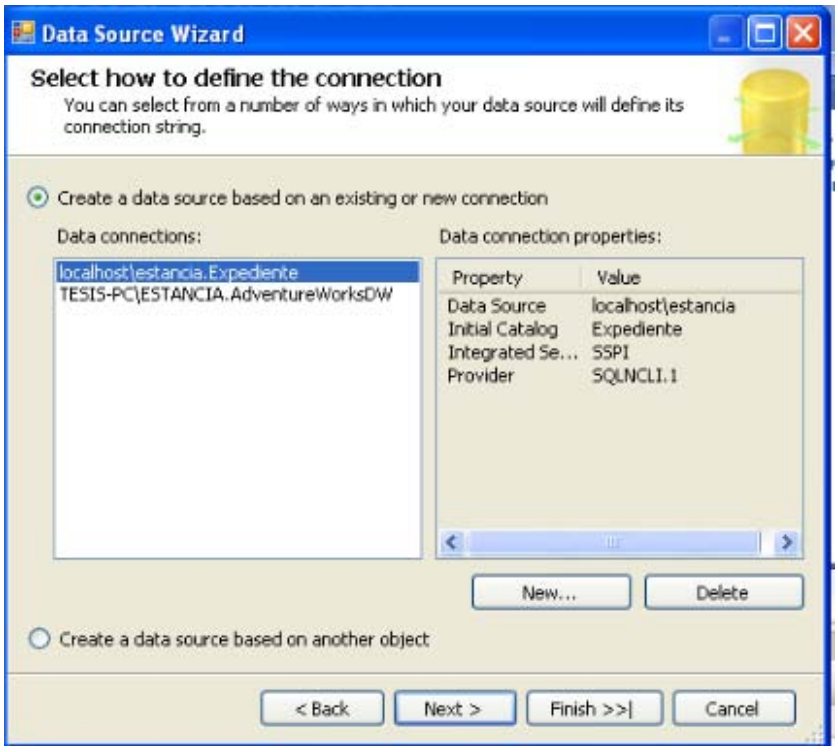


Fig 3.68 Seleccionando la conexión

Comprobamos que la opción **Default** está seleccionada.

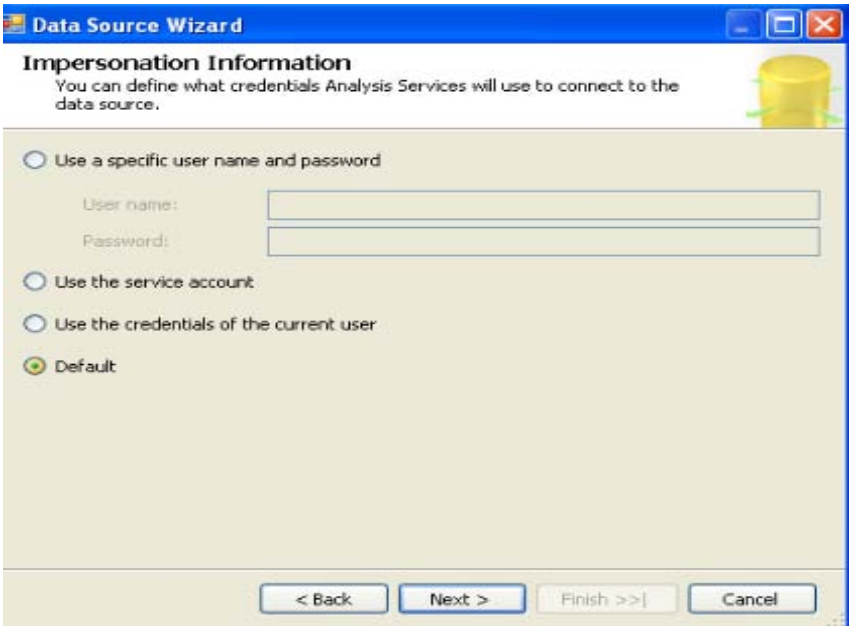


Fig 3.69 Seleccionando valores por default

Ok, ahora seleccionamos **Utilizar cuenta de servicio** y hacemos clic en **Siguiente**.

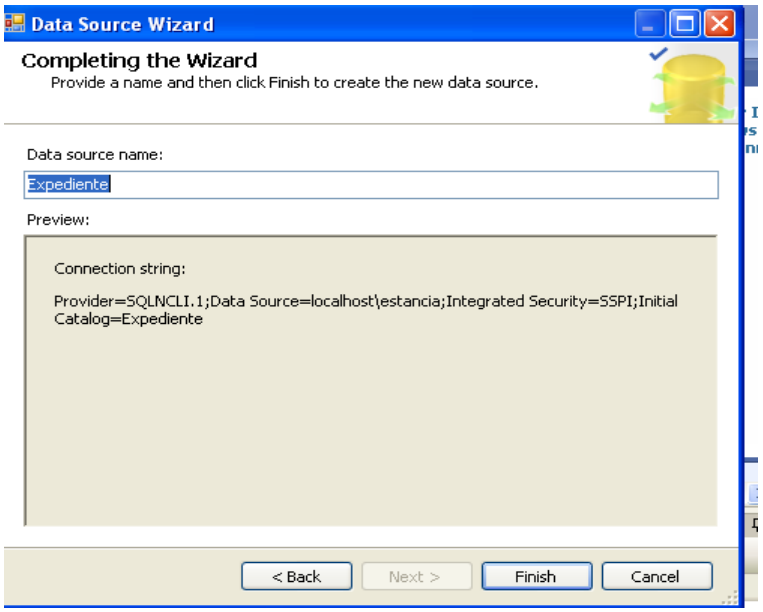


Fig 3.70 Seleccionando cuenta de servicio

Hemos definido correctamente el origen de datos Expediente para el proyecto clasedepositosDatos.

Ahora vamos a definir una ***vista de origen de datos nueva.***

En el Explorador de soluciones, hacemos clic con el botón secundario en **Vistas de origen de datos** y, a continuación, hacemos clic en **Nueva vista de origen de datos**.

Fig 3.71 Seleccionando la opción "vistas de origen de datos" en el "Panel explorador de soluciones"

Se abre el Asistente para vistas de origen de datos.

En la página **Asistente para vistas de origen de datos**, hacemos clic en **Siguiente**.

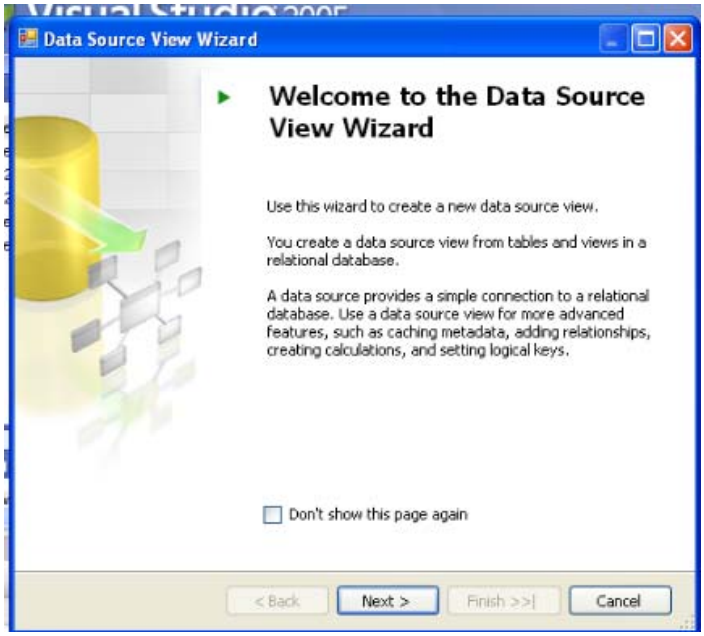


Fig 3.72 Pantalla de bienvenida del asistente de vistas de origen de datos

Aparece la página **Seleccionar un origen de datos**. En **Orígenes de datos relacionales**, el origen de datos **Expediente** aparece seleccionado. Hacemos clic en **Siguiente**.

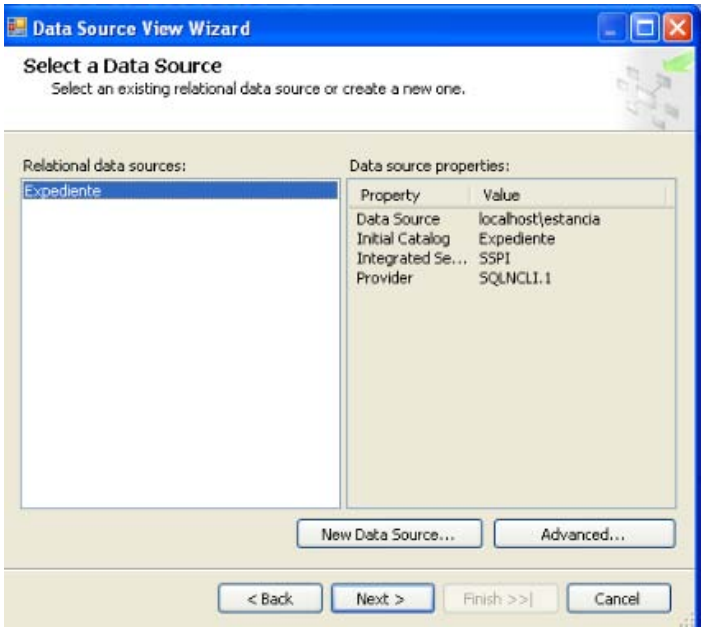


Fig 3.73 Seleccionando origen de datos

Aparece la página **Seleccionar tablas y vistas**. En esta página seleccionamos todas las tablas pues son de las que estará compuesto nuestro datamart Expediente.

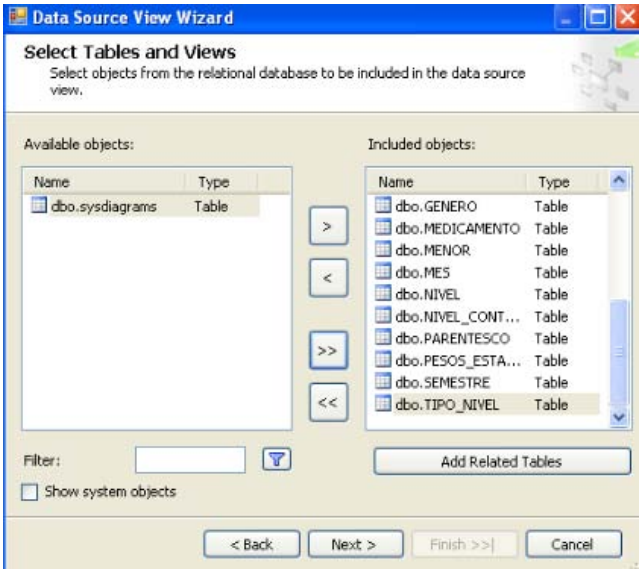


Fig 3.74 Seleccionando tablas y vistas

Hacemos clic en **Siguiente** y, a continuación, hacemos clic en **Finalizar**.

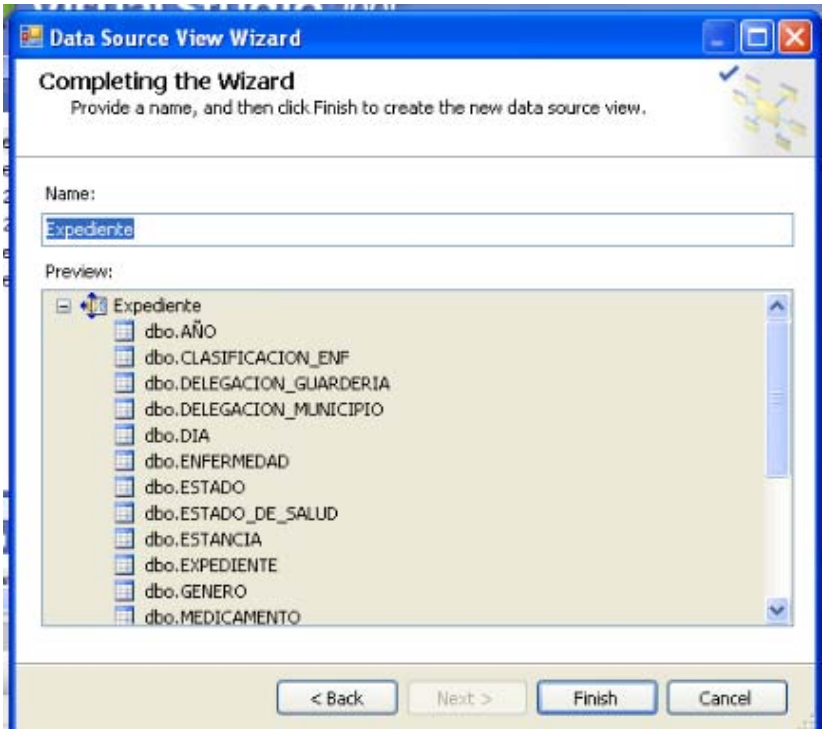


Fig 3.75 Finalización del wizard

Ahora vamos procedemos a ***definir nuestro cubo y sus propiedades***:

En el Explorador de soluciones, hacemos clic con el botón secundario en **Cubos** y, a continuación, hacemos clic en **Nuevo cubo**.

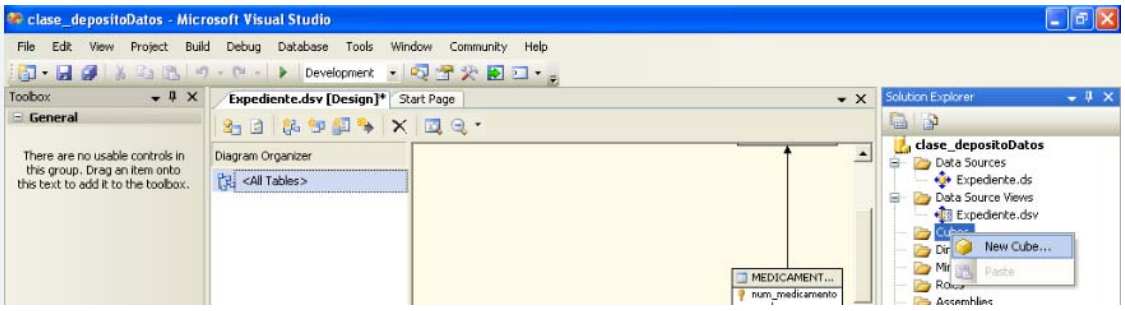


Fig 3.76 Explorador de solucione

En la página **Asistente para cubos**, hacemos clic en **Siguiente**.

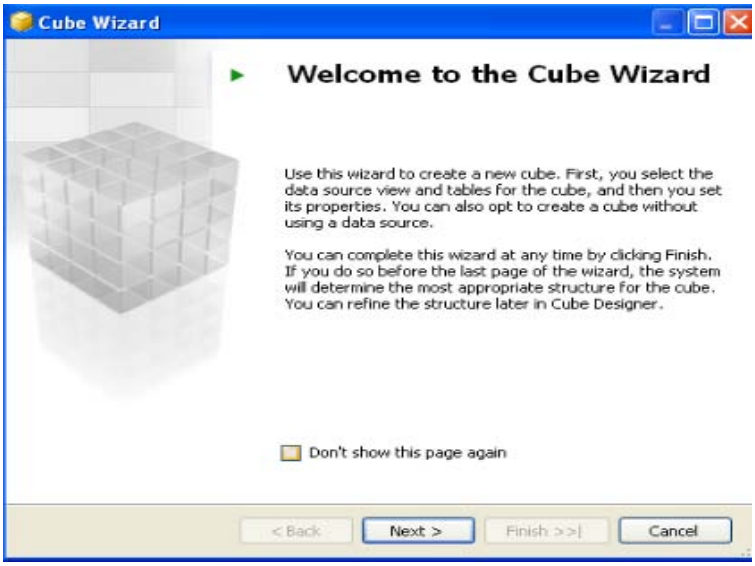


Fig 3.77 Pagina de bienvenida del asistente para la creación de cubos

En la página **Seleccionar método de generación**, comprobamos que las opciones **Generar el cubo con un origen de datos** y **Generación automática** están seleccionadas y hacemos clic en **Siguiente**.

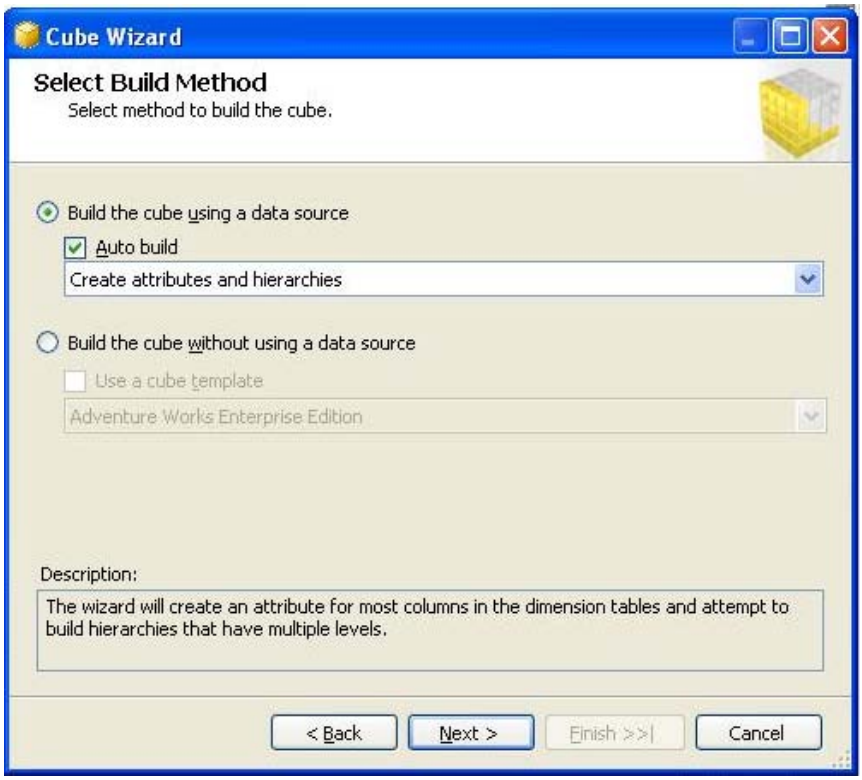


Fig 3.78 Seleccionando método de generación

En la página **Seleccionar vista de origen de datos**, comprobamos que la vista de origen de datos **Expediente** está seleccionada. Hacemos clic en **Siguiente**.

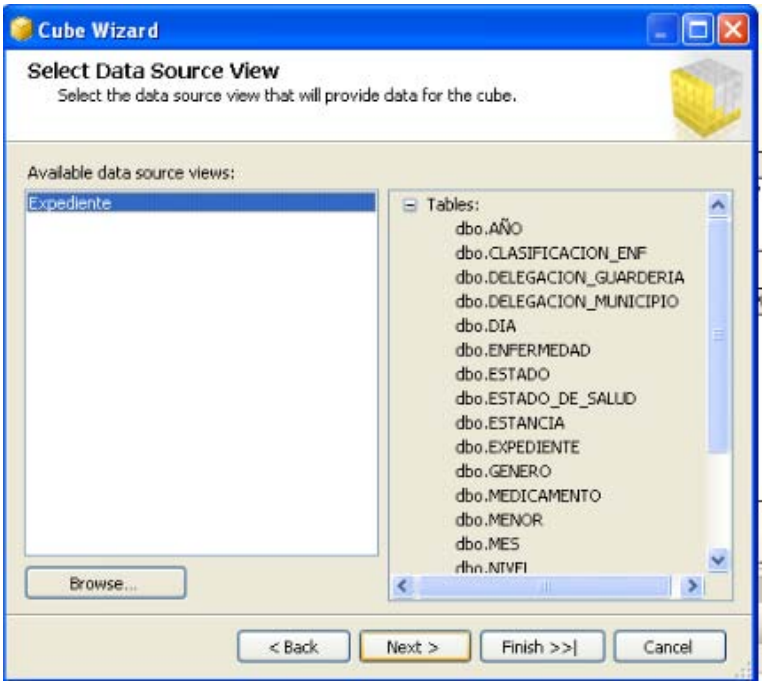


Fig 3.79 Seleccionando vista de origen de datos

En la página **Detectando tablas de hechos y de dimensiones**, hacemos clic en **Siguiente** cuando el asistente haya identificado las tablas de hechos y de dimensiones.

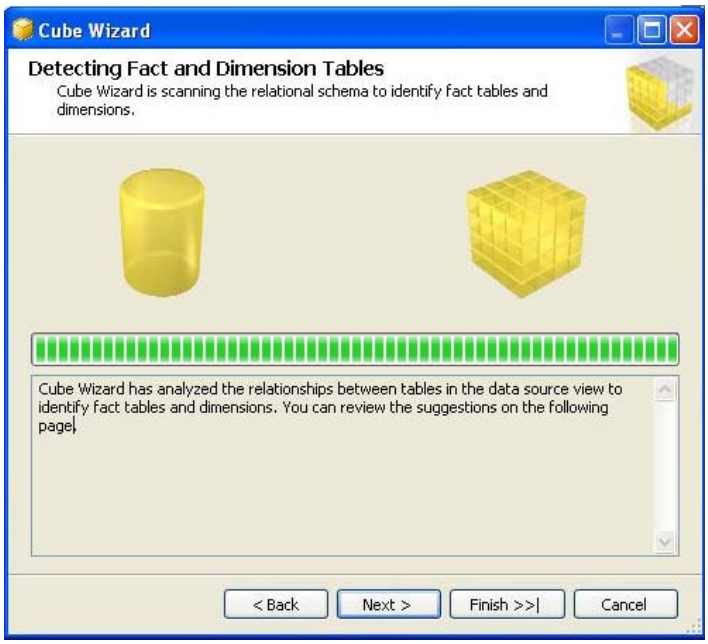


Fig 3.80 Detectando tablas de hechos y de dimensiones

En la página **Detectar tablas de hechos y de dimensiones** se muestran las tablas de hechos y de dimensiones identificadas por el asistente, verificamos que la tabla de hechos es **EXPEDIENTE.**

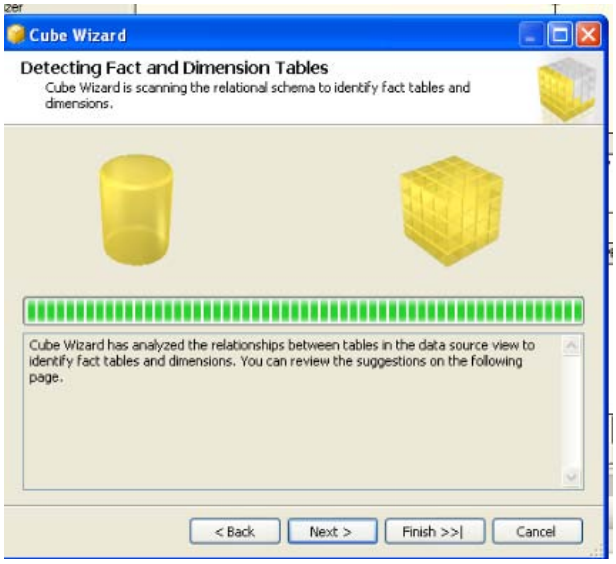


Fig 3.81 Detectando tablas de hechos y dimensiones

En la página **Identificar tablas de hechos y de dimensiones**, seleccione **DIA** en la lista **Tabla de dimensiones de tiempo** y hacemos clic en **Siguiente**.

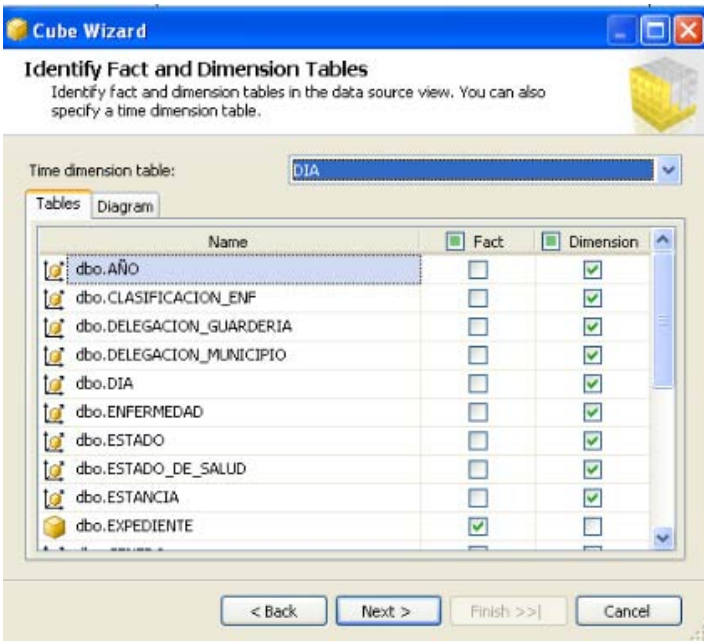


Fig 3.82 Identificar tablas de hechos y de dimensiones

En la página **Seleccionar períodos de tiempo**

- Asigne la propiedad **Year** a la columna **id_año**
- Asigne la propiedad **Half Year** a la columna **id_sem**.
- Asigne la propiedad **Month** a la columna **id_mes**.
- Asigne la propiedad **Day Month** a la columna **id_dia**
- Asigne la propiedad **Date** a la columna **fecha**.

Hacemos clic en **siguiente.**

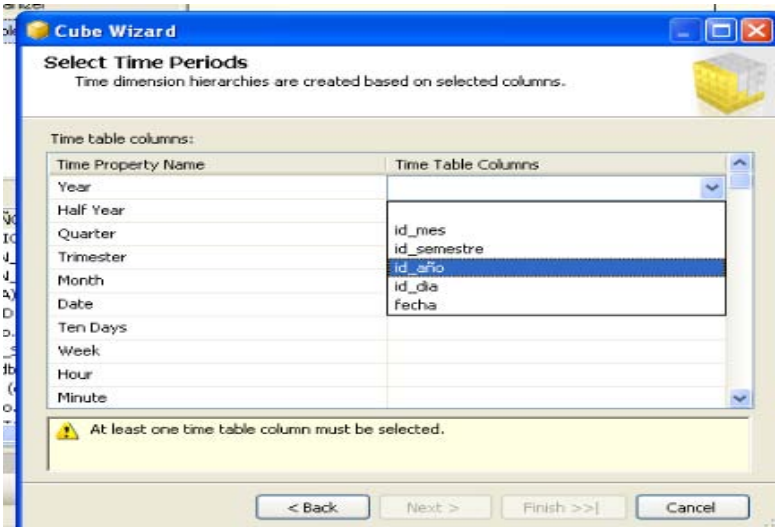


Fig 3.83 Seleccionando periodos de tiempo

En la página **Seleccionar medidas**, revise las medidas seleccionadas en el grupo de medida **Expediente**.

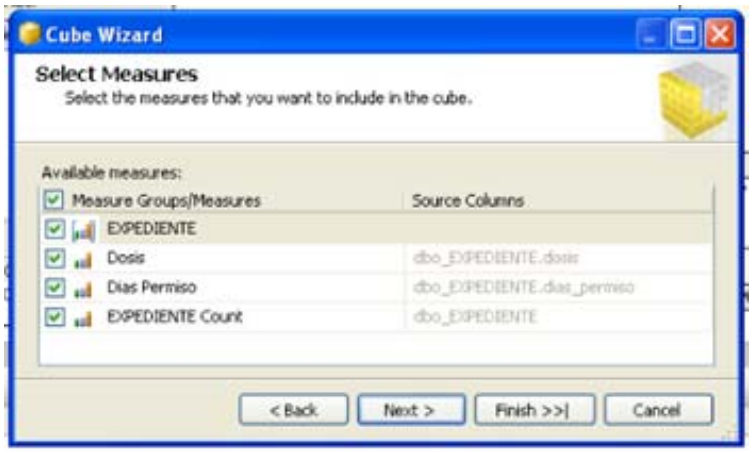


Fig 3.84 Seleccionando medidas

Hacemos clic en siguiente, después en la página **Detectando jerarquías**, hacemos clic en **Siguiente**.

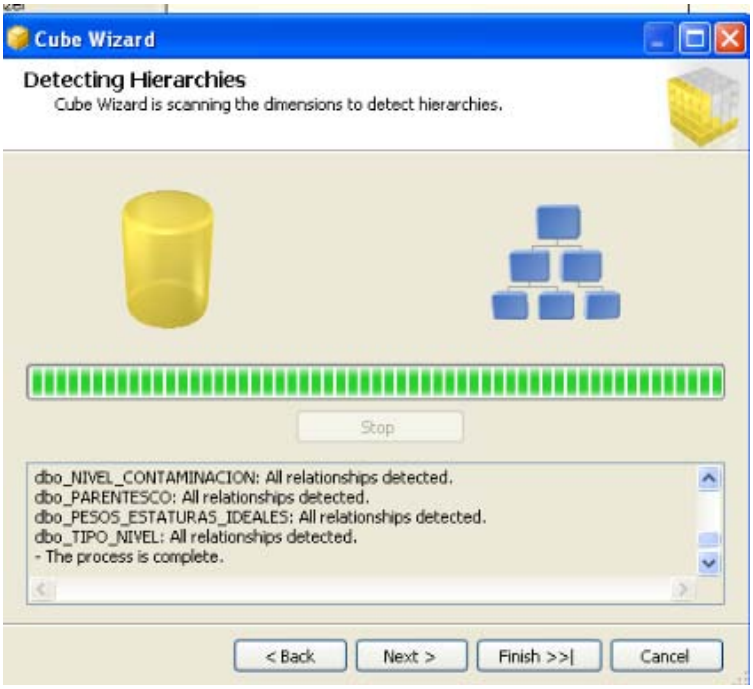


Fig 3.85 Detectando jerarquías

En la página **Revisar las nuevas dimensiones**, revisamos la estructura de la jerarquía de dimensiones de las tres dimensiones expandiendo el control de árbol para ver las jerarquías y los atributos que el asistente ha detectado para cada dimensión.

Fig 3.86 Revisando las nuevas dimensiones

Fig 3.87 Pantalla final del wizard

Como podemos ver las dimensiones mostradas son correctas, entonces hacemos clic en **finalizar**

Una vez hecho esto **guardamos Todo**, tenemos nuestro cubo construido.
Después de revisar las propiedades de cubo y dimensiones.

Procedemos a implementar el proyecto para esto:

En el Explorador de soluciones, hacemos clic con el botón secundario en el proyecto **EXPEDIENTE** y, a continuación, hacemos clic en **Propiedades**.

En el nodo **Propiedades de configuración** del panel de la izquierda, hacemos clic en **Implementación**. En la ficha destino cambiamos el nombre del servidor a **localhost\estancia**, como se muestra a continuación:

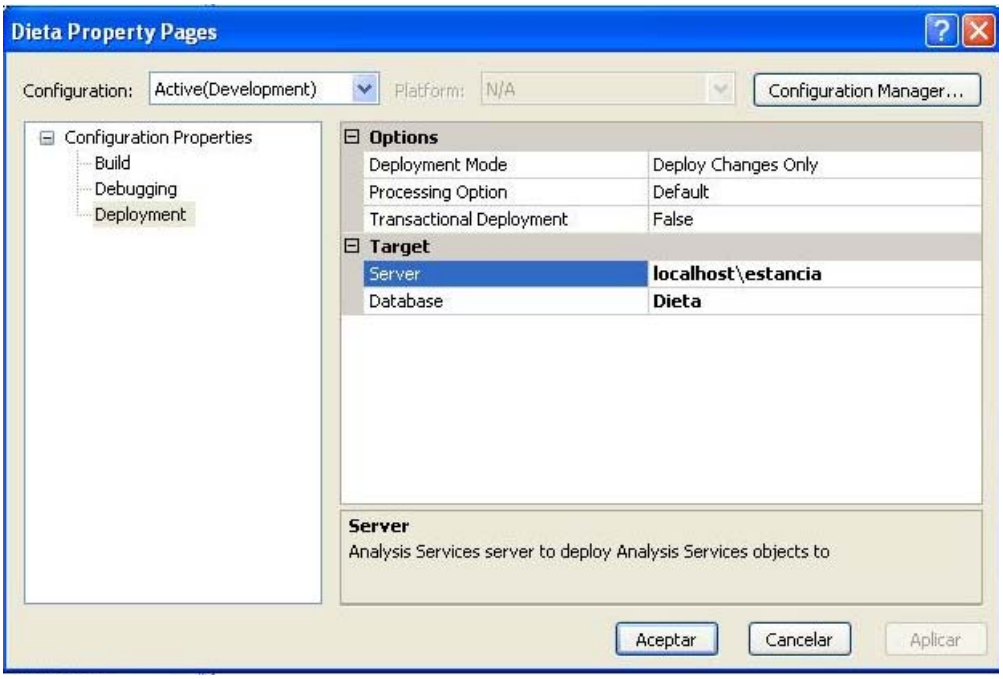


Fig 3.88. Modificando las propiedades de configuración del servidor

Y, a continuación, hacemos clic en **Implementar.**

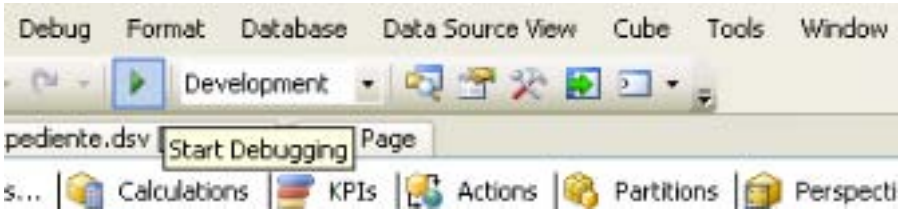


Fig 3.89 Implementando

Y esperamos que se implemente nuestro proyecto finalmente podremos observar que se completo exitosamente.

Fig 3.90 Mensaje de proceso exitoso

Cambiamos al Diseñador de cubos de BI Development Studio haciendo clic en la ficha del diseñador del cubo Expediente. Seleccionamos la ficha **Examinador** y hacemos clic en **Volver a conectar** en la barra de herramientas del diseñador. Alternativamente, hacemos clic en el vínculo **Haga clic aquí para volver a intentar cargar el examinador** que aparece en el centro del panel del examinador.

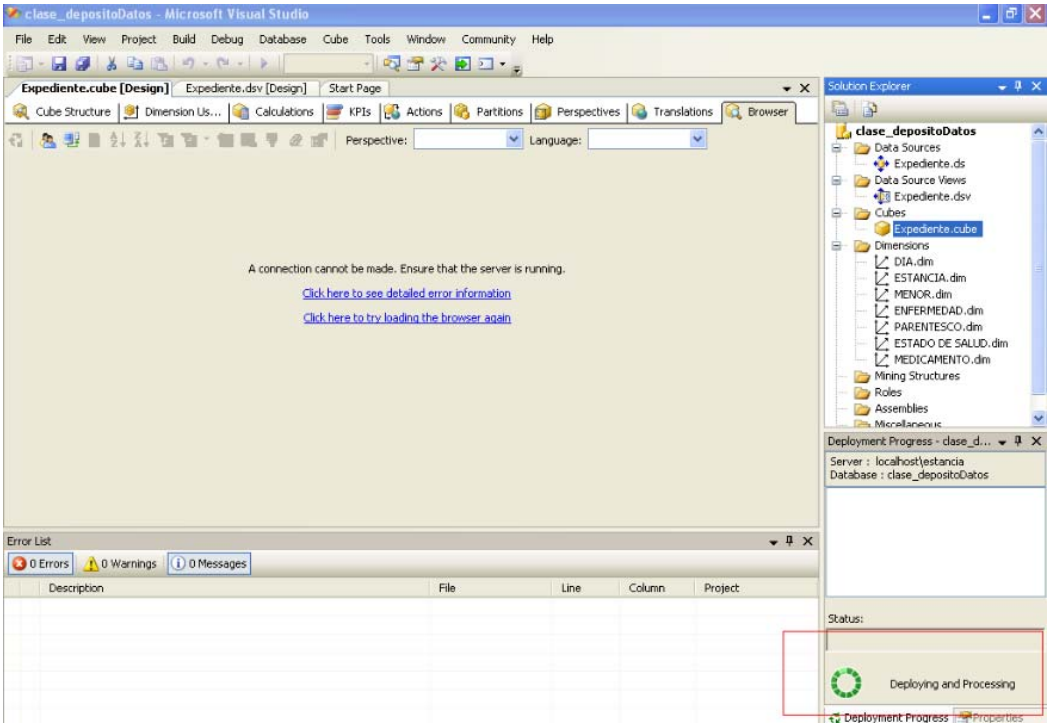


Fig 3.91 Panel del examinador

En el panel izquierdo del diseñador se muestran los metadatos del cubo Expediente. Podemos ver que las opciones **Perspectiva** e **Idioma** están disponibles en la barra de herramientas de la ficha **Examinador**. También vemos que la ficha **Examinador** incluye dos paneles a la derecha del panel **Metadatos**: el superior es el panel **Filtro** y el inferior es el panel **Datos**.
En la imagen siguiente aparecen resaltados los paneles individuales en el Diseñador de cubos.

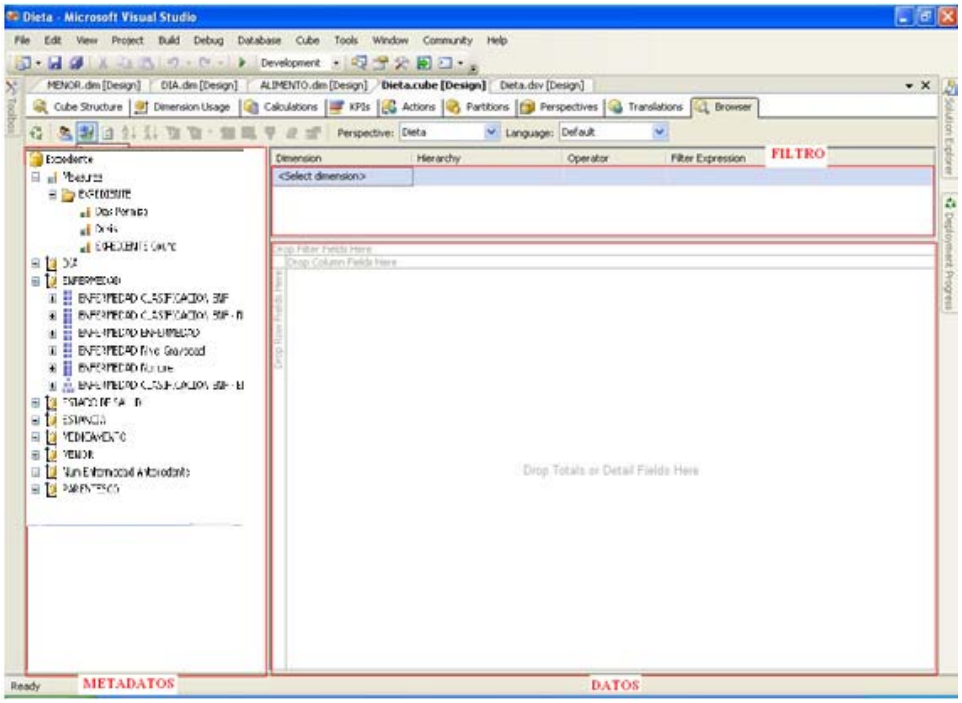


Fig 3.92 Paneles individuales en el Diseñador de cubos

Ahora procedemos a consultar la información más relevante de acuerdo a las necesidades presentadas inicialmente.

Primero expandimos **Measures,** expandimos **Expediente** y arrastramos la medida **Dosis** a el panel de datos en el área **Coloque Campos de totales o campos detallados aquí.**
Arrastramos la dimensión **NIVEL** en el área **Coloque campos de fila aquí**, Arrastramos la dimensión **CLASIFICACION ENF y ENFERMEDADES** en el área **Coloque campos de columnas aquí.**

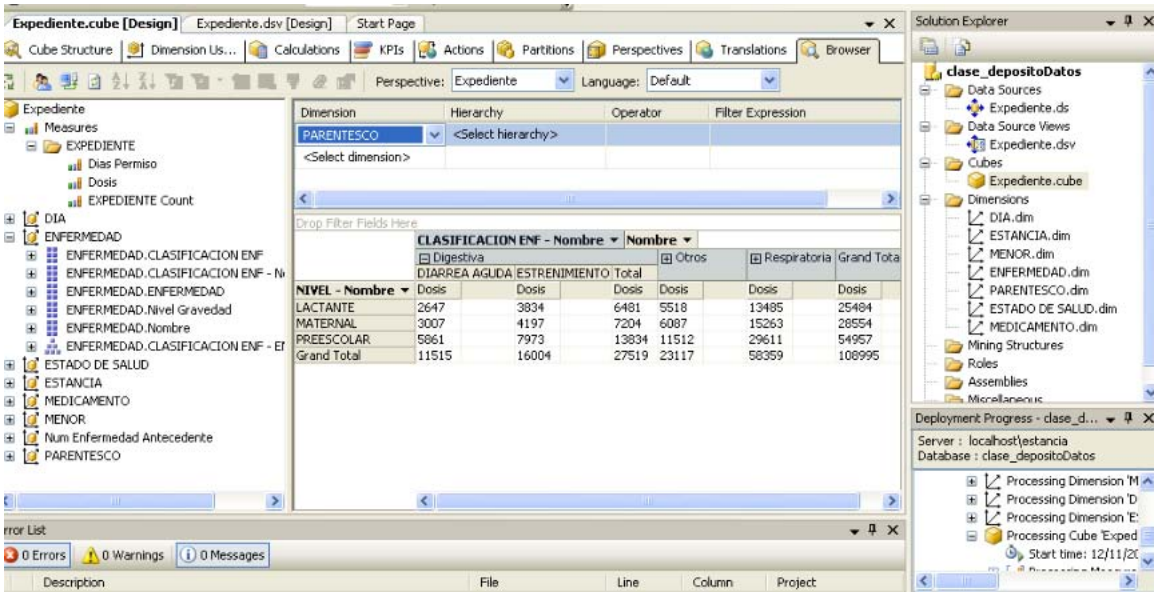


Fig 3.93 Consultando información

Después arrastramos la dimensión **DIA** en el panel filtro y en operador seleccionamos **equal** y en expresión de filtro seleccionamos **el segundo semestre del 1958.**

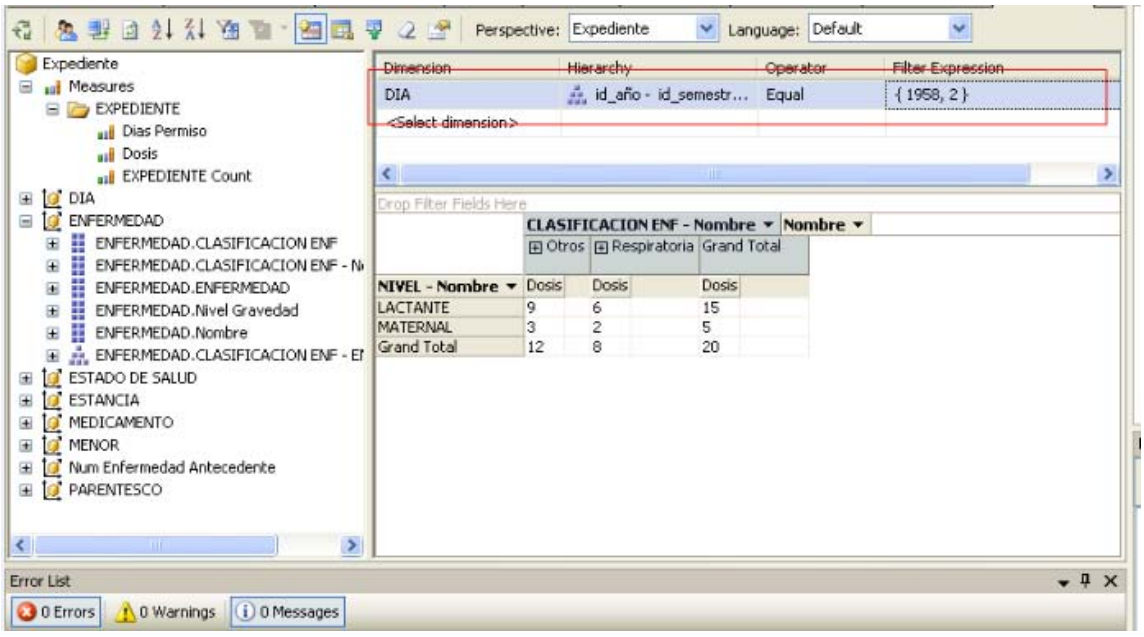


Fig 3.94 Refinando los resultados

Reportes

Después de realizar los cubos Dieta y Expediente, podemos ver su gran utilidad a la hora de responder de manera rápida y efectiva a las necesidades plasmadas en nuestro diagrama jerárquico funcional OLAP.

Expediente

2.1.1 Reportar los niños que faltan más por alguna enfermedad

En este caso seleccionamos los niños que en su expediente tienen mayor número de días permiso en el mes de diciembre del año 1970

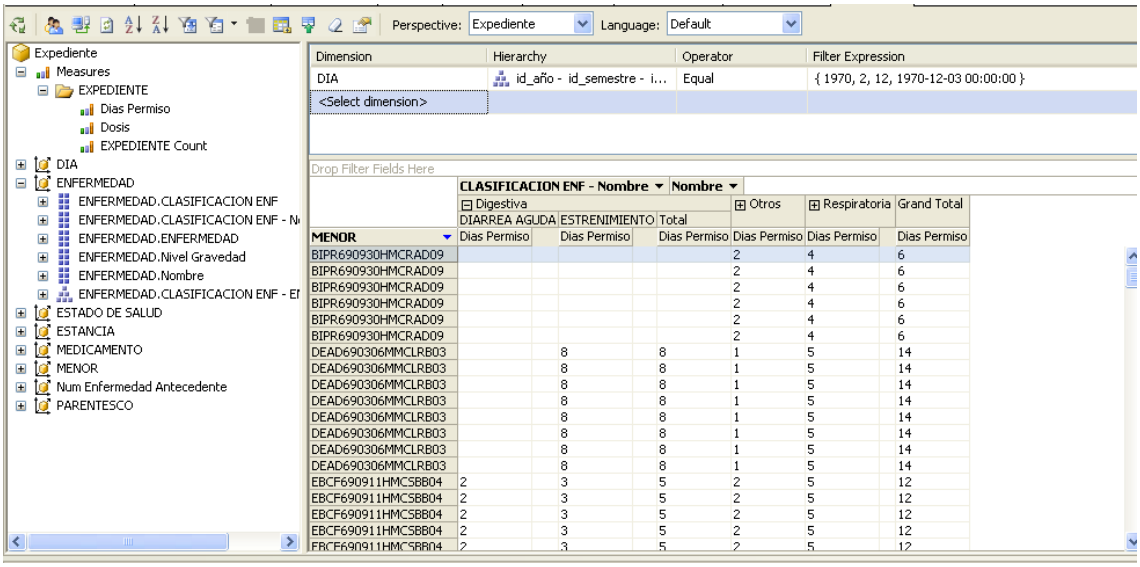


Fig 3.95 Reportar los niños que faltan más por alguna enfermedad

2.1.2 Reportar niños que se enferman más

En este caso seleccionamos los niños que es su expediente tienen mayor cantidad de veces que se enferman en nivel maternal y en el segundo semestre de 1991

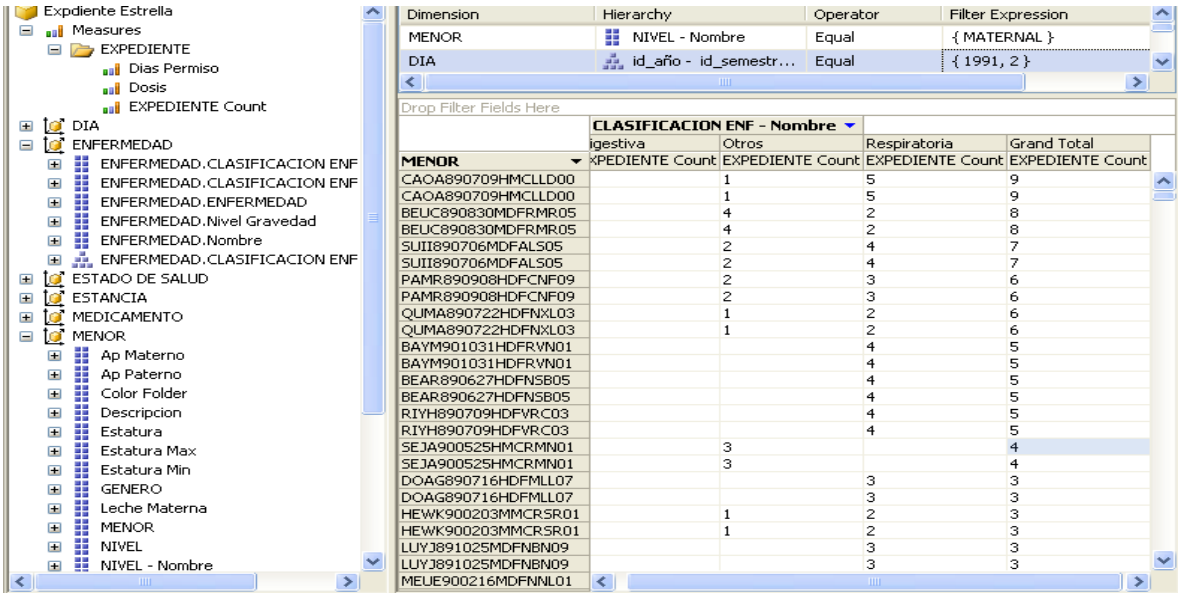


Fig 3.96 Reportar niños que se enferman más

2.1.3 Reportar las enfermedades más comunes según el nivel del infante

A continuación podemos ver las enfermedades que aquejan a los infantes de acuerdo al nivel en que se encuentren en la estancia

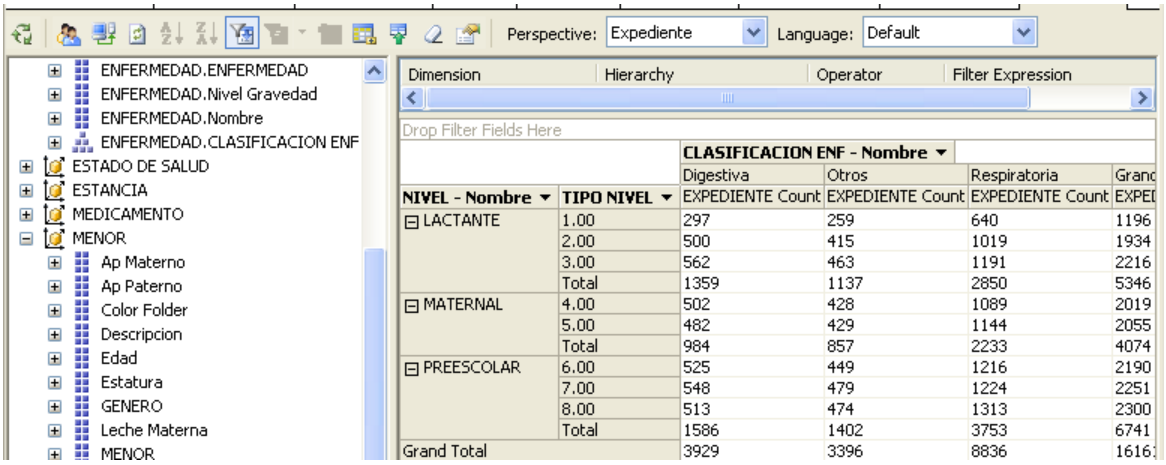


Fig 3.97 Reportar las enfermedades más comunes según el nivel del infante 1/3

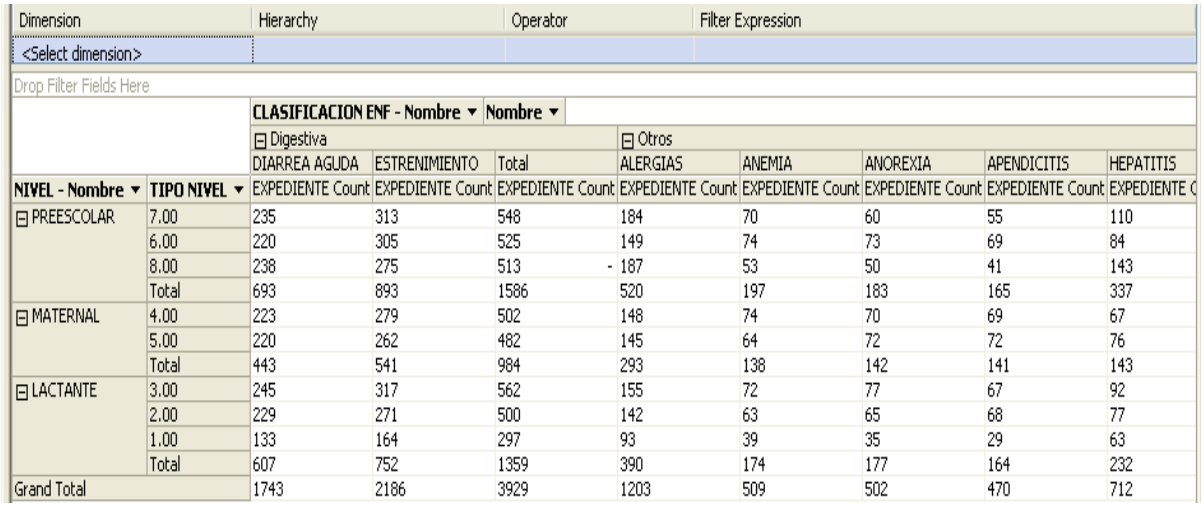


Fig 3.98 Reportar las enfermedades más comunes según el nivel del infante 2/3

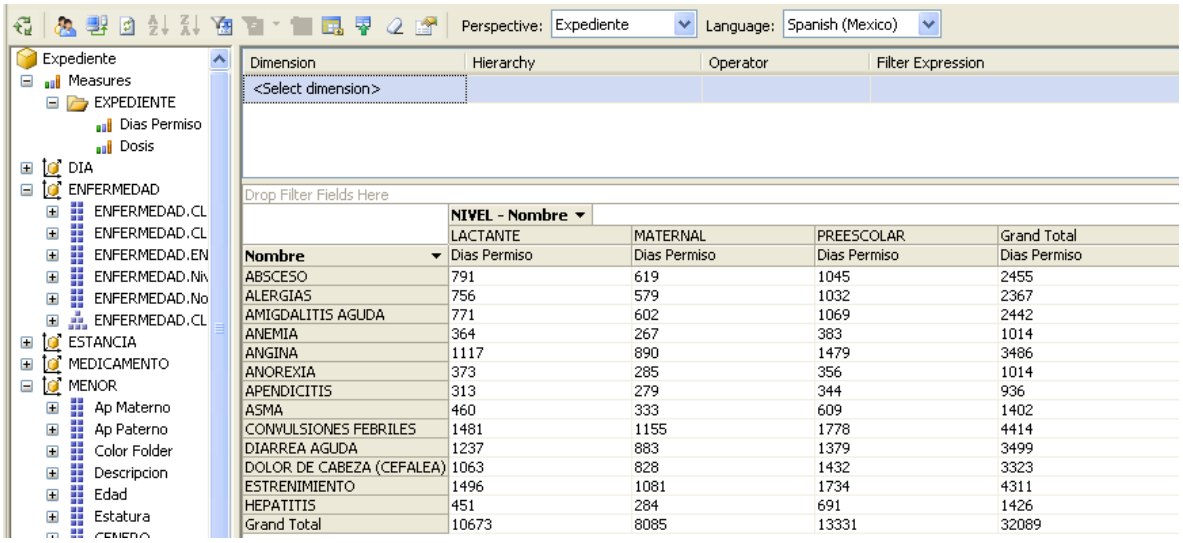


Fig 3.99 Reportar las enfermedades más comunes según el nivel del infante 3/3

2.1.4 Reportar enfermedades que sean consideradas como graves según el nivel

En este caso seleccionamos el nivel de los infantes y el número de días permiso que tuvieron a causa de alguna enfermedad, y filtramos las enfermedades que estén consideradas como graves

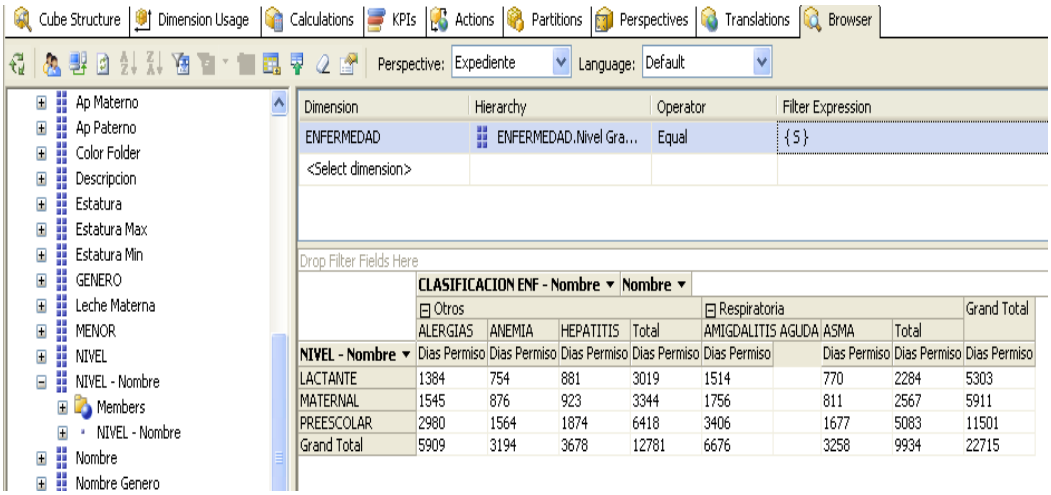


Fig 3.100 Reportar enfermedades que sean consideradas como graves según el nivel

2.1.5 Reportar estancias donde existe mayor nivel de contaminación

En este caso seleccionamos la delegación o municipio donde se encuentra la guardería , además del número de estancia y la descripción del nivel de contaminación, filtramos las estancias donde el nivel de contaminación se mala y muy mala.

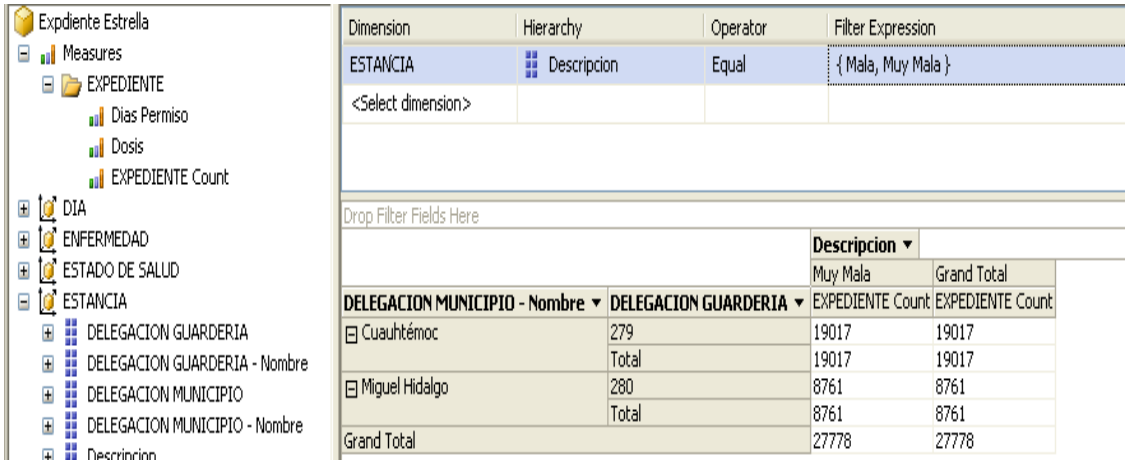


Fig 3.101 Reportar estancias donde existe mayor nivel de contaminación

Dieta

2.2.1 Reportar niños que no comen o solo prueban alimentos

En este caso hicimos una selección de los niños, que No Comen (Consumo = NC) o solo prueban (Consumo =PR), del mes de enero de 2008. Se muestra a continuación.

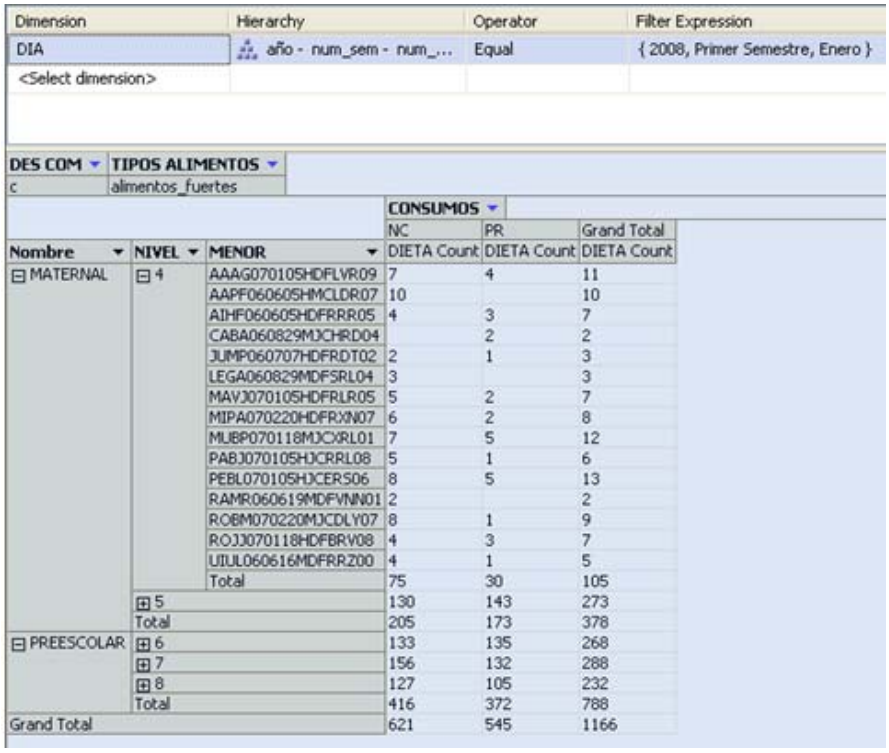


Fig 3.102 Reportar niños que no comen o solo prueban alimentos

2.2.1.1 Reportar niños que no comen o sólo prueban verduras por nivel

Para poder distinguir a los infantes que no comen bien o sólo prueban las verduras, necesitamos seleccionar solo las verduras de la dimensión Alimento-Alimento original-nombre= verduras.

Dimension	Hierarchy	Operator	Filter Expression
DIA	año - num_sem - num_...	Equal	{ 2008, Primer Semestre, Enero }
<Select dimension>			

Nombre
Verduras

			CONSUMOS		
			NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count
LACTANTE	2		89	77	166
	3		120	98	218
	Total		209	175	384
MATERNAL	4	AAAG070105HDFLVR09	1		1
		AAPF060605HMCLDR07		1	1
		CABA060829MJCHRD04		1	1
		MAVJ070105HDFRLR05	1		1
		MIPA070220HDFRXN07		1	1
		PABJ070105HJCRRL08	1		1
		RAMR060619MDFVNN01		1	1
		Total	3	4	7
	5		23	22	45
	Total		26	26	52
PREESCOLAR	6		21	21	42
	7		19	27	46
	8		8	18	26
	Total		48	66	114
Grand Total			283	267	550

Fig 3.103 Reportar niños que no comen o sólo prueban verduras por nivel

2.2.1.2 Reportar niños que no desayunan bien

Para visualizar los infantes que no desayunan bien, discriminamos las comidas, seleccionando des_com =d y consumo=NC. para el mismo periodo.

DES COM
d

			CONSUMOS	
			NC	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count
LACTANTE	2		475	475
	3		435	435
	Total		910	910
MATERNAL	4	AAAG070105HDFLVR09	13	13
		AAPF060605HMCLDR07	9	9
		AIHF060605HDFRRR05	9	9
		CABA060829MJCHRD04	5	5
		JUMP060707HDFRDT02	2	2
		LEGA060829MDFSRL04	5	5
		MAVJ070105HDFRLR05	19	19
		MIPA070220HDFRXN07	11	11
		MUBP070118MJCXRL01	10	10
		PABJ070105HJCRRL08	12	12
		PEBL070105HJCERS06	11	11
		RAMR060619MDFVNN01	8	8
		ROBM070220MJCDLY07	15	15
		ROJJ070118HDFBRV08	13	13
		UIUL060616MDFRRZ00	12	12
		Total	154	154
	5		717	717
	Total		871	871
PREESCOLAR	6		780	780
	7		910	910
	8		417	417
	Total		2107	2107

Fig 3.104 Reportar niños que no desayunan bien

2.2.2 Reportar alimentos de menor consumo

Para esta consulta se necesita ver qué tipo de alimento son los de menor consumo, reducimos el tiempo a un mes (ENERO), y después haremos una consulta por tipo de alimento para obtener un mejor panorama de lo que queremos.

Para Jugos_Aguas

TIPOS ALIMENTOS: jugos_aguas			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 2		176	1164	175	191	1706
	⊞ 3		209	1495	215	237	2156
	Total		385	2659	390	428	3862
⊟ MATERNAL	⊞ 4		66	498	79	57	700
	⊞ 5		282	1933	303	286	2804
	Total		348	2431	382	343	3504
⊟ PREESCOLAR	⊞ 6		286	1763	251	252	2552
	⊞ 7		304	1992	296	284	2876
	⊞ 8		172	792	107	129	1200
	Total		762	4547	654	665	6628
Grand Total			1495	9637	1426	1436	13994

Fig 3.105 Reportar alimentos de menor consumo (Jugos_Aguas)

Para Frutas_yogourth_postres

TIPOS ALIMENTOS: fruta_yogourth_postre			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 2		480	770	286	170	1706
	⊞ 3		635	963	336	222	2156
	Total		1115	1733	622	392	3862
⊟ MATERNAL	⊞ 4		198	362	60	80	700
	⊞ 5		872	1373	283	276	2804
	Total		1070	1735	343	356	3504
⊟ PREESCOLAR	⊞ 6		924	994	384	250	2552
	⊞ 7		1184	1000	284	408	2876
	⊞ 8		483	417	173	127	1200
	Total		2591	2411	841	785	6628
Grand Total			4776	5879	1806	1533	13994

Fig 3.106. Reportar alimentos de menor consumo (Frutas_yogourth_postres)

Para Alimentos_fuertes

TIPOS ALIMENTOS							
alimentos_fuertes							
			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 3		550	300	118	110	1078
	Total		550	300	118	110	1078
⊟ MATERNAL	⊞ 4		321	225	101	53	700
	⊞ 5		1390	859	271	284	2804
	Total		1711	1084	372	337	3504
⊟ PREESCOLAR	⊞ 6		1006	762	407	377	2552
	⊞ 7		1175	973	314	414	2876
	⊞ 8		460	333	184	223	1200
	Total		2641	2068	905	1014	6628
Grand Total			4902	3452	1395	1461	11210

Fig 3.107 Reportar alimentos de menor consumo (Alimentos_fuertes)

Para Cereales y leguminosas

TIPOS ALIMENTOS							
cereales_leguminosa							
			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 2		276	300	187	90	853
	⊞ 3		760	879	288	229	2156
	Total		1036	1179	475	319	3009
⊟ MATERNAL	⊞ 4		288	288	48	76	700
	⊞ 5		1116	1120	283	285	2804
	Total		1404	1408	331	361	3504
⊟ PREESCOLAR	⊞ 6		1038	1016	266	232	2552
	⊞ 7		978	1136	315	447	2876
	⊞ 8		474	472	131	123	1200
	Total		2490	2624	712	802	6628
Grand Total			4930	5211	1518	1482	13141

Fig 3.108 Reportar alimentos de menor consumo (Cereales y leguminosas)

Para Verduras y Ensaladas

TIPOS ALIMENTOS							
verduras_ensaladas							
			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 3		335	221	309	213	1078
	Total		335	221	309	213	1078
⊟ PREESCOLAR	⊞ 6		374	132	505	265	1276
	⊞ 7		415	147	583	293	1438
	⊞ 8		117	59	300	124	600
	Total		906	338	1388	682	3314
Grand Total			1241	559	1697	895	4392

Fig 3.109 Reportar alimentos de menor consumo (Verduras y Ensaladas)

Para Sopas

TIPOS ALIMENTOS: sopas			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 2		333	354	89	77	853
	⊞ 3		436	424	120	98	1078
	Total		769	778	209	175	1931
⊞ MATERNAL			721	678	183	170	1752
⊟ PREESCOLAR	⊞ 6		510	507	127	132	1276
	⊞ 7		572	576	137	153	1438
	⊞ 8		165	259	56	120	600
	Total		1247	1342	320	405	3314
Grand Total			2737	2798	712	750	6997

Fig 3.110 Reportar alimentos de menor consumo (Sopas)

Para Variaciones_leche

TIPOS ALIMENTOS: variaciones_leche			CONSUMOS				
			CM	CT	NC	PR	Grand Total
Nombre	NIVEL	MENOR	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
⊟ LACTANTE	⊞ 1		361	1344			1705
	⊞ 2		92	581	95	85	853
	⊞ 3		231	727		120	1078
	Total		684	2652	95	205	3636
⊟ MATERNAL	⊞ 4		143	94	32	81	350
	⊞ 5		596	538	142	126	1402
	Total		739	632	174	207	1752
⊟ PREESCOLAR	⊞ 6		374	530	131	241	1276
	⊞ 7		453	393	286	306	1438
	⊞ 8		266	112	120	102	600
	Total		1093	1035	537	649	3314
Grand Total			2516	4319	806	1061	8702

Fig 3.111 Reportar alimentos de menor consumo (Variaciones_leche)

Al hacer una consulta de la siguiente manera, en donde limitamos la dimensión tiempo al año 2008, y seleccionando un alimento en particular:

Dimension	Hierarchy	Operator	Filter Expression
DIA	año - num_sem - num_...	Equal	{ 2008 }
<Select dimension>			

Nombre: Albondiga en Salsa	CONSUMOS				
	CM	CT	NC	PR	Grand Total
	DIETA Count	DIETA Count	DIETA Count	DIETA Count	DIETA Count
s Here	89	59	24	17	189

Fig 3.112 Limitando al año 2008

Como vemos si consideramos el total de los consumos, para este caso, son:
Comió Mitad: 89, Comió Todo: 59, No Comió: 24, Probó:17

Ahora al consular dicho resultado para cada alimento tenemos que:

Tipo_Alimento	Nombre	CM	CT	NT	CM	TOTAL
7	Avena	1565	1728	653	536	4482
4	Hotcakes	1282	1047	376	402	3107
7	Arroz	1072	1183	298	363	2916
1	Naranja_J	263	1820	297	277	2657
5	Verduras	1090	1127	283	267	2767
1	Mandarina_J	272	1841	276	261	2650
4	Cornflakes	1034	1079	274	300	2687
6	Nopales	140	76	262	120	598
1	Pina_J	300	1819	261	278	2658
6	Pepino	201	73	261	123	658
2	Uvas	771	911	258	256	2196
2	Coctel	867	1019	256	274	2416
6	Zanahorias	184	75	250	146	655
6	Lechuga	172	98	248	136	654
1	Uva_J	281	1718	244	262	2505
6	Brocoli	180	69	228	126	603
6	Papas	169	79	228	125	601
6	Ejotes	195	89	220	119	623
3	Pan Francés	766	462	214	284	1726
2	Papaya	353	493	172	112	1130
2	Melon	352	453	168	121	1094
7	Natural o Formula	684	2206	162	200	3252
2	Yogurt	291	370	157	105	923
2	Platano	295	360	153	113	921
4	Pan	533	554	133	174	1394
5	Pasta con espinacas	310	361	93	107	871
7	Canela	231	261	91	100	683
2	Sandia	208	273	91	72	644
7	Atole	240	271	88	124	723
5	Zanahoria	346	313	87	92	838
5	Pasta	336	325	86	100	847
5	Calabaza	344	345	84	95	868
2	Durazno	211	309	84	69	673
7	Chocolate	219	230	83	104	636
5	Papa	311	327	79	89	806
7	fresa	193	275	77	110	655
3	Molletes	216	186	77	74	553
7	Cajeta	240	273	74	106	693
7	Caramelo	238	265	73	104	680
2	Manzana al horno	193	266	73	64	596
3	Lentejas	293	268	71	71	703
2	Pina	215	304	70	64	653
1	Limon_A	67	451	65	60	643
2	Naranja	153	182	62	42	439
2	Tuna	163	180	58	45	446
1	Mandarina_A	49	286	57	39	431
2	Gelatina	147	166	55	36	404
4	Garbanzo	168	146	54	41	409
2	Galletas	162	196	53	56	467
3	Caldo tlapeño	120	122	53	39	334
3	Pollo	140	90	52	43	325
1	Guayaba_A	41	344	52	38	475

2	Guayaba	193	181	49	48	471
2	Mandarina	202	216	47	56	521
3	Tortitas de papa	163	132	46	38	379
3	Papas al horno	149	121	44	48	362
3	Papas con jamon	149	121	44	48	362
3	Higado empanizado	148	119	44	31	342
3	Salchichas a la mexicana	163	108	43	44	358
3	Ensalada de pepino	149	107	41	43	340
4	Frijoles	172	164	41	38	415
4	Tortilla	172	157	40	45	414
3	Ensalada de jamon	157	100	38	40	335
3	Huevo	157	126	37	43	363
3	Sopa de verduras	154	104	36	45	339
3	Bistec	66	64	36	18	184
3	Arroz blanco	145	114	35	40	334
1	Melón_A	40	310	34	50	434
3	Crema de espinacas	148	97	34	43	322
3	Crema brocoli	174	115	33	46	368
3	Quesadillas de picadillo	163	98	31	49	341
3	Carne Campestre	59	62	30	23	174
1	Jamaica_A	27	128	26	23	204
3	Pescado	79	65	26	17	187
3	Salpicon	93	73	26	14	206
1	Horchata_A	31	163	25	18	237
3	Albondiga en Salsa	89	59	24	17	189
3	Alambre de Pollo	82	67	22	17	188
1	Sandía_A	21	153	21	30	225
3	Pollo Empanizado	90	63	21	23	197
1	Pina_A	22	130	20	22	194
1	Papaya_A	24	167	19	25	235
3	Chilaquiles verdes	62	33	16	18	129
1	Tuna_A	32	154	15	28	229
3	Enchiladas suizas	44	31	15	15	105
1	Tamarindo_A	25	153	14	25	217
3	Alambre de Res	72	48	13	22	155

Dicha Tabla fue ordenada de acuerdo a al consumo, como podemos ver NC es la que prevalece y de esta manera podemos ver que alimentos son los que menos se consumen.

Propuesta de soluciones a los reportes:

Expediente

2.1.1 Reportar los niños que faltan más por alguna enfermedad

En este caso se identifica la enfermedad que causa que los menores falten y se harán campañas de salud preventiva, campañas de higiene personal, aplicación de vitaminas y de medicamentos para la prevención de enfermedades respiratorias. Además de contar con los medicamentos adecuados para tratar estas enfermedades que son las más comunes

2.1.2 Reportar niños que se enferman más

En este caso identificamos a los niños que son más enfermizos, y se propone aplicar medicamentos para prevención de enfermedades, campañas de salud preventivas y de higiene personal.

2.1.3 Reportar las enfermedades más comunes según el nivel del infante

Se identifico las enfermedades que son más comunes según el nivel en cada una de las estancias, se propone tener campañas de prevención de dichas enfermedades, tener limpios los lugares de convivencia de los niños, identificar las enfermedades de manera temprana para evitar contagios.

2.1.4 Reportar enfermedades que sean consideradas como graves según el nivel

Se identificaron las enfermedades que son consideradas como enfermedades graves (Alergias, Anemia, Hepatitis, Amigdalitis Aguda y Asma), y se propone contar con conocimientos del personal de cómo tratar a las niños si se presenta una eventualidad, además de que los niños estén en supervisión frecuente por el médico de la estancia.

2.1.5 Reportar estancias donde existe mayor nivel de contaminación

Se identificaron las estancias que cuentan con un nivel de contaminación malo, las cuales se ubican en las delegaciones Cuauhtémoc y Miguel Hidalgo. Se propone que estas estancias no tengan actividades al aire libre.

Dieta

2.2.1 Reportar niños que no comen o solo prueban alimentos

En este caso la solución sería difundir a los padres y maestras de la estancia infantil la lista de los niños que no comen o que sólo prueban alimentos para así poder actuar oportunamente ofreciendo alternativas a los alimentos o a la forma de cocinarlos, para así evitar posibles síntomas de desnutrición.

2.2.1.1 Reportar niños que no comen o sólo prueban verduras por nivel

De igual manera convendría informar a los padres también para que ellos puedan actuar en su casa, en tanto que en la estancia podríamos reestructurar los menús de tal manera que puedan ofrecer a los niños variedad de alimentos y alternativas que contengan verduras por su importancia nutricional (como por ejemplo preguntar al infante ¿prefieres brócoli o zanahoria para el almuerzo?) de tal manera que incremente el gusto de verduras en los niños.

2.2.1.2 Reportar niños que no desayunan bien

También habría que informar a los padres o tutores y personal de la estancia para que puedan tomar acciones, ya que como sabemos es la comida más importante del día, podrían proponer acciones como ofrecer variedad y alternativas de alimentos para de esta manera estimular el apetito de los niños y hacer que los infantes adquirirán el hábito de desayunar bien y a si evitar que los niños presenten desnutrición.

2.2.2 Reportar alimentos de menor consumo

Como podemos observar la preferencia por la comida chatarra es mayor en lo infantes , así que la solución sería reestructurar el menú de tal manera que se le pueda ofrecer al infante una variedad de frutas, vegetales, leche, carne, queso, cereales, panes y postres, y para hacerlo más interesante se podría servirse por ejemplo la carne cortada en tiras, o corte los vegetales y frutas como anillos, de esta manera llamar su atención y así estimular su apetito por aquellos alimento que le ofrecen mayor porcentaje nutricional que la comida chatarra.

CONCLUSIÓN CAPÍTULO 3

En este capítulo pudimos estudiar y aplicar a detalle la teoría vista en nuestro curso de “Depósitos de datos”, ya que vivimos de cerca los procesos de Análisis, Diseño y Construcción, incluyendo además la limpieza y extracción de datos para la toma de decisiones a partir de datos históricos.

El uso de SQL Server y en específico “SQL Server Business Intelligence Development Studio” nos permitió ver desde cero el proceso de pasar de la simple información acumulada, a un sistema que nos facilitara la toma de decisiones, o la simple visualización de nuestra información, algo que en ocasiones se vuelve muy complicado tomando en cuenta que tratamos con grandes volúmenes de información.

Lo aplicado en este capítulo es una parte fundamental para la extracción de conocimiento útil.

Capítulo 4
Minería de Datos.

Introducción KDD

(Introducción a la Minería de Datos)

La gran cantidad de datos almacenados en los sistemas de información de instituciones, empresas, gobiernos y particulares han pasado de ser solo "el resultado histórico de los sistemas de información" a ser una "materia prima" que hay que explotar para obtener el verdadero "producto elaborado", el conocimiento.
La estadística fue la primera ciencia que consideró los datos como su materia prima, pero las nuevas necesidades y, en particular, las nuevas características de los datos (en volumen y tipología) hacen que se integren numerosas disciplinas heterogéneas que se conocen como "minería de datos".

Nuevas necesidades

A parte de su función de "memoria de la organización", la información histórica es útil para explicar el pasado, entender el presente y predecir la información futura.
En muchas situaciones, el método tradicional de convertir los datos en conocimiento consiste en un análisis e interpretación realizado de forma manual, actividad que resulta ser lenta, cara y altamente subjetiva.
Hasta no hace mucho el análisis de los datos de una base de datos se realizaba mediante consultas efectuadas con lenguajes de consulta, como el SQL, y se producía sobre la base de datos operacional, es decir, OLTP (On-Line Transaction Processing).
No obstante, esta manera de actuar sólo permitía generar información resumida (informes), poco flexible y poco escalable a grandes volúmenes de datos.
La tecnología de Bases de Datos respondió con una nueva arquitectura llamada "almacén de datos", que es un repositorio de fuentes heterogéneas de datos, integrados y organizados bajo un esquema unificado.
Esta tecnología incluye operaciones de procesamiento analítico en línea (On-Line Analytical Processing, OLAP).
Sin embargo, a pesar de que las herramientas OLAP soportan cierto análisis descriptivo y de sumarización, no generan reglas y patrones, es decir conocimiento que pueda ser aplicado a otros datos.
Todos estos problemas y limitaciones de las aproximaciones clásicas han hecho surgir la necesidad de una nueva generación de herramientas y técnicas para soportar la extracción de conocimiento útil y que se engloban bajo la denominación de minería de datos.

Minería de datos

Minería de datos es el proceso de extraer conocimiento útil y comprensible, previamente desconocido, desde grandes cantidades de datos almacenados en distintos formatos.
La tarea fundamental de la Minería de datos es encontrar modelos a partir de los datos, para ayudar a tomar decisiones más seguras.
La minería de datos se distingue porque no obtiene información extensional (datos) sino intencional (conocimiento) y, además, el conocimiento no es, un modelo preestablecido o intuido por el usuario, sino que es un modelo novedoso y original.

Tipos de Datos

La minería de datos puede aplicarse a cualquier tipo de información, aunque las técnicas utilizadas podrían ser diferentes para cada una de ellas.

Se puede aplicar a datos estructurados provenientes de Bases de Datos relacionales, espaciales, temporales, textuales y multimedia y datos no estructurados provenientes de la web o de otros tipos de repositorios de documentos.

Bases de Datos relacionales

Aunque las Bases de Datos relacionales son la fuente de datos para la mayoría de aplicaciones de minería de datos, la mayoría de las técnicas de minería de datos no son capaces de trabajar con toda la base de datos, sino que sólo son capaces de tratar con una sola tabla a la vez (llamada vista minable)
Dentro de este tipo de bases podemos distinguir principalmente dos tipos de atributos, numéricos (valores enteros o reales) y categóricos (toman valores de un conjunto finito y preestablecido de categorías).

Otros tipos de Bases de Datos que contienen datos complejos

Existen otros tipos de Bases de Datos que contienen algunos tipos de datos complejos.
- Las Bases de Datos espaciales. Incluyen datos geográficos, imágenes médicas, redes de transporte, etc. La minería de datos permite encontrar patrones, como las características de las casas en una zona montañosa o la planificación de nuevas líneas de metro.
- Las Bases de Datos temporales. Almacenan datos relacionados con el tiempo, las técnicas de minería de datos pueden utilizarse para encontrar características de la evolución o las tendencias del cambio.
- Las Bases de Datos documentales. Contienen descripciones para los objetos (documentos de texto). Pueden contener: Documentos no estructurados (como una biblioteca digital de novelas), Documentos semi-estructurados (si se puede extraer la información por partes, con índices, etc.), Documentos estructurados (como una base de datos de fichas bibliográficas.

Para la World Wide Web la diversidad de información que contiene la World Wide Web hace que la minería web se organice en torno a 3 categorías:
- Minería del contenido. Encuentra patrones de los datos de las páginas web.
- Minería de la estructura. Hipervínculos y URL's .
- Minería del uso que hace el usuario de las páginas web (navegación).

Tipos de modelos

El conocimiento que se extrae con la minería de datos puede ser en forma de relaciones, patrones o reglas inferidos de los datos y desconocidos, o bien en forma de una descripción más concisa (es decir, un resumen de los mismos). Estas relaciones o resúmenes constituyen el modelo de los datos analizados. Existen muchas formas diferentes de representar los modelos y cada una de ellas determina el tipo de técnica que puede usarse para inferirlos.
En la práctica, los modelos pueden ser de 2 tipos: Predictivos y Descriptivos. Los modelos predictivos pretenden estimar valores futuros o desconocidos de variables de interés, que denominamos variables objetivo o dependientes, usando otras variables o campos de la base de datos, a las que nos referiremos como variables independientes o predictivas.

KDD

Se define el KDD (knowledge Discovery in Databases, KDD) como "el proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y, en última instancia, comprensibles a partir de los datos".
El conocimiento extraído debe ser válido (los patrones deben seguir siendo precisos para datos nuevos), novedoso, potencialmente útil (la información debe conducir a acciones que reporten algún tipo de beneficio) y comprensible.

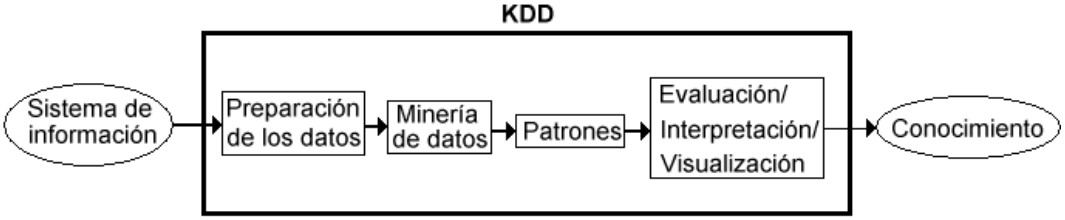


Fig. 4.1 Sistema KDD

Los sistemas de KDD permiten la selección, limpieza, transformación y proyección de los datos; analizar los datos para extraer patrones y modelos adecuados; evaluar e interpretar los patrones para convertirlos en conocimiento; consolidar el conocimiento resolviendo posibles conflictos con conocimiento previamente extraído; y hacer el conocimiento disponible para su uso.
El KDD es el proceso global de descubrir conocimiento útil desde las Bases de Datos mientras que la minería de datos se refiere a la aplicación de los métodos de aprendizaje y estadísticos para la obtención de patrones y modelos, siendo una fase del KDD.

Relación con otras disciplinas

El KDD está estrechamente relacionado con otras disciplinas como son: Bases de Datos, La recuperación de información, Estadística, El aprendizaje automático (inteligencia artificial), Los sistemas para la toma de decisión, La visualización de datos, La computación paralela y distribuida, etc.

Aplicaciones

Existe un gran abanico de posibilidades en donde es posible aplicar la minería de datos de manera efectiva, como en: Aplicaciones financieras y banca, Análisis de mercado, Seguros y salud privada, Educación, Procesos Industriales, Medicina, Biología, Telecomunicaciones, etc.
Ejemplos concretos en donde se puede aplicar la minería de datos son:
- Análisis de créditos bancarios. Permitiría predecir qué personas de las que solicitan un crédito no lo devolverían.
- Análisis de la cesta de la compra. Obtendríamos patrones sobre el comportamiento de compra de los clientes, mejorando la ubicación de los productos que se suelen comprar juntos.
- Determinar las ventas de un producto. Lograríamos optimizar el funcionamiento de los almacenes, manteniendo un stock óptimo de cada producto.

Sistemas y herramientas de minería de datos

A finales de la década de los 90's comienzan a aparecer suites, que son capaces de trabajar con distintos formatos, de incorporar distintas técnicas, de obtener distintos tipos de conocimiento y de aplicarse a un gran abanico de áreas.
La aparición de numerosas herramientas y paquetes de "minería de datos" posibilitan el uso de técnicas de minería de datos a no especialistas. Sin embargo para poder sacar mayor provecho a dichas herramientas, es necesaria tener una visión global que nos permita el uso de técnicas diferentes dependiendo de la naturaleza del caso.

Extracción del conocimiento inductivo en las Bases de Datos OLTP y OLAP de Estancia

1. Predecir que infantes presentarán una o varias enfermedades hereditarias
2. Predecir que enfermedades se presentaran de acuerdo a la temporada
4. Predecir nivel de infantes con mayor predisposición presentar desnutrición.
5. Predecir preferencias de alimentos x nivel
6. Asociar el consumo en cada nivel con la estatura de los infantes.
7. Que medicamento funcionó mejor
8. Predecir que enfermedades se presentan de acuerdo a su alimentación

Fase de minería de datos

A. Determinar qué tipo de tarea de minería es el más apropiado. Por ejemplo, podríamos usar la *clasificación* para predecir en un banco los clientes que dejarán de serlo
B. Elegir el tipo de modelo. Por ejemplo, para una tarea de clasificación podríamos usar un *árbol de decisión.* Porque queremos obtener un modelo en forma de reglas.
C. Elegir el algoritmo de minería que resuelva la tarea y obtenga el tipo de modelo que estamos buscando. Esta elección es pertinente porque existen muchos métodos para construir los modelos. Por ejemplo, para crear árboles de decisión para clasificación podríamos usar CART o C5.0 entre otros.

El Problema de la extracción de patrones:
De una manera genérica vamos a ver que tienen en común estás técnicas, a que problema se enfrentan, de qué depende la dificultad de cada método y de qué manera se puede expresar el resultado de estas técnicas.
La perspectiva del proceso ideal de minería de datos se puede resumir en la siguiente figura.

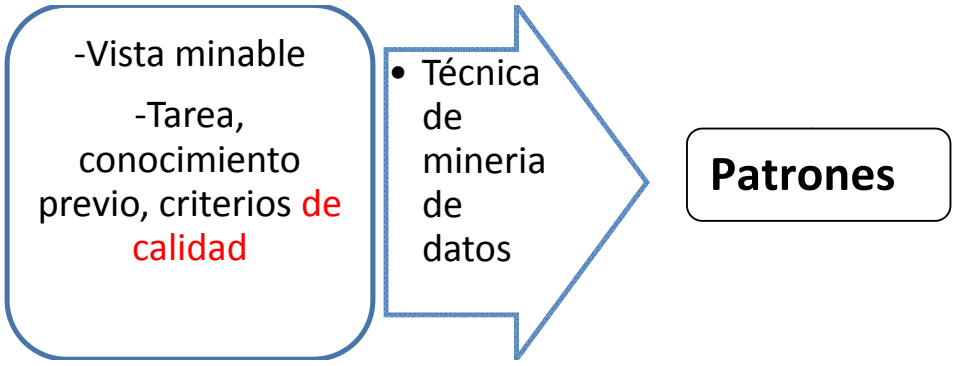


Fig. 4.2 Proceso ideal de la minería de datos

4.1 Tareas y modelos

Una tarea de minería de datos es un (tipo de) problema de minería de datos. Por ejemplo, "clasificar las piezas del proveedor Minatronix en óptimas, defectuosas reparables y defectuosas irreparables" es una tarea de clasificación, que podría resolverse mediante árboles de decisión o redes neuronales, entre otros métodos. Es muy importante distinguir el problema de los métodos para solucionarlo.
Las distintas tareas pueden ser predictivas o descriptivas.

Tareas Predictivas
Hablamos de tareas predictivas cuando se trata de problemas en los que hay que predecir uno o más valores para uno o más ejemplos. Dependiendo de cómo sea la correspondencia entre los *ejemplos* y los *valores de salida* y la *presentación* de los ejemplos podemos definir las siguientes tareas predictivas.

A) Clasificación o discriminación (término utilizado en estadística)
B) Regresión

Tareas Descriptivas:
Los ejemplos se presentan como un conjunto sin etiquetar ni ordenar de ninguna manera, el objetivo, es *describir* los existentes.

A. las tablas de frecuencias,
B. el análisis de componentes principales
C. Agrupamiento (clustering)
D. Sumarización.
E. "Descubrimiento de subgrupos"
F. Correlaciones y factorizaciones
G. Reglas de Asociación
H. Dependencias funcionales
I. Detección de valores e instancias anómalas

Breve descripción de tareas más utilizadas de la minería de datos.
Modelo de Clasificación o discriminación (término utilizado en estadística)(predictiva):
Es quizá la tarea más utilizada. En ella, cada instancia (o registro de la base de datos) pertenece a una clase, la cual se indica mediante el valor de un atributo que llamamos la clase de la instancia. Este atributo puede tomar diferentes valores discretos, cada uno de los cuales corresponde a una clase. El resto de los atributos de la instancia (los relevantes a la clase) se utilizan para predecir la clase.
El objetivo del algoritmo es maximizar *la razón de precisión* de la clasificación de las nuevas instancias, la cual se calcula como el cociente entre las predicciones correctas y el número total de predicciones (correctas e incorrectas).
Los ejemplos se presentan como un conjunto de pares de elementos de dos conjuntos,

$\delta=\{<e,s>:e \in E, s \in S\}$,

donde S es el conjunto de valores de salida.
E es el conjunto de ejemplos,
<e,s> se denominan comúnmente *ejemplos etiquetados* y,
δ se denomina *conjunto de datos etiquetado*.

El objetivo es aprender una función $\lambda:E \rightarrow S$, denominada clasificador, que represente la correspondencia existente en los ejemplos, es decir, para cada valor de E tenemos un único valor para S, además, S es nominal, es decir, puede tomar un conjunto de valores $c_1,c_2,\ldots c_m$ denominados *clases* (cuando el número de clases es dos, tenemos lo que se llama *clasificación_binaria*). La *función* aprendida será capaz de determinar la clase para cada nuevo ejemplo sin etiquetar.
Ejemplo. Un oftalmólogo desea clasificar nuevos pacientes, para decidir si es conveniente operarlos o no en función de una base de datos de sus antiguos pacientes.
Modelo de Regresión (predictiva)
También conocida con otros nombres: *interpolación* (generalmente cuando el valor predicho está en *medio de otros*) o *estimación* (cuando se trata de *algo futuro)y* es quizá la *tarea más sencilla* de definir.
Su objetivo es minimizar el error entre el valor predicho y el valor real.
La diferencia con la clasificación es que S es numérico, es decir puede ser un valor entero o real.
Hablamos de modelo de regresión cuando la variable de respuesta y las variables explicativas son todas ellas *cuantitativas.*
Si sólo disponemos de una variable explicativa hablamos de *regresión simple*, mientras que si disponemos de varias variables explicativas se trata de un problema de *regresión múltiple.*
EJEMPLO DE REGRESION LINEAL
El contador de una compañía constructora quiere predecir el costo de construcción de casas el año próximo, para poder asignar un precio de venta a cada casa.Cree que el costo de la construcción tiene una relación con el tamaño del lote. Se selecciona una muestra aleatoria de 12 casas construidas el año pasado.

Observación	Tamaño de lote (miles de pies cuadrados)	Costo (miles de doláres)
1	5	31.6
2	7	32.4
3	10	41.7
4	10	50.2
5	12	46.2
6	20	58.5
7	22	59.3
8	15	48.4
9	30	63.7
10	40	85.3
11	12	53.4
12	15	54.5

En la siguiente figura se indica una *relación creciente* entre el tamaño X del lote y el costo Y de construcción.La ecuación de regresión de la muestra que representa el modelo de *regresión rectilíneo*, sería: $\hat{Y}_i = b_o + b_1 X_i$

Una vez obtenidos **b_o** *(intercepción* con el eje Y) y **b_1** (la *pendiente)* se conoce la línea recta y se puede trazar en el diagrama de dispersión y se puede ver si los datos originales están cerca de la línea o se desvían.

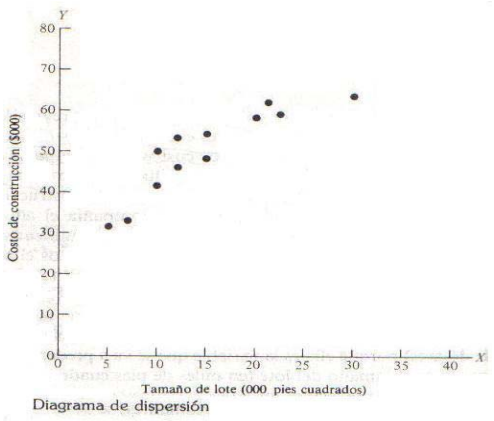


Fig. 4.3 Diagrama de dispersión

Una técnica matemática que determina los valores de b_o y b_1 que mejor ajustan a los datos observados, se conoce como el ***método de los mínimos cuadrados***.

$$b_1 = \frac{n\sum_{i=1}^{n} X_i Y_i - \left(\sum_{i=1}^{n} X_i\right)\left(\sum_{i=1}^{n} Y_i\right)}{\quad}$$

$$n\sum X_i^2 - (\sum X_i)^2$$
$$i=1 \qquad i=1$$

$b_o = \bar{Y} - b_1\bar{X}$

$$\text{Donde } \bar{Y} = \frac{n\sum Y_i}{n} \quad i=1 \quad \text{y} \quad \bar{X} = \frac{\sum X_i}{n} \quad i=1$$

Observación	tamaño de lote X	costo construcción Y	XY	X^2	Y^2
1	5	31,6	158	25	998,56
2	7	32,4	226,8	49	1049,76
3	10	41,7	417	100	1738,89
4	10	50,2	502	100	2520,04
5	12	46,2	554,4	144	2134,44
6	20	58,5	1170	400	3422,25
7	22	59,3	1304,6	484	3516,49
8	15	48,4	726	225	2342,56
9	30	63,7	1911	900	4057,69
10	40	85,3	3412	1600	7276,09
11	12	53,4	640,8	144	2851,56
12	15	54,5	817,5	225	2970,25
	198	625,2	11840,1	4396	34878,58

$$b_1 = \frac{12(11840.1) - (198)(625.2)}{12(4396) - (198)^2}$$
$$= +1.35$$
$$b_o = 52.1 - (1.35)(16.5) = 29.825$$

Por tanto $\hat{Y}_i = 29.825 + 1.35\ X_i$
En este problema del costo de construcción, por cada aumento de 1000 pies cuadrados en el tamaño del lote, el costo de construcción aumentará en $1350 (1.35 miles de dólares)
bo con el eje Y se calculo que era de 29.825 que se puede considerar la parte fija del costo de construcción, representa el valor de Y cuando X=0.
La ecuación de regresión que se ha ajustado a los datos, se puede utilizar ahora para predecir el valor Y para un valor dado de X.

Por ejemplo, predecir el costo promedio de construcción de una casa que se va a construir en un lote de 15000 pies cuadrados.

$\hat{Y}i = 29.825 + 1.35(15) = 29.825 + 20.25 = 50.075$

Las predicciones del costo de construcción sólo se deben hacer para casas con terrenos entre 5000 y 40000 pies cuadrados.

La ecuación de regresión no es perfecta para las predicciones. La línea de regresión sirve sólo para la predicción *aproximada* de un valor Y, para un valor dado de X.

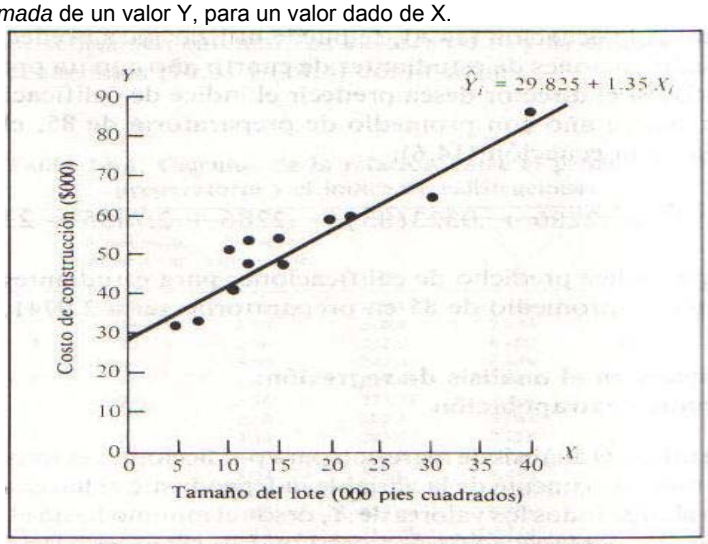


Diagrama de dispersión y recta de regresión

Fig. 4.4 Diagrama de dispersión y recta de regresión

Error estándar de la estimación:

Entonces se necesita, desarrollar un estadístico que mida la variabilidad en torno a la línea de regresión, se llama el error estándar de la estimación.

$$S_{YX} = \sqrt{\frac{\sum_{i=1}^{n} Y_i^2 - b_o \sum_{i=1}^{n} Y_i - b_1 \sum_{i=1}^{n} X_i Y_i}{n-2}}$$

$$Syx = \sqrt{\frac{34878.58 - (29.825)(625.2) - (1.35)(11840.1)}{12-2}}$$

Syx = 4.9785 (miles de dólares) ó $4978.50

Este error estándar de la estimación igual a $4978.50, representa una medida de la variación en torno a la recta ajustada de regresión.

Se mide en unidad de la variable Y dependiente.

4.2 Estadística básica:

Como sabemos el éxito de un proceso de minería de datos depende, no sólo de tener todos los d*atos necesarios* (una buena recopilación), sino de que éstos tengan una buena limpieza e integración.
En la mayoría de BD existe mucha información que es incorrecta. Este problema se acentúa cuando integramos de distintas fuentes.
En esta parte veremos consejos para la *integración*, para la *limpieza* y algunas *transformaciones* para convertir los datos en otros más apropiados para la minería.
Estos procesos reciben nombres bastante variados: preparación de datos, data cooking, pre-procesamiento, etc.
Los objetivos de la preparación de datos: son *eliminar* el mayor número posible de datos erróneos o inconsistentes (limpieza) y datos irrelevantes (criba) y *presentar* los datos de la manera más apropiada para la minería de datos.

Integración

El primer problema a la hora de realizar una integración de distintas fuentes de datos es identificar los objetos, es decir, conseguir que datos sobre el mismo objeto se unifiquen. Este problema se conoce como el problema del esclarecimiento de la identidad.

Reconocimiento

Cuando tenemos integrados todos los datos lo primero que podemos realizar es un resumen de las características de atributos (ya sea tabla para valores nominales y tabla para valores numéricos o tabla a tabla o para toda la base o almacén de datos). En este tipo de tablas se muestran las ***características generales*** de los atributos (medias, mínimos, máximos, posibles valores, etc…).

Por ejemplo, para una compañía de seguros, tenemos los datos de pólizas de vehículos.

Atributo	Tabla	Tipo	# total	# nulos	# dists	Media	Desv.e.	Moda	Min	Max
Código postal	Cliente	Nominal	10320	150	1672	-	-	"46003"	"01001"	"50312"
Sexo	Cliente	Nominal	10320	23	6	-	-	"V"	"E"	"M"
Estado civil	Cliente	Nominal	10320	317	8	-	-	Casado	"Casado"	"Viudo"
Edad	Cliente	Numérico	10320	4	66	42,3	12,5	37	18	87
Total póliza p/a	Póliza	Numérico	17523	1325	142	737,24€	327€	680€	375€	6200€
Asegurados	Póliza	Numérico	17523	0	7	1,31	0,25	1	0	10
Matrícula	Vehículo	Nominal	16324	0	16324	-	-	-	"A-0003-BF"	"Z-9835-AF"
Modelo	Vehículo	Nominal	16324	1321	2429	-	-	"O. Astra"	"Audi A3"	"VW Polo"
...	...	...	...	...	...	...	...	...	...	...

Tabla resumen de atributos.

Fig. 4.5 Tabla resumen de atributos

La tabla anterior es sencilla de construir (se puede hacer incluso a partir de un conjunto de consultas SQL)
Hay aspectos respecto a la calidad de los datos que saltan a la vista; por ejemplo, ¿cómo es posible que haya 5 valores (más el nulo) diferentes para el atributo "sexo"?
Al observar los datos, podemos darnos cuenta de que en realidad el nombre "sexo" para el atributo no está muy bien elegido.

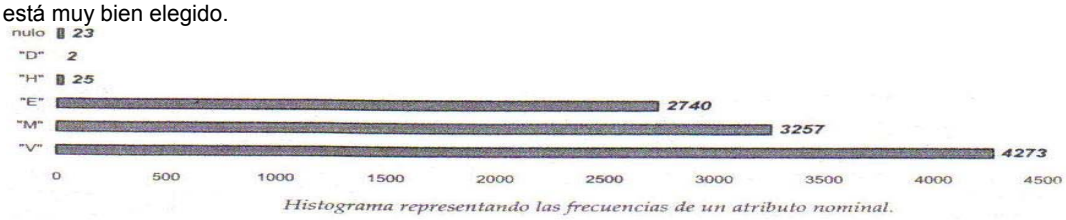


Fig. 4.6 Histograma representado las frecuencias de un atributo nominal

Tras un análisis

- "E" no es una persona si no una empresa que asegura el vehículo.
- La mayoría de" H" se supone que viene de "hombre", pero algunos pueden venir, equivocadamente del término erróneo "hembra".
- "D" puede deberse a que las aplicaciones de la empresa son internacionales y "dona" es "mujer" en catalán, con lo que se decide unificar con "M".

Debido a esta ambigüedad, todos los valores "H" se dejan como valores nulos (luego se verá que muchos de ellos se podrán rellenar, mirando los nombres de los clientes).

Valores faltantes

Los valores faltantes, perdidos o ausentes (missing values) pueden ser reemplazados por varias razones. En primer lugar, la mayoría de los métodos de minería de datos que utilicemos no pueden tratar los

campos faltantes, en segundo lugar si el método es capaz de tratar campos faltantes, es posible que ignore todo el ejemplo o es posible que tenga un método de sustitución de campos faltantes que no sea adecuado.

Las posibles acciones sobre datos faltantes son:

- Ignorar (dejar pasar): algunos *algoritmos* pueden manipular datos faltantes (por ejemplo árboles de decisión)
- Eliminar o reemplazar: A veces la proporción de nulos es tan alta que la columna se elimina (solución extrema). Otras veces, se reemplaza por otra columna dependiente con datos de mayor calidad.
- Filtrar la fila: Se separa la instancia para su posterior revisión.
- Reemplazar el valor: Se puede intentar reemplazar el valor manualmente (en el caso de que no haya muchos) por ejemplo, determinar el sexo a partir del nombre o automáticamente por un valor que preserve la media por clases o grupos.
- Modificar la política de calidad de datos y esperar hasta que los datos faltantes estén disponibles.

Valores erróneos.

La detección de campos erróneos se puede realizar de maneras muy diversas, dependiendo del formato y origen del campo.

Datos erróneos que si se ajusten al formato serán mucho más difíciles (o imposibles) de detectar.

Existen herramientas estadísticas que sugieren los valores anómalos. Pero es el usuario el que finalmente determina si son erróneos o no.

Tratamiento de valores anómalos o erróneos:

- Ignorar (dejar pasar): algunos *algoritmos* pueden manipular datos faltantes (por ejemplo árboles de decisión)
- Eliminar o reemplazar: A veces la proporción de errores es tan alta que la columna se elimina (solución extrema). Otras veces, se reemplaza por otra columna dependiente con datos de mayor calidad (Preferible)
- Filtrar la fila: Se separa la instancia para su posterior revisión.
- Reemplazar el valor por “nulo”.
- Discretizar: transformar un valor en uno discreto (por ejemplo, muy alto, alto, medio, bajo, muy bajo) hace que los valores anómalos caigan en “muy alto” o “muy bajo” sin mayores problemas.

- Transformación de atributos. Creación de características

La transformación de datos engloba

- numerización
- discretización
- reducir la dimensionalidad (número menor de atributos) o
- aumentar la dimensionalidad (manteniendo los mismos atributos o aumentando atributos).

Reducción de dimensionalidad

La alta dimensionalidad es, muchas veces, un gran problema a la hora de aprender de los datos. Si tenemos muchas dimensiones (atributos) respecto a la cantidad de instancias o ejemplos, los patrones no tienen datos donde apoyarse a la hora de tomar una u otra forma. Este problema se conoce popularmente como “la maldición de la dimensionalidad”.

La reducción de dimensionalidad se puede realizar por

- selección de un subconjunto de atributos o
- Transformación de atributos iniciales por otros diferentes (proyección). Para esto puede ayudar la técnica de análisis de componentes principales.
- Análisis de componentes principales

Aumento de la dimensionalidad

La creación, o agregación, de características consiste en crear nuevos atributos que son combinación de otros y, en cierto modo, son atributos redundantes, para mejorar la calidad, visualización o comprensibilidad del conocimiento extraído.

Este proceso recibe muchos nombres: construcción, creación o descubrimiento de características.

Atributos numéricos: se utilizan, generalmente, operaciones matemáticas básicas de uno o más argumentos: suma, resta, producto, división, máximo, mínimo, media, cuadrado, raíz cuadrada, senos, cosenos, etc. Todas ellas retornan un valor numérico. También se pueden generar valores nominales a partir de valores numéricos, por ejemplo, crear un atributo que indique si un determinado valor numérico es mayor, menor o igual que un determinado valor o que otro atributo, etc.

Atributos nominales: se utilizan, generalmente, operaciones lógicas: conjunción, disyunción, negación, implicación, igualdad o desigualdad. Todas ellas retornan un valor nominal. También se pueden generar valores numéricos a partir de valores nominales.

Discretrización

La discretización, o cuantización (también llamada "binning") es la conversión de un valor numérico en un valor nominal ordenado (que representa un intervalo o "bin"). Por ejemplo, convertir una calificación de 0 a 10 en una serie de valores ordenados

reprobado 0-4.99,
aprobado 5-6.99,
notable 7-8.49,
sobresaliente 8.5-9.99
matrícula de honor (10).

Así se integran escalas diferentes en un tipo de escala común.

Una última razón y, quizá la más obvia, es cuando tenemos algunos atributos nominales y otros numéricos, y queremos que todos sean nominales para, por ejemplo, establecer reglas de asociación.

Numerización

La numerización es menos común que la discretización, es útil cuando el método de minería de datos que vamos a utilizar no admite datos nominales.

Por ejemplo, supongamos que tenemos el nivel de estudios de un conjunto de individuos, con valores posibles (sin estudios, primarios, secundarios, bachillerato, universitario, doctor). Nos puede interesar convertirlo en una escala de (0,1,2,3,4,5).

4.3 Métodos.

Árboles de decisión

Un árbol de decisión es un conjunto de condiciones organizadas en una estructura jerárquica, de tal manera que la decisión final a tomar se puede determinar siguiendo las condiciones que se cumplen desde la raíz del árbol hasta alguna de sus hojas.
Una de las grandes ventajas de los árboles de decisión es que, las opciones posibles a partir de una determinada condición son excluyentes. Esto permite llegar a una sola acción o decisión a tomar.

Árboles de decisión para clasificación.
La tarea de aprendizaje para la cual los árboles de decisión se adecuan mejor es la clasificación.
Otra característica importante de los algoritmos de aprendizaje de árboles de decisión es que una vez elegida la partición ya no se puede cambiar.
Por tanto, uno de los aspectos más importantes en los sistemas de aprendizaje de árboles de decisión es el denominado criterio de partición, ya que una mala elección de la partición generará un peor árbol.
El algoritmo va construyendo el árbol añadiendo particiones a los hijos resultantes de cada partición, los ejemplos se van dividiendo entre los hijos. Finalmente, los nodos inferiores son de la misma clase y esa rama ya no sigue creciendo.
La cardinalidad de los nodos irá disminuyendo a medida que se desciende en el árbol.

Los dos puntos más importantes para que el algoritmo funcione bien son las "*Particiones a considerar"* y el *"Criterio de selección de particiones".*

Particiones a considerar
Las particiones son un conjunto de condiciones exhaustivas y excluyentes. Lógicamente, cuantos más tipos de condiciones permitamos, más posibilidades tendremos de encontrar los patrones que hay detrás de los datos.
Cuantas más particiones permitamos más expresivos podrán ser los árboles de decisión generados y, probablemente, más precisos. No obstante, cuantas más particiones elijamos, la complejidad del algoritmo será mayor. Por tanto, un buen algoritmo es encontrar un buen compromiso entre expresividad y eficiencia. Existen dos tipos de particiones:

Particiones nominales: Una condición con la igualdad entre el atributo y un posible valor.
Particiones numéricas: Si un atributo x_i es numérico y continuo, puede haber tomado muchos valores diferentes en los ejemplos y puede tomar infinitos posibles valores en general. Por esta razón, se intentan obtener particiones que separen los ejemplos en intervalos.

Criterio de selección de particiones
Si existen n atributos y m valores posibles para cada atributo, el número de particiones posibles es de $n \cdot m$), una vez elegida la partición se continua hacia abajo la construcción del árbol y no vuelven a plantearse las particiones ya construidas.

Ejemplo 1: Árboles de decisión

Justificación

Las causas principales de la desnutrición son:

- Disminución de la ingesta dietética.
- Mal absorción.
- Aumento de los requerimientos, como ocurre por ejemplo en los lactantes prematuros, en infecciones, traumatismo importante o cirugía.
- Psicológica; por ejemplo, depresión o anorexia nerviosa.

La desnutrición se puede presentar debido a la carencia de una sola vitamina en la dieta o debido a que la persona no está recibiendo suficiente alimento. La inanición es una forma de desnutrición. La desnutrición también puede ocurrir cuando se consumen los nutrientes adecuadamente en la dieta, pero uno o más de estos nutrientes no es/son digerido(s) o absorbido(s) apropiadamente.
La desnutrición puede ser lo suficientemente leve como para no presentar síntomas o tan grave que el daño ocasionado sea irreversible, a pesar de que se pueda mantener a la persona con vida.
A nivel mundial, especialmente entre los niños que no pueden defenderse por sí solos, la desnutrición continúa siendo un problema significativo

Nuestra estancia

Ya que se trata de un problema en México y el mundo, en la estancia se ha decidido tratar de ***Predecir el nivel académico de los infantes con mayor predisposición por presentar desnutrición.***

El *estado nutricional* en condiciones normales es la resultante del balance entre lo consumido y lo requerido, lo cual está determinado por la *calidad* y *cantidad* de nutrientes de la dieta y por su utilización completa en el organismo, por lo que aquí nos inclinaremos a la información que tenemos disponible en nuestro OLAP; en particular en la *cantidad* de alimentos consumidos y la edad o *nivel* del infante .

Para esto necesitamos una vista minable que contenga el *curp de infante*, su *consumo* y el *nivel*, pero para saber cómo determinar quiénes son propensos y quienes no, vamos a agregar un atributo que determinará esto según el consumo de cada infante, es decir los niños que presenten mayor incidencia de (No Comió o Probó) en alimentos con mayor contenido nutricional como (alimentos_fuertes = tipo_alimento:3) serán propensos a presentar dicha enfermedad.

El Query que utilizamos para obtener la vista minable desde nuestro datamart llamado Dieta es:

```
SELECT      distinct D.curp_infante, D.consumo,
            N.nombre, D.nivel
FROM        dieta D, menor M, nivel N, alimento A, des_com DC, alimento_original AO,
            tipos_alimentos T
WHERE       D.año=2008
AND         D.nivel             = M.nivel
 AND        M.nivel             =N.nivel
AND         D.clave             =A.clave
 AND        A.tipo_alimento     =AO.tipo_alimento
AND         AO.tipo_alimento    = '3'
 AND        A.des_com           = DC.des_com
AND         DC.des_com          ='c',
            CASE WHEN consumo ='CM' THEN 'no' WHEN consumo ='CT' THEN 'no'
            ELSE  ' yes' END desnutricion;
```

Tomando una muestra de la vista minable, tenemos que luce así:

curp_infante	consumo	nombre	nivel	desnutricion
OACF040807HJCLSL05	CM	PREESCOLAR	7	no
MAMG040823HDFNDN02	NC	PREESCOLAR	7	yes
ZACM050826HJCMRG02	PR	PREESCOLAR	6	yes
BACG060527HDFRRN06	NC	MATERNAL	5	yes
EUCG040904MJCSHR04	CM	PREESCOLAR	6	no
GACV050803MJCRSR09	CM	PREESCOLAR	6	no
TADA040223HJCLXR05	CT	PREESCOLAR	7	no
AUET060223MDFGNN04	NC	MATERNAL	5	yes
MEGJ040506HDFRZS06	CT	PREESCOLAR	7	no
MUGM041019HDFXNG09	CM	PREESCOLAR	6	no
OICP060319MJCRLM09	CT	MATERNAL	5	no
SUGJ050826HDFBNR05	CM	PREESCOLAR	6	no
YABC060605MJCXRR08	CT	MATERNAL	5	no
VIRI050913MDFLDV09	CT	MATERNAL	5	no

…

Donde *desnutrición* es el parámetro que utilizamos para determinar si un infante es o no propenso a presentar desnutrición.

Tabla característica de atributos:

Atributo	Tabla	Tipo	Total	Nulos	Dist	Media	Desv.E.	Moda	Min	Max
curp_infante	Menor	nominal	693	0	165	*	*	*	*	*
consumo	Consumo	nominal	693	0	4	*	*	*	NC	CM
nombre	nivel	nominal	693	0	3	*	*	PREESCOLAR	MATERNAL	PREESCOLAR
nivel	nivel	numérico	693	0	8	6	1.15657563	5	1	8
desnutricion	Fte.Externa	nominal	693	0	2	*	*	no	no	yes

Donde:

Consumo	NC- no comio, PR -Probo, CM- Comio mucho, CT –Comio todo
Nombre	LACTANTE,MATERNAL,PREESCOLAR
Nivel	1-Lactante A, 2 –Lactante B, 3 –Lactante C, 4 – Maternal A, 5 - Maternal B, 6 – Preescolar A, 7 – Preescolar B, 8 – Preescolar C
desnutrición	NO,YES

Siendo así vamos a utilizar el modelo de clasificación con una técnica basada en *Árboles de decisión* pues como sabemos se trata de un algoritmo que nos permite clasificar o tomar decisiones a través de un conjunto de condiciones estructuradas jerárquicamente.

Procedimiento

Utilizando Weka, importamos el archivo ***desnutrición.csv*** que es el archivo que contiene nuestra vista minable.

Una vez procesado nuestro archivo podemos ver la estadística básica, que se cálculo:

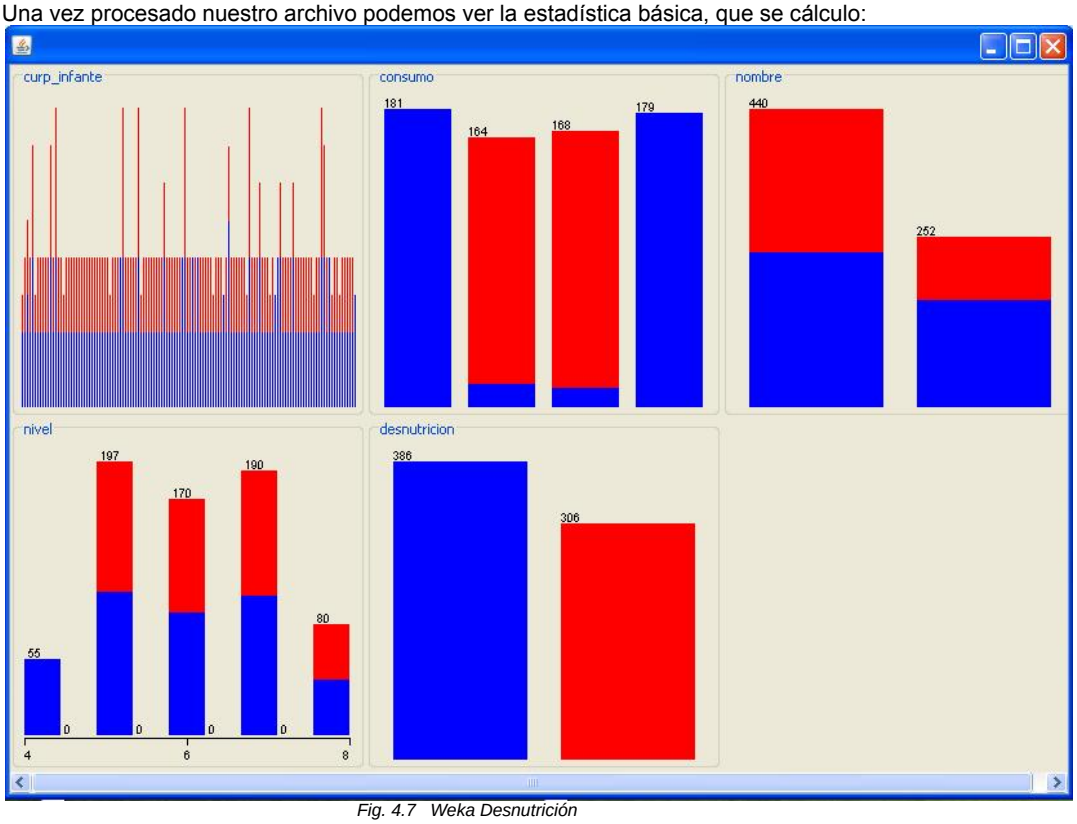


Fig. 4.7 Weka Desnutrición

Para poder apreciar mejor el comportamiento de las variables utilizadas vamos a proceder a recurrir a algún algoritmo que nos ayude a interpretar en este caso el nivel que estamos buscando.

Ahora para procesar nuestra vista con ayuda del algoritmo de clasificación: *Árboles de decisión* seleccionamos la pestaña *"Classify",* damos clic en el botón *"Choose"* , y seleccionamos dentro del apartado para árboles, el algoritmo "***J48"***,y ahora lo ejecutamos, seleccionando en el apartado de ***"Test options"*** la variable que queremos predecir, en este caso **desnutrición.**

Esta es la salida que nos muestra una vez ejecutado el algoritmo:

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     desnutricion
Instances:    692
Attributes:   5
              curp_infante
              consumo
              nombre
              nivel
              desnutricion
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

consumo = CM: no (181.0/3.0)
consumo = NC
|   nivel <= 4: no (14.0)
|   nivel > 4: yes (150.0/1.0)
consumo = PR
|   nivel <= 4: no (12.0)
|   nivel > 4: yes (156.0/4.0)
consumo = CT: no (179.0/2.0)

Number of Leaves  :         6

Size of the tree :  9


Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         682               98.5549 %
Incorrectly Classified Instances        10                1.4451 %
Kappa statistic                          0.9707
Mean absolute error                      0.0285
Root mean squared error                  0.1198
Relative absolute error                  5.7702 %
Root relative squared error             24.1293 %
Total Number of Instances              692

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.987    0.016      0.987     0.987     0.987      0.98     no
                 0.984    0.013      0.984     0.984     0.984      0.98     yes
Weighted Avg.    0.986     0.015      0.986     0.986     0.986      0.98

=== Confusion Matrix ===

   a   b   <-- classified as
 381   5 |   a = no
   5 301 |   b = yes
```

Como podemos observar el algoritmo realizó una clasificación en base a los datos más representativos, tomando en cuenta las condiciones necesarias, dividiendo así los datos en varias instancias.
Analizando los resultados arrojados por el algoritmo tenemos que:

El número de instancias clasificadas correctamente fue de

Correctly Classified Instances	682	98.5549 %
Incorrectly Classified Instances	10	1.4451 %

La matriz de confusión:

```
  a   b   <-- classified as
381   5 |  a = no
  5 301 |  b = yes
```

Y principalmente la lógica de construcción del árbol, que para apreciar gráficamente, hacemos clic derecho sobre el algoritmo y seleccionamos *vizualize tree*:

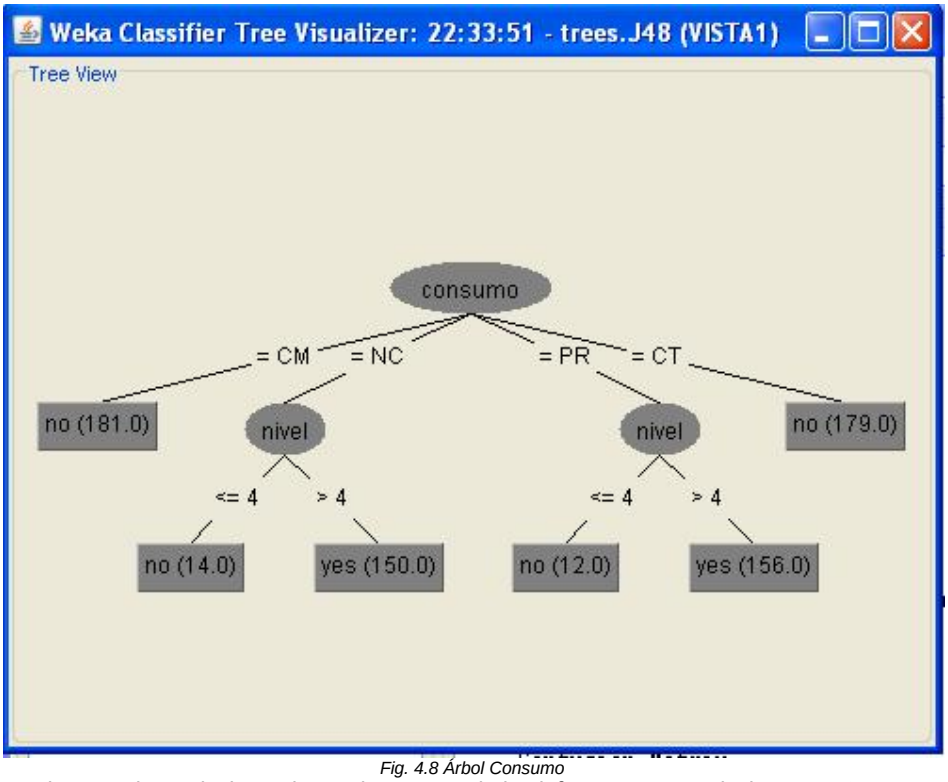


Fig. 4.8 Árbol Consumo

Con lo cual se puede predecir con base al consumo de los infantes, en que niveles encontraremos mayor propensión a la desnutrición, en este caso fueron 5, 6, 7, 8, que se tratan del nivel maternal y preescolar principalmente.

Conclusión

Dados los resultados obtenidos, podríamos proponer a la estancia organizar una serie de campañas en las que se fomente el desarrollo nutricional de los infantes, teniendo mayor énfasis en dichos niveles y de esta manera disminuir el porcentaje de desnutrición en cada estancia infantil.
Para esto podríamos analizar la preferencia de alimentos que se tienen en cada nivel, de esta manera determinaríamos una solución que promueva una buena alimentación.
Así ayudaríamos a evitar que la desnutrición sea un factor que afecta el desarrollo de salud de los niños.

Ejemplo 2: Árboles de decisión

Justificación
Generalmente, es durante la infancia cuando se desarrollan los hábitos nutricionales, y cuando el aprendizaje se realiza en gran medida por imitación de los adultos. Los padres o tutores deben tener presente que el apetito y los gustos del niño varían con el tiempo.
Además, muchos niños llegan a descartar una clase completa de alimentos, por ejemplo las verduras, por lo cual puede haber riesgo de carencia de nutrientes esenciales. Es importante entonces variar las preparaciones, hacerlas atractivas y disimular los alimentos que no son del agrado del niño, dentro de otras comidas que sí lo entusiasmen, hasta que con el tiempo desaparezcan las resistencias. Habitualmente, los niños tienen gusto por las frutas, y una buena ingesta de éstas puede sustituir temporariamente el rechazo por otros vegetales y proveer los minerales y vitaminas necesarios.

Nuestra estancia
Como vimos en el caso anterior, para poder desarrollar campañas o programas que fomenten el desarrollo nutricional se necesita tener una base que nos pueda orientar en cuanto a la preferencia de alimentos que se tiene de acuerdo al nivel de los infantes.
Debido a esto vamos a intentar ***predecir las preferencias de alimentos de los infantes por nivel***, con base nuevamente a la cantidad de consumo (CT comió Todo) y frecuencia de alimentos que consumen los infantes, con la finalidad de estimular hábitos nutricionales que proporcionen los nutrientes esenciales que necesitan los infantes para su desarrollo.

Para tal efecto necesitaremos incluir en nuestra vista los siguientes atributos;
Tipo_alimento, nombre_alimento, consumo y nivel
Como podemos observar estos datos pueden ser obtenidos de nuestro OLAP "Dieta", utilizando el siguiente Query, para formar nuestra vista minable:

```
SELECT distinct T.descripcion as tipo_alimento ,AO.nombre as nombre_alimento,
        D.consumo, N.nombre as nombre_nivel, D.nivel
FROM   dieta D, menor M, nivel N alimento A, alimento_original AO, tipos_alimentos T
WHERE D.año                 =2008
AND     D.consumo           ='CT'
AND     D.curp_infante      =M.curp_infante
AND     D.nivel             =M.nivel
AND     M.nivel             =N.nivel
AND     D.clave             =A.clave
AND     A.tipo_alimento     =AO.tipo_alimento
AND     AO.tipo_alimento    = T.tipo_alimento
GROUP BY D.nivel, D.consumo, T.descripcion, AO.nombre;
```

Tomando una muestra de la vista minable, tenemos que luce así:

descripcion	nombre	consumo	nivel_nombre	nivel
variaciones_leche	Chocolate	CT	LACTANTE	2
variaciones_leche	Atole	CT	LACTANTE	2
variaciones_leche	Avena	CT	LACTANTE	1
variaciones_leche	Cajeta	CT	LACTANTE	3
variaciones_leche	Canela	CT	LACTANTE	2
variaciones_leche	Caramelo	CT	LACTANTE	3
variaciones_leche	arroz	CT	LACTANTE	3
variaciones_leche	fresa	CT	LACTANTE	3
variaciones_leche	Natural o Formula	CT	LACTANTE	1

En donde se puede apreciar el tipo de alimento y el nombre del alimento, que serán los parámetros que nos interesan predecir.

Tabla característica de atributos:

Atributo	Tabla	Tipo	Total	Nulos	Dist	Media	Desv.E.	Moda	Min	Max
descripción	Tipos_alimentos	nominal	603	0	7	*	*	alimentos_fuertes	*	*
consumo	Consumo	nominal	603	0	4	*	*	*	*	*
nombre	Nivel	numérico	603	0	3	*	*	PREESCOLAR	LACTANTES	PREESCOLAR
nivel	Menor	nominal	603	0	8	5	1.9803	3	1	8

Donde:

Descripción Tipos_alimento	Molletes, Filete, Chilaquiles etc..
Consumo	NC- no comió, PR -Probo, CM- Comió mucho, CT –Comió todo
Nombre	LACTANTE,MATERNAL,PREESCOLAR
Nivel	1-Lactante A, 2 –Lactante B, 3 –Lactante C, 4 – Maternal A, 5 - Maternal B, 6 – Preescolar A, 7 – Preescolar B, 8 – Preescolar C

Al igual que el ejemplo anterior utilizaremos el modelo de clasificación con una técnica basada en *Árboles de decisión*.

Procedimiento

Utilizando Weka, importamos el archivo ***preferencia.cvs***

Ahora decidimos procesar nuestra vista con ayuda nuevamente del algoritmo de clasificación *Árboles de decisión* pues necesitamos dividir en instancias tales que nos muestre la preferencia de alimentos de los infantes por nivel así que es el más adecuado para mostrarnos dicha relación.
Seleccionamos la pestaña *"Classify",* damos clic en el botón *"Choose"* , y seleccionamos dentro del apartado para árboles, el algoritmo "***J48"***,y ahora lo ejecutamos, seleccionando en el apartado de ***"Test options"*** la variable que queremos predecir, en este caso **nombre** (que se refiere al nombre del alimento)**,** y obtenemos los siguientes resultados:

```
=== Run information ===
Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     preferencia
Instances:    602
Attributes:   5
              descripcion
              nombre
              consumo
              nivel_nombre
              nivel
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===
J48 pruned tree
------------------
descripcion = alimentos_fuertes
|   nivel_nombre = LACTANTE
|   |   nivel <= 2: Ensalada de jamon (3.0/2.0)
|   |   nivel > 2
|   |   |   nivel <= 3: Alambre de Pollo (25.0/12.0)
|   |   |   nivel > 3: Hotcakes (2.0/1.0)
|   nivel_nombre = MATERNAL
|   |   nivel <= 4: Alambre de Res (27.0/14.0)
|   |   nivel > 4: Hamburguesa (33.0/16.0)
|   nivel_nombre = PREESCOLAR
|   |   nivel <= 6: Huevo (30.0/21.0)
|   |   nivel > 6: Hamburguesa (60.0/23.0)
descripcion = cereales_leguminosas
|   nivel_nombre = LACTANTE: Avena (18.0)
|   nivel_nombre = MATERNAL: Cornflakes (18.0)
|   nivel_nombre = PREESCOLAR
|   |   nivel <= 6: Avena (8.0)
|   |   nivel > 6
|   |   |   nivel <= 7: Frijoles (9.0/5.0)
|   |   |   nivel > 7: Arroz (9.0)
descripcion = fruta_yogourth_postres
|   nivel_nombre = LACTANTE
|   |   nivel <= 2: Manzana al horno (6.0/5.0)
|   |   nivel > 2: Coctel (26.0/1.0)
|   nivel_nombre = MATERNAL
|   |   nivel <= 4: Yogurt (15.0/5.0)
```

```
|   |   nivel > 4: Platano (17.0/9.0)
|   nivel_nombre = PREESCOLAR
|   |   nivel <= 6: Yogurt (16.0/3.0)
|   |   nivel > 6: Galletas (32.0/10.0)
descripcion = jugos_aguas
|   nivel_nombre = LACTANTE
|   |   nivel <= 2: Papaya_A (15.0/8.0)
|   |   nivel > 2: Naranja_J (15.0/7.0)
|   nivel_nombre = MATERNAL
|   |   nivel <= 4: Naranja_J (15.0/2.0)
|   |   nivel > 4: Horchata_A (15.0/1.0)
|   nivel_nombre = PREESCOLAR: Limon_A (45.0/9.0)
descripcion = sopas
|   nivel <= 7
|   |   nivel <= 2: Calabaza (5.0/4.0)
|   |   nivel > 2: Pasta (31.0/3.0)
|   nivel > 7: Calabaza (6.0)
descripcion = variaciones_leche
|   nivel_nombre = LACTANTE
|   |   nivel <= 2: Natural o Formula (14.0/2.0)
|   |   nivel > 2: Chocolate (13.0/7.0)
|   nivel_nombre = MATERNAL
|   |   nivel <= 3
|   |   |   nivel <= 1: Natural o Formula (3.0)
|   |   |   nivel > 1: Chocolate (2.0)
|   |   nivel > 3
|   |   |   nivel <= 4: Avena (9.0/2.0)
|   |   |   nivel > 4: Cajeta (4.0/1.0)
|   nivel_nombre = PREESCOLAR
|   |   nivel <= 6: Cajeta (9.0/5.0)
|   |   nivel > 6: Chocolate (18.0/2.0)
descripcion = verduras_ensaladas
|   nivel <= 6
|   |   nivel <= 4: Brocoli (6.0/4.0)
|   |   nivel > 4: Nopales (8.0/2.0)
|   nivel > 6: Pepino (14.0/3.0)

Number of Leaves  :     37

Size of the tree :      63

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         492               82.3216%
Incorrectly Classified Instances       109               17.6784 %
Kappa statistic                          0.6276
Mean absolute error                      0.0129
Root mean squared error                  0.0831
Relative absolute error                 46.1434 %
Root relative squared error             70.3248 %
Total Number of Instances              601
Ignored Class Unknown Instances                 1


=== Detalle en la clase de precisión===
               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0         0.003     0         0         0         0.454    Ensalada de jamon
                 0         0.002     0         0         0         0.498    Pan Frances
                 0         0         0         0         0         0.429    Pollo
                 0.5       0.022     0.381     0.5       0.432     0.932    Alambre de Pollo
                 0.042     0.009     0.167     0.042     0.067     0.939    Hotcakes
                 0.867     0.024     0.481     0.867     0.619     0.971    Alambre de Res
                 0.9       0.036     0.3       0.9       0.45      0.925    Huevo
                 0         0         0         0         0         0.981    Molletes
                 0         0         0         0         0         0.619    Tortitas de papa
                 0.931     0.072     0.581     0.931     0.715     0.948    Hamburguesa
                 0         0         0         0         0         0.475    Higado empanizado
                 0         0         0         0         0         0.823    Pollo Empanizado
                 0         0         0         0         0         0.923    Salchichas a la mexicana
                 0         0         0         0         0         0.897    Albondiga en Salsa
                 0         0         0         0         0         0.478    Arroz blanco
```

0	0	0	0	0	0.976	Bistec
0	0	0	0	0	0.434	Ensalada de pepino
0	0	0	0	0	0.478	Salpicon
0	0	0	0	0	0.432	Sopa de verduras
0.943	0.004	0.943	0.943	0.943	0.969	Avena
1	0	1	1	1	1	Cornflakes
1	0.008	0.444	1	0.615	0.993	Frijoles
0	0	0	0	0	0.495	Garbanzo
0.692	0	1	0.692	0.818	0.916	Arroz
0	0.008	0	0	0	0.496	Manzana al horno
0	0.002	0	0	0	0.496	Pina
0.143	0.012	0.222	0.143	0.174	0.945	Platano
0	0	0	0	0	0.484	Sandia
0	0	0	0	0	0.485	Uvas
0.885	0.014	0.742	0.885	0.807	0.93	Yogurt
0.962	0.002	0.962	0.962	0.962	0.978	Coctel
0.25	0.01	0.25	0.25	0.25	0.988	Melon
0	0	0	0	0	0.488	Tuna
0	0	0	0	0	0.477	Durazno
1	0.017	0.688	1	0.815	0.991	Galletas
0	0	0	0	0	0.973	Naranja
0	0	0	0	0	0.445	Guayaba_A
0.824	0.002	0.933	0.824	0.875	0.903	Horchata_A
0.947	0.016	0.8	0.947	0.867	0.961	Limon_A
0	0	0	0	0	0.884	Mandarina_A
0.7	0.014	0.467	0.7	0.56	0.983	Papaya_A
0	0	0	0	0	0.489	Mandarina_J
0	0	0	0	0	0.49	MelÑn_A
1	0.016	0.7	1	0.824	0.993	Naranja_J
0	0	0	0	0	0.489	Tuna_A
0	0	0	0	0	0.456	Uva_J
0	0	0	0	0	0.468	Jamaica_A
0	0	0	0	0	0.949	Pina_A
0.857	0.005	0.667	0.857	0.75	0.928	Calabaza
0	0.003	0	0	0	0.497	Papa
0	0	0	0	0	0.474	Pasta con espinacas
0	0	0	0	0	0.475	Verduras
0	0	0	0	0	0.474	Zanahoria
1	0.005	0.903	1	0.949	0.996	Pasta
1	0.007	0.789	1	0.882	0.998	Natural o Formula
0.607	0.021	0.586	0.607	0.596	0.971	Chocolate
0	0.008	0	0	0	0.981	fresa
0	0	0	0	0	0.496	Atole
0.571	0.01	0.4	0.571	0.471	0.99	Cajeta
0	0	0	0	0	0.488	Canela
0	0	0	0	0	0.489	Caramelo
0.786	0.009	0.688	0.786	0.733	0.957	Pepino
0	0	0	0	0	0.495	Ejotes
0	0	0	0	0	0.495	Lechuga
0	0	0	0	0	0.495	Papas
0	0.008	0	0	0	0.992	Brocoli
0.833	0.003	0.714	0.833	0.769	0.914	Nopales
0	0	0	0	0	0.985	Zanahorias
Weighted Avg.	0.644	0.015	0.541	0.644	0.577	0.919

Analizando los resultados arrojados por el algoritmo tenemos que.
El número de instancias clasificadas correctamente fue de

Correctly Classified Instances	492	82.3216%
Incorrectly Classified Instances	109	17.6784 %

Y la matriz de confusión resultó realmente grande debido al número de combinaciones que se tuvieron, por lo que será más conveniente observar directamente la lógica de construcción del árbol, así que hacemos clic derecho sobre el algoritmo y seleccionamos *vizualize tree*:

Debido a que el árbol fue grande mostraremos cada una de sus ramas por separado, siendo el nodo principal; el atributo **descripción** (que se refiere al nombre del tipo de alimento).

Alimentos fuertes:

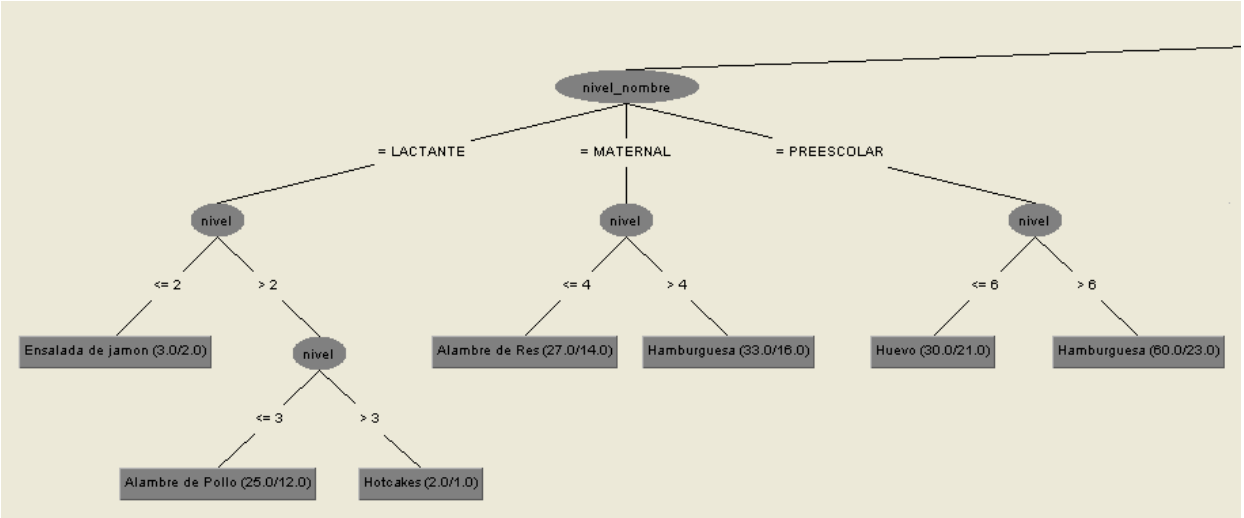


Fig. 4.9 Árbol de alimentos fuertes

Cereales y leguminosas:

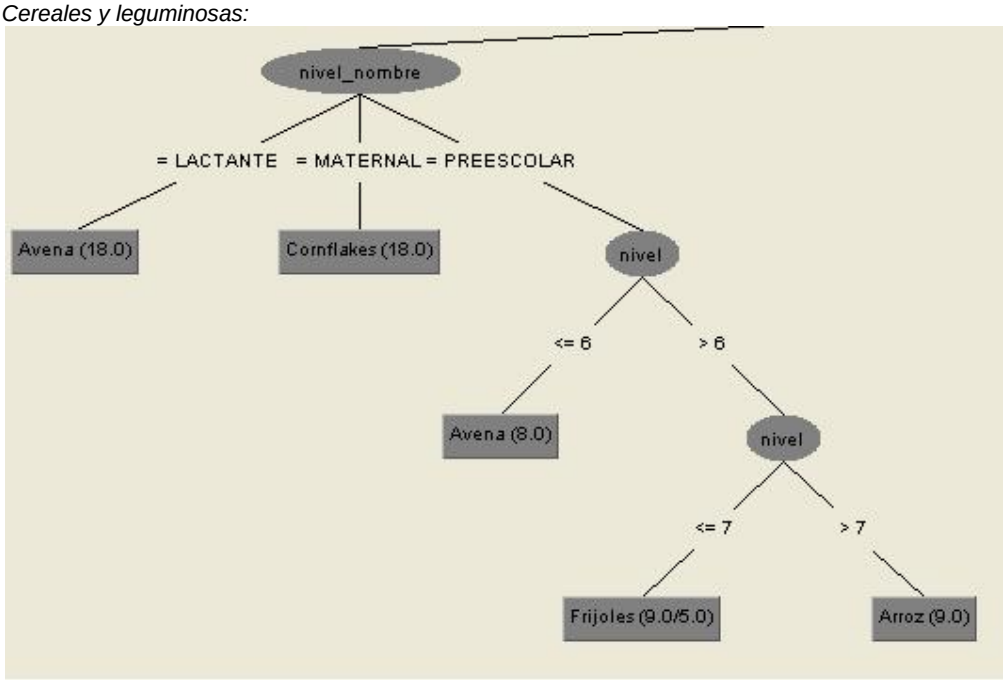


Fig. 4.10 Árbol de Cereales y leguminosas

Frutas Yogurt y postre:

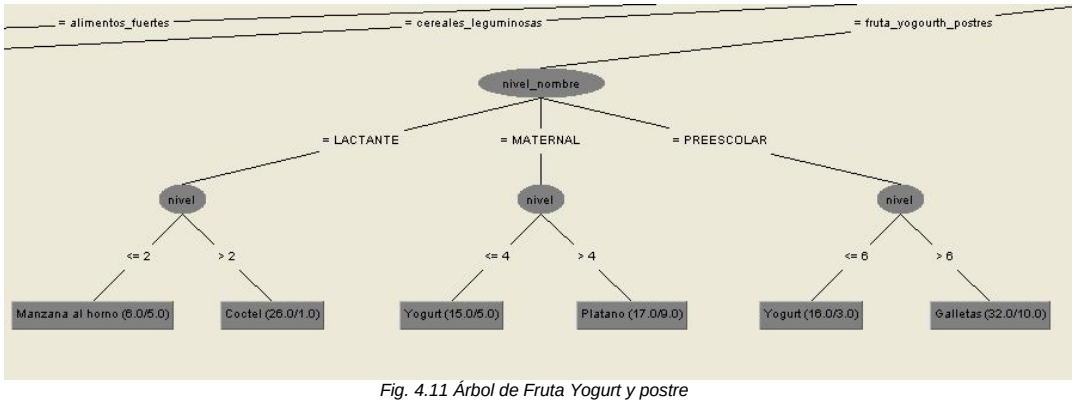


Fig. 4.11 Árbol de Fruta Yogurt y postre

Jugos, Aguas y Sopas:

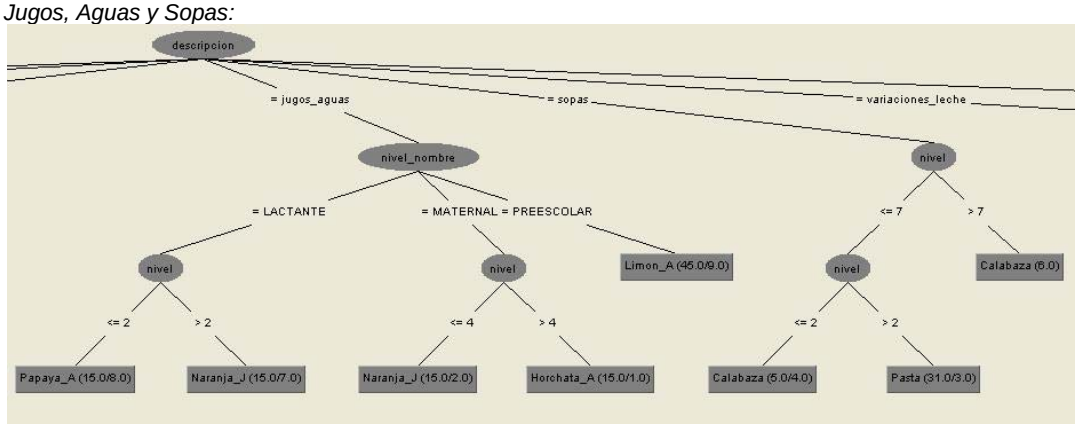


Fig. 4. 12 Jugos, Aguas y Sopas

Variaciones de leche:

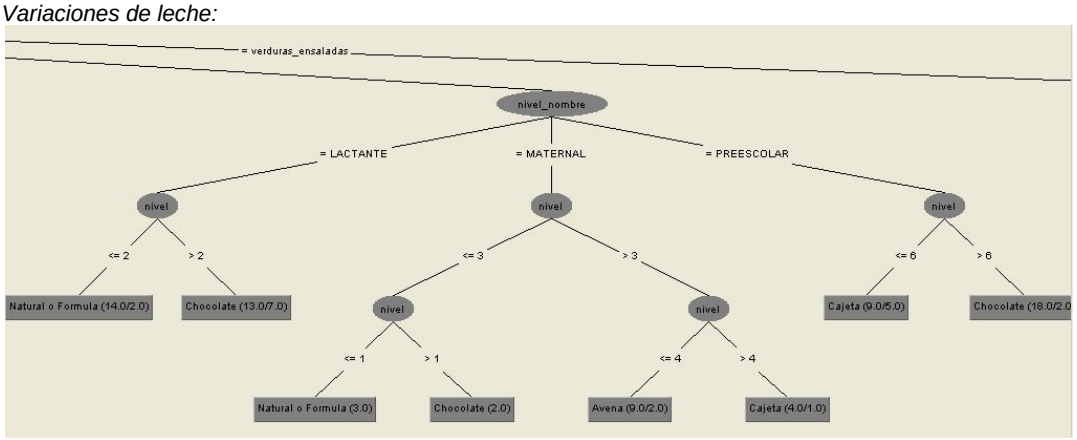


Fig. 4. 13 Variaciones de leche

Verduras y ensaladas:

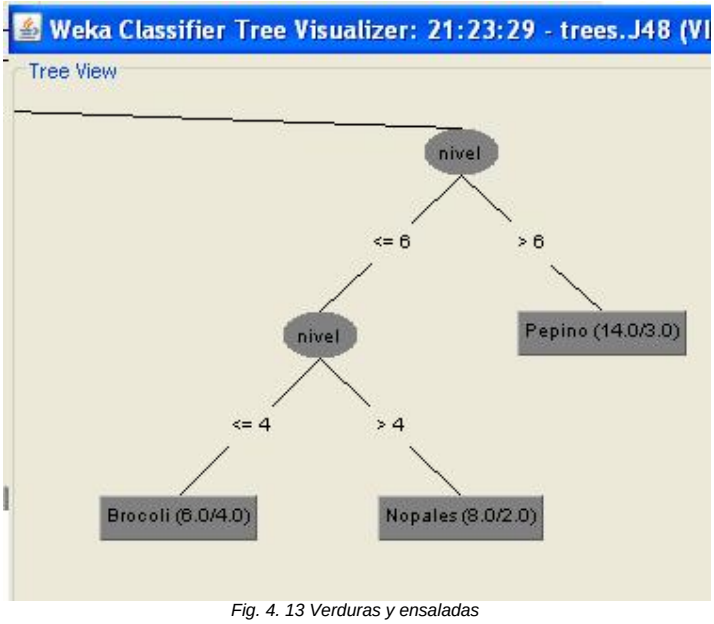


Fig. 4. 13 Verduras y ensaladas

Conclusión

Como podemos notar la preferencia de alimentos varía de acuerdo al nivel, aunque el gusto por ciertos alimentos prevalece en varios niveles.
Con esta información podemos crear programas que incluyan menús que proporcionen a los infantes, mayor variedad de alimentos que contengan los nutrientes necesarios para el desarrollo de los infantes, conociendo además que alimentos les agradan y cuáles no para y así poder ofrecer infinidad de alimentos para que aprenda a degustarlos mejor sin necesidad de que se le obligue a comer, ya que a esta edad el niño participa en su alimentación y es libre de escoger y decidir la cantidad y tipo de alimentos que consume; aún así quien se los proporciona es responsable de ella.
Si la falta de apetito es frecuente , es necesario verificar que las comidas intermedias no interfieran con las principales.
La Educación nutricional debe ser parte de los programas académicos de los escolares, de los deportistas, pero debe continuarse y reforzarse en el grupo familiar.

Ejemplo 3: Árboles de decisión

Justificación

En México, de acuerdo a la Encuesta Nacional de Nutrición realizada en 1999, 27.5% de los niños en edad escolar presentan sobrepeso.

El Servicio de Endocrinología Pediátrica del Hospital Infantil de México "Federico Gómez" declaró que el 40% de la población infantil en nuestro país sufre sobrepeso y obesidad. Esto se debe en gran parte al cambio de vida que ha tenido la sociedad tanto en sus hábitos alimenticios, como en la actividad física que realizan.

Los niños con obesidad pueden sufrir de hipertensión, colesterol elevado y resistencia a la insulina desde la infancia o pubertad y continúan con el riesgo en la etapa adulta.

Las razones fundamentales por la cual un niño es obeso incluyen: Sobrealimentación, Sedentarismo, Factores hereditarios, Situación hormonal, Situaciones psicosociales y ambientales.

El sobrepeso y la obesidad infantil está detonando la aparición de diversas enfermedades que anteriormente sólo se veían en la población adulta, además de que la obesidad es una enfermedad crónica, progresiva e incurable de forma espontánea. Por lo tanto, es sumamente valiosa la intervención de padres y personal médico.

Nuestra estancia

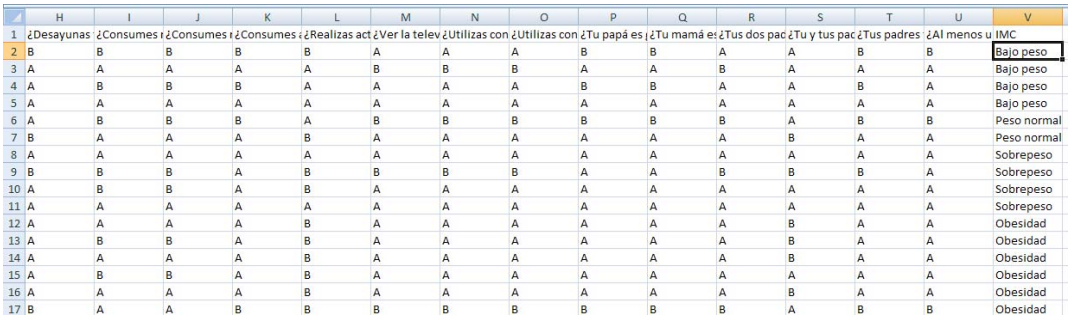
Tomando en cuenta este grave problema, se ha decidido en la estancia ayudar a los doctores a detectar rápidamente factores que inciden en la aparición del sobrepeso en los infantes.
La estancia ha estado aplicando periódicamente encuestas a los infantes, con preguntas orientadas a detectar los factores que los médicos, a través de su experiencia, han logrado detectar como los principales causales del sobrepeso. Las preguntas que se han aplicado son:

1. ¿Consumes regularmente refrescos?
2. ¿Consumes 1 o más veces a la semana botanas y/o golosinas?
3. ¿Regularmente realizas tus comidas en tu casa?
4. ¿Consumes 3 o más veces a la semana alimentos que no sean caseros (industrializados)?
5. ¿Tus padres están regularmente contigo durante tus comidas?
6. ¿Regularmente realizas tus comidas a la misma hora?
7. ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas?
8. ¿Desayunas todos los días?
9. ¿Consumes regularmente cereales?
10. ¿Consumes regularmente frutas y verduras?
11. ¿Consumes alimentos con mucha sal?
12. ¿Realizas actividades físicas al menos 3 veces por semana?
13. ¿Ver la televisión es una actividad a la que le dedicas varias horas al día?
14. ¿Utilizas con regularidad la computadora?
15. ¿Utilizas con regularidad juegos electrónicos y de video?
16. ¿Tu papá es gordito?
17. ¿Tu mamá es gordita?
18. ¿Tus dos padres trabajan?
19. ¿Tú y tus padres comparten la misma casa?
20. ¿Tus padres tienen largas jornadas de trabajo o están fuera de casa por periodos largos?
21. ¿Al menos uno de tus padres fuma?

Adicionalmente, cada que se realizaba la encuesta, el médico en turno realizaba una evaluación minuciosa de la condición del niño y añadía el veredicto (IMC, Índice de Masa Corporal) de la condición del infante, el cual podría ser:

- Bajo peso.
- Peso normal.
- Sobrepeso.
- Obesidad.

El número de encuestas de este tipo, con las que actualmente cuenta la estancia son 2166:



	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
1	¿Desayunas	¿Consumes r	¿Consumes r	¿Consumes ¿	¿Realizas act	¿Ver la telev	¿Utilizas con	¿Utilizas con	¿Tu papá es	¿Tu mamá es	¿Tus dos pad	¿Tu y tus pad	¿Tus padres	¿Al menos u	IMC
2	B	B	B	B	B	A	A	A	B	B	A	A	B	B	Bajo peso
3	A	A	A	A	A	B	B	B	A	A	B	A	A	A	Bajo peso
4	A	B	B	B	A	A	A	A	B	B	A	A	B	A	Bajo peso
5	A	A	A	A	A	A	A	A	A	A	A	A	A	A	Bajo peso
6	A	B	B	B	A	B	B	B	B	B	B	A	B	B	Peso normal
7	B	A	A	A	B	A	A	A	A	A	A	B	A	A	Peso normal
8	A	A	A	A	A	A	A	A	A	A	A	A	A	A	Sobrepeso
9	B	B	B	A	B	B	B	B	A	A	B	B	B	A	Sobrepeso
10	A	B	B	A	B	A	A	A	A	A	A	A	A	A	Sobrepeso
11	A	A	A	A	A	A	A	A	A	A	A	A	A	A	Sobrepeso
12	A	A	A	A	B	A	A	A	A	A	A	B	A	A	Obesidad
13	A	B	B	A	B	A	A	A	A	A	A	B	A	A	Obesidad
14	A	A	A	A	B	A	A	A	A	A	A	B	A	A	Obesidad
15	A	B	B	A	B	A	A	A	A	A	A	A	A	A	Obesidad
16	A	A	A	A	B	A	A	A	A	A	A	B	A	A	Obesidad
17	B	A	A	B	B	B	B	B	B	B	B	A	B	B	Obesidad

Donde A=Si y B =No

Tabla característica de atributos:

Atributo	Tabla	Tipo	Total	Nulos	Dist	Media	Desv.E.	Moda	Min	Max
¿Consumes regularmente refrescos?	Fte.Externa	Nominal	2166	2				A	A	B
¿Consumes 1 o más veces a la semana botanas y/o golosinas?	Fte.Externa	Nominal	2166	2				A	A	B
¿Regularmente realizas tus comidas en tu casa?	Fte.Externa	Nominal	2166	2				A	A	B
¿Consumes 3 o más veces a la semana alimentos que no sean caseros (industrializados)?	Fte.Externa	Nominal	2166	2				A	A	B
¿Tus padres están regularmente contigo durante tus comidas?	Fte.Externa	Nominal	2166	2				A	A	B
¿Regularmente realizas tus comidas a la misma hora?	Fte.Externa	Nominal	2166	2				A	A	B
¿Todos los días dedicas regularmente el mismo tiempo para tus comidas?	Fte.Externa	Nominal	2166	2				A	A	B
¿Desayunas todos los días?	Fte.Externa	Nominal	2166	2				A	A	B
¿Consumes regularmente cereales?	Fte.Externa	Nominal	2166	2				B	A	B
¿Consumes regularmente frutas y verduras?	Fte.Externa	Nominal	2166	2				B	A	B
¿Consumes alimentos con mucha sal?	Fte.Externa	Nominal	2166	2				A	A	B
¿Realizas actividades físicas al menos 3 veces por semana?	Fte.Externa	Nominal	2166	2				B	A	B
¿Ver la televisión es una actividad a la que le dedicas varias horas al día?	Fte.Externa	Nominal	2166	2				A	A	B
¿Utilizas con regularidad la computadora?	Fte.Externa	Nominal	2166	2				A	A	B
¿Utilizas con regularidad juegos electrónicos y de video?	Fte.Externa	Nominal	2166	2				A	A	B
¿Tu papá es gordito?	Fte.Externa	Nominal	2166	2				A	A	B
¿Tu mamá es gordita?	Fte.Externa	Nominal	2166	2				A	A	B

¿Tus dos padres trabajan?	Fte.Externa	Nominal	2166	2		A	A	B
¿Tú y tus padres comparten la misma casa?	Fte.Externa	Nominal	2166	2		A	A	B
¿Tus padres tienen largas jornadas de trabajo o están fuera de casa por periodos largos?	Fte.Externa	Nominal	2166	2		A	A	B
¿Al menos uno de tus padres fuma?	Fte.Externa	Nominal	2166	2		A	A	B
IMC	Fte.Externa	Nominal	2166	2		Obesidad	Peso Normal	Obesidad

Se ha decidido emplear métodos de minería de datos, aplicando árboles de decisión para lo cual se empleará el software Weka.

Procedimiento

Importamos nuestros datos que se encuentran en el archivo ***sobrepesoCSV.csv***:

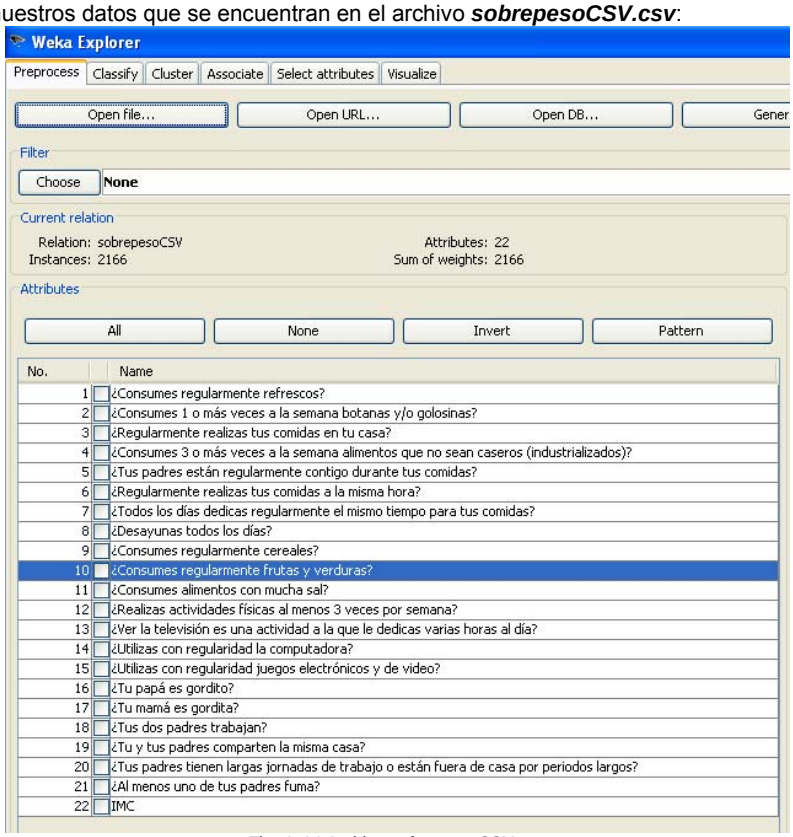


*Fig. 4. 14 Archivo **sobrepesoCSV.csv***

Nos vamos a la pestaña *"Classify"*, damos clic en el botón *"Choose"*

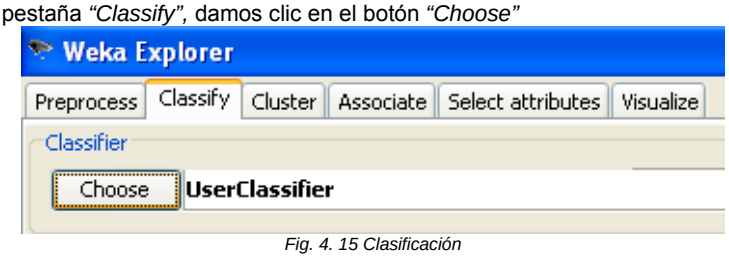


Fig. 4. 15 Clasificación

Seleccionamos dentro del apartado para árboles, el primer modelo

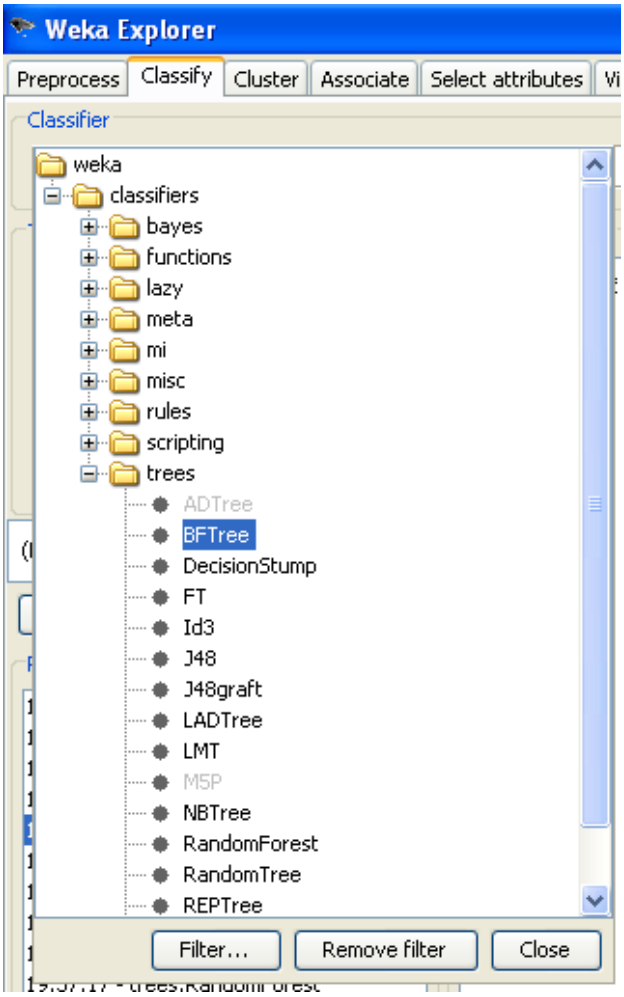


Fig. 4. 16 Selección del modelo

Y posteriormente lo ejecutamos, seleccionando en el apartado de ***"Test options"*** la variable que queremos predecir, en este caso IMC, además de indicarle a Weka que emplee el método de validación cruzada (Cross-validation) para evaluar los resultados que arroje.

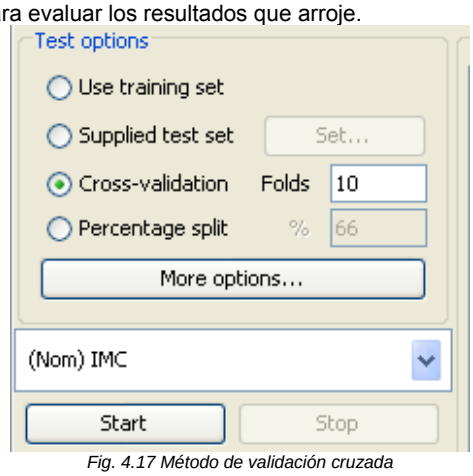


Fig. 4.17 Método de validación cruzada

Repetiremos los tres pasos anteriores para todos los modelos, para posteriormente analizar los resultados obtenidos y tomar la decisión de cuál es el mejor modelo:

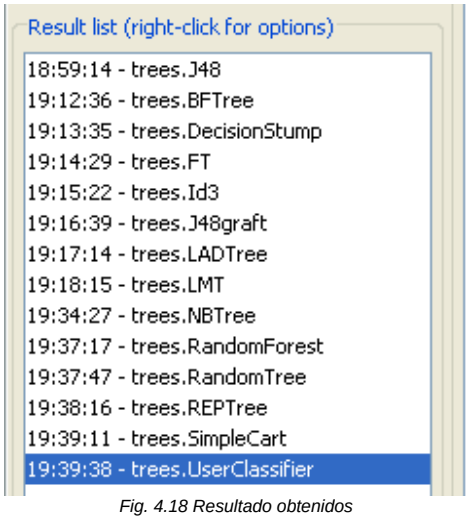


Fig. 4.18 Resultado obtenidos

La siguiente tabla muestra los resultados sobre la correcta clasificación de las instancias:

Modelo – Instancias clasificadas correctamente		
Scheme: trees.J48		
Correctly Classified Instances	**2091**	**96.5374 %**
Scheme: trees.BFTree		
Correctly Classified Instances	**2091**	**96.5374 %**
Scheme: trees.DecisionStump		
Correctly Classified Instances	**1067**	**49.2613 %**
Scheme: trees.FT		
Correctly Classified Instances	**2090**	**96.4912 %**
Scheme: trees.Id3		
Correctly Classified Instances	**2097**	**96.8144 %**
Scheme: trees.J48graft		
Correctly Classified Instances	**2091**	**96.5374 %**
Scheme: trees.LADTree		
Correctly Classified Instances	**1871**	**86.3804 %**
Scheme: trees.LMT		
Correctly Classified Instances	**2098**	**96.8606 %**
Scheme: trees.NBTree		
Correctly Classified Instances	**2091**	**96.5374 %**
Scheme: trees.RandomForest		
Correctly Classified Instances	**2100**	**96.9529 %**
Scheme: trees.RandomTree		
Correctly Classified Instances	**2097**	**96.8144 %**
Scheme: trees.REPTree		
Correctly Classified Instances	**2087**	**96.3527 %**
Scheme: trees.SimpleCart		
Correctly Classified Instances	**2091**	**96.5374 %**
Scheme: trees.UserClassifier		
Correctly Classified Instances	**866**	**39.9815 %**

Podemos ver que sobresalen los modelos ***Id3*** con 96.8144%, ***LMT*** con 96.8606% y ***RandomForest*** con 96.9529%

Una vez examinando a detalles los resultados completos arrojados por los diferentes modelos, llegamos a la conclusión que el mejor y más claro es el ***id3:***

```
=== Run information ===
Scheme:       weka.classifiers.trees.Id3
Relation:     sobrepesoCSV
Instances:    2166
Attributes:   22
              ¿Consumes regularmente refrescos?
              ¿Consumes 1 o más veces a la semana botanas y/o golosinas?
              ¿Regularmente realizas tus comidas en tu casa?
              ¿Consumes 3 o más veces a la semana alimentos que no sean caseros (industrializados)?
              ¿Tus padres están regularmente contigo durante tus comidas?
              ¿Regularmente realizas tus comidas a la misma hora?
              ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas?
              ¿Desayunas todos los días?
              ¿Consumes regularmente cereales?
              ¿Consumes regularmente frutas y verduras?
              ¿Consumes alimentos con mucha sal?
              ¿Realizas actividades físicas al menos 3 veces por semana?
              ¿Ver la televisión es una actividad a la que le dedicas varias horas al día?
              ¿Utilizas con regularidad la computadora?
              ¿Utilizas con regularidad juegos electrónicos y de video?
              ¿Tu papá es gordito?
              ¿Tu mamá es gordita?
              ¿Tus dos padres trabajan?
              ¿Tu y tus padres comparten la misma casa?
              ¿Tus padres tienen largas jornadas de trabajo o están fuera de casa por periodos largos?
              ¿Al menos uno de tus padres fuma?
              IMC
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===
Id3
********************************************
*    lógica de construcción del árbol      *
*    (se agrega en la parte de abajo para su análisis)  *
********************************************

Time taken to build model: 0.05 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances        2097               96.8144 %
Incorrectly Classified Instances        69                3.1856 %
Kappa statistic                          0.9544
Mean absolute error                      0.0202
Root mean squared error                  0.1032
Relative absolute error                  5.7745 %
Root relative squared error             24.6657 %
Total Number of Instances             2166

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.903    0.001      0.995     0.903     0.947      0.993    Bajo peso
                 0.991    0.001      0.995     0.991     0.993      0.995    Peso normal
                 0.997    0.032      0.93      0.997     0.962      0.999    Sobrepeso
                 0.973    0.013      0.98      0.973     0.977      0.997    Obesidad
Weighted Avg.    0.968    0.015      0.97      0.968     0.968      0.997

=== Confusion Matrix ===

   a     b     c     d   <-- classified as
 391   1    26    15 |   a = Bajo peso
   2   215    0     0 |   b = Peso normal
   0     0   648   2  |   c = Sobrepeso
   0     0    23  843|   d = Obesidad
```

Información importante de estos resultados arrojados es el número de instancias clasificadas correctamente:

Correctly Classified Instances	2097	96.8144 %
Incorrectly Classified Instances	69	3.1856 %

La matriz de confusión:

```
=== Confusion Matrix ===

   a    b    c    d   <-- classified as
 391    1   26   15 |  a = Bajo peso
   2  215    0    0 |  b = Peso normal
   0    0  648    2 |  c = Sobrepeso
   0    0   23  843 |  d = Obesidad
```

Y principalmente la ***lógica de construcción del árbol***:

```
¿Consumes 3 o más veces a la semana alimentos que no sean caseros (industrializados)? = B
| ¿Desayunas todos los días? = B
| | ¿Tu y tus padres comparten la misma casa? = A
| | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = A: Bajo peso
| | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = B
| | | | ¿Consumes regularmente refrescos? = B
| | | | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = A
| | | | | | ¿Regularmente realizas tus comidas en tu casa? = B: Bajo peso
| | | | | | ¿Regularmente realizas tus comidas en tu casa? = A: Obesidad
| | | | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = B: Obesidad
| | | | ¿Consumes regularmente refrescos? = A: Bajo peso
| | ¿Tú y tus padres comparten la misma casa? = B
| | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = A
| | | | ¿Ver la televisión es una actividad a la que le dedicas varias horas al día? = A: Bajo peso
| | | | ¿Ver la televisión es una actividad a la que le dedicas varias horas al día? = B: Sobrepeso
| | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = B
| | | | ¿Consumes alimentos con mucha sal? = B
| | | | | ¿Consumes regularmente refrescos? = B
| | | | | | ¿Regularmente realizas tus comidas en tu casa? = B
| | | | | | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = A: Sobrepeso
| | | | | | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = B
| | | | | | | | ¿Realizas actividades físicas al menos 3 veces por semana? = B: Obesidad
| | | | | | | | ¿Realizas actividades físicas al menos 3 veces por semana? = A: Obesidad
| | | | | | ¿Regularmente realizas tus comidas en tu casa? = A: Obesidad
| | | | | ¿Consumes regularmente refrescos? = A: Bajo peso
| | | | ¿Consumes alimentos con mucha sal? = A: Sobrepeso
| ¿Desayunas todos los días? = A
| | ¿Al menos uno de tus padres fuma? = B: Peso normal
| | ¿Al menos uno de tus padres fuma? = A
| | | ¿Consumes regularmente cereales?  = B: Bajo peso
| | | ¿Consumes regularmente cereales?  = A: Peso normal
¿Consumes 3 o más veces a la semana alimentos que no sean caseros (industrializados)? = A
| ¿Tus padres están regularmente contigo durante tus comidas? = B
| | ¿Desayunas todos los días? = B
| | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = A: Bajo peso
| | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = B
| | | | ¿Consumes regularmente refrescos? = B: Peso normal
| | | | ¿Consumes regularmente refrescos? = A
| | | | | ¿Realizas actividades físicas al menos 3 veces por semana? = B
| | | | | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = A
| | | | | | | ¿Consumes regularmente cereales?  = B: Peso normal
| | | | | | | ¿Consumes regularmente cereales?  = A
| | | | | | | | ¿Al menos uno de tus padres fuma? = B: Bajo peso
| | | | | | | | ¿Al menos uno de tus padres fuma? = A: Peso normal
| | | | | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = B: Peso normal
| | | | | ¿Realizas actividades físicas al menos 3 veces por semana? = A: Bajo peso
| | ¿Desayunas todos los días? = A: Obesidad
| ¿Tus padres están regularmente contigo durante tus comidas? = A
| | ¿Consumes regularmente cereales?  = B: Sobrepeso
| | ¿Consumes regularmente cereales?  = A
| | | ¿Consumes regularmente refrescos? = B: Bajo peso
| | | ¿Consumes regularmente refrescos? = A
| | | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = A
| | | | | ¿Realizas actividades físicas al menos 3 veces por semana? = B
| | | | | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = A
| | | | | | | ¿Tú y tus padres comparten la misma casa? = A: Obesidad
| | | | | | | ¿Tú y tus padres comparten la misma casa? = B: Obesidad
```

```
| | | | | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = B: Bajo peso
| | | | | ¿Realizas actividades físicas al menos 3 veces por semana? = A
| | | | | | ¿Tú y tus padres comparten la misma casa? = A
| | | | | | | ¿Regularmente realizas tus comidas en tu casa? = B: Sobrepeso
| | | | | | | ¿Regularmente realizas tus comidas en tu casa? = A
| | | | | | | | ¿Ver la televisión es una actividad a la que le dedicas varias horas al día? = A
| | | | | | | | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = A: Sobrepeso
| | | | | | | | | ¿Consumes 1 o más veces a la semana botanas y/o golosinas? = B: Sobrepeso
| | | | | | | | ¿Ver la televisión es una actividad a la que le dedicas varias horas al día? = B: Bajo peso
| | | | | | ¿Tú y tus padres comparten la misma casa? = B: Obesidad
| | | | ¿Todos los días dedicas regularmente el mismo tiempo para tus comidas? = B
| | | | | ¿Tú y tus padres comparten la misma casa? = A: Sobrepeso
| | | | | ¿Tú y tus padres comparten la misma casa? = B: Bajo peso
```

El árbol obtenido mediante Weka se muestra gráficamente en la siguiente página:

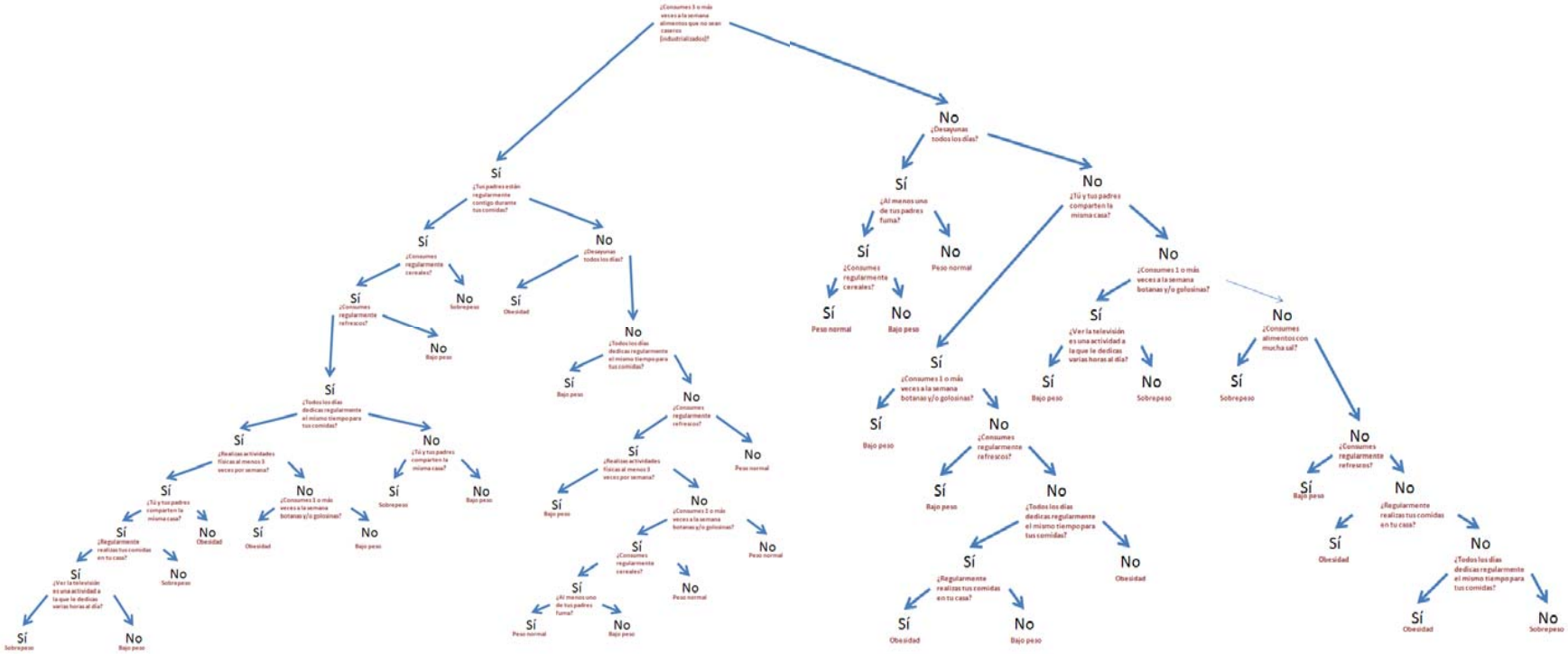


Fig. 4.19 árbol de Decisión en Weka

Clustering

El agrupamiento (clustering) o también llamado “segmentación” o “aglomeramiento”, es una técnica descriptiva que consiste en obtener grupos “naturales” a partir de los datos. Se crean grupos formados por objetos muy similares entre sí, y que al mismo tiempo sean muy diferentes a los objetos de otro grupo.

Lo importante del agrupamiento respecto a la clasificación es que a diferencia de la clasificación, se desconoce cómo son los grupos y en ocasiones el número de ellos. Son los grupos y la pertenencia a los grupos los que se quiere determinar.
Algunas veces se determina el número de grupos con ayuda de un algoritmo de agrupamiento, según las características de los datos.
Posteriormente al agrupamiento, se toman decisiones diferentes para cada grupo.

El agrupamiento se puede utilizar para reducir un conjunto de datos de miles de ejemplos a media docena de grupos, y así entender mejor los datos originales, estos grupos sirven como resumen de los datos originales, de hecho, muchos autores consideran el agrupamiento con este objetivo como una tarea nueva, llamada sumarización.
De modo similar si buscamos subgrupos que se separen del resto de la población, tenemos lo que se conoce como “descubrimiento de subgrupos” que también se suele considerar una tarea en sí misma.

Ejemplo 4: Clustering

Justificación

Nuestra estancia decidió utilizar el software Weka de minería de datos para intentar sacar provecho de los registros con los que cuenta actualmente acerca de las enfermedades que afectan a los infantes de nuestra estancia.

Nuestra estancia
La tarea consistió en encontrar las enfermedades que en mayor medida afectan a nuestros infantes para tomar las medidas pertinentes para prevenirlas e intentar erradicarlas en la medida de lo posible.

Se decidió emplear el método clustering con el algoritmo "**SimpleKMeans**" que está dentro del software de minería de datos "**weka**".
Los datos obtenidos (vista minable), fueron obtenidos de los infantes de la estancia, dando un total de 2202 incidencias de infantes enfermos.
Esta información proviene del OLTP de nuestra estancia y las tablas que se emplearon para obtener esta información fueron: INFANTES, EXPEDIENTES_MEDICO_GUARDERIA, RECETAS, DETALLES_RECETAS y MEDICAMENTOS.

	A	B	C	D	E
1	id	curp	id_infante	enfermedad	medicamento
2	1318	RASI821008MJCMRV05	A1102	ALERGIAS	A22
3	1313	SOCF050808MDFSHB08	A1536	ALERGIAS	A22
4	1274	CAEJ040208MMCMSL06	A101	AMIGDALITIS AGUDA	A9
5	1231	SASV820808HJCNAC01	A1091	AMIGDALITIS AGUDA	A9
6	1041	SASM821208MJCLNR09	A1111	AMIGDALITIS AGUDA	A9
7	885	VESE830108HJCLLR00	A1116	AMIGDALITIS AGUDA	A8
8	890	MERF880208HJCJVR09	A1138	AMIGDALITIS AGUDA	A7
9	1309	CAIJ950408HJCSSM07	A1285	ALERGIAS	A22
10	1321	RAGA070708HJCMRD07	A1455	ALERGIAS	A22
11	1305	ZUMV080308HDFRRC00	A1590	ALERGIAS	A21
12	1140	CASM821208MJCTNY01	A1106	AMIGDALITIS AGUDA	A9
13	1267	SARR830508HJCLDD05	A1124	AMIGDALITIS AGUDA	A9

Tabla característica de atributos:

Atributo	Tabla	Tipo	Total	Nulos	Dist	Media	Desv. E.	Moda	Min	Max
id_infante	Fte.Externa	nominal	2202	0	2167			*	AAAA71108MDFLBN07	ZUVU690108MMCXNN05
enfermedad	Fte.Externa	nominal	2202	0	6			Estreñimiento	Alergias	Estreñimiento
medicamento	Fte.Externa	nominal	2202	0	16			A2	A1	A34

El uso del software Weka fue el siguiente:
Arrancamos el programa y seleccionamos la opción *"Explorer":*

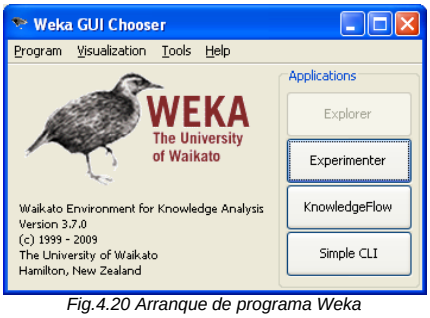


Fig.4.20 Arranque de programa Weka

Damos clic en *"Open file"* y seleccionamos el archivo CSV que contiene nuestra vista minable:

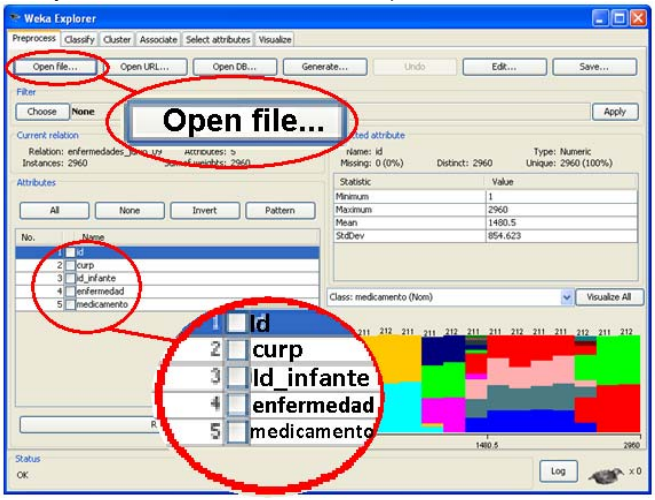


Fig.4.21 Seleccionando el archivo

Removemos algunos datos que no se necesitan de la vista, esto se hace seleccionando los atributos que deseamos borrar y posteriormente dando clic en el botón *"Remove":*

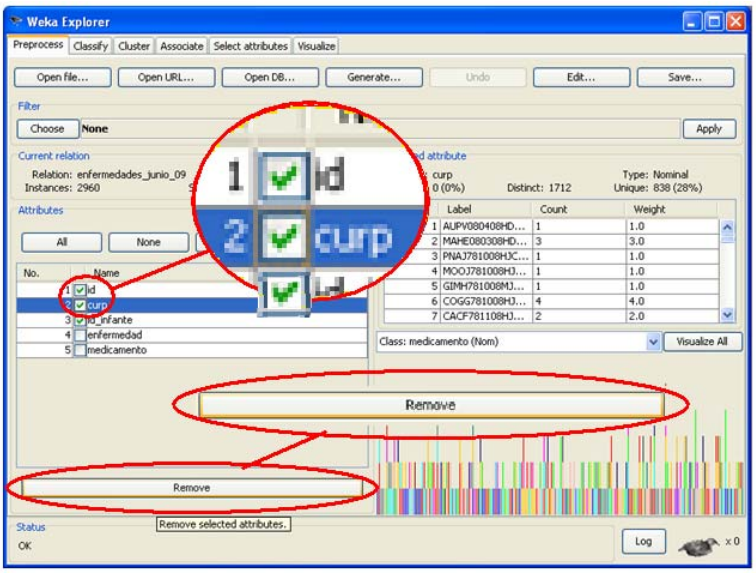


Fig.4.22 Removiendo datos no necesarios

Damos clic en la pestaña *"Cluster":*

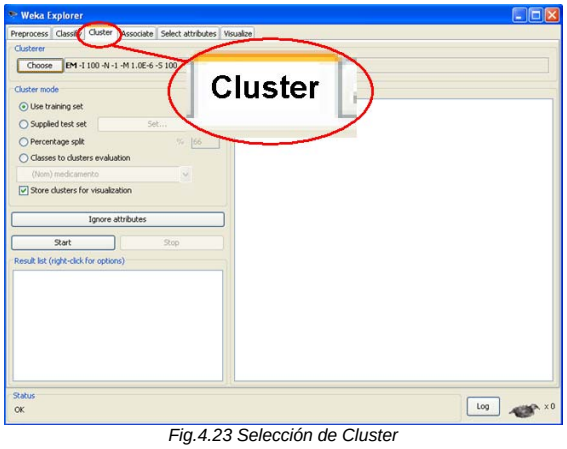


Fig.4.23 Selección de Cluster

Ahora seleccionamos el algoritmo a emplear dando clic en el botón *"Choose"* y seleccionando *"SimpleKmeans":*

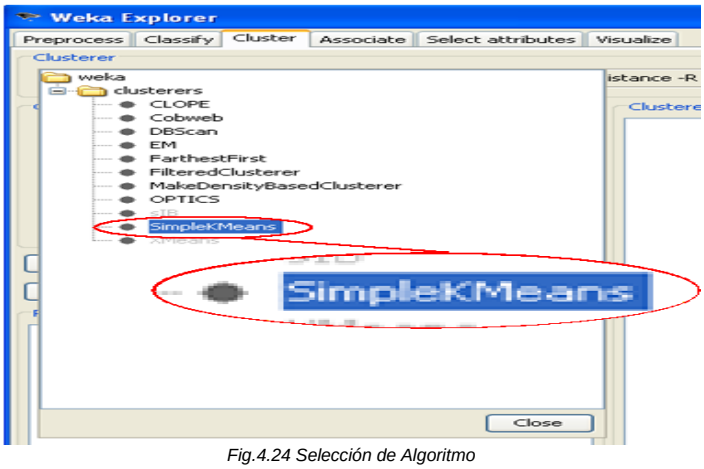


Fig.4.24 Selección de Algoritmo

Seleccionamos la opción *"Classes to clusters evaluation"* y seleccionamos la opción *"(Nom) enfermedad"*:

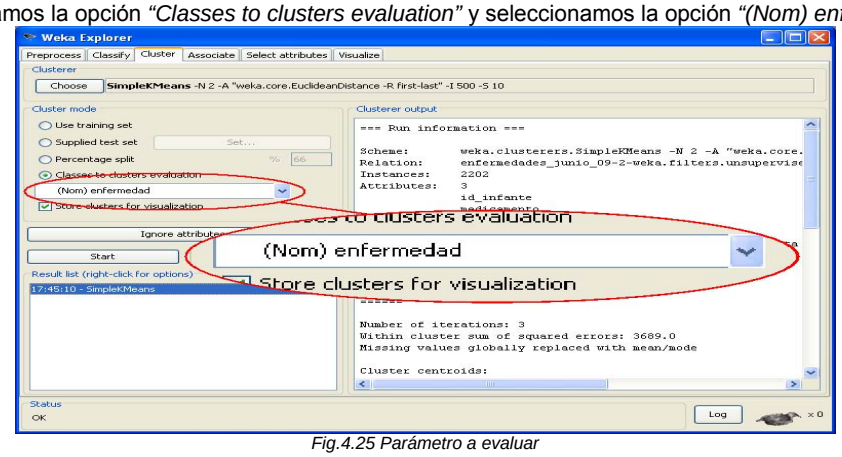


Fig.4.25 Parámetro a evaluar

Finalmente damos clic en el botón *"Start"* y Weka nos arroja los siguientes resultados:

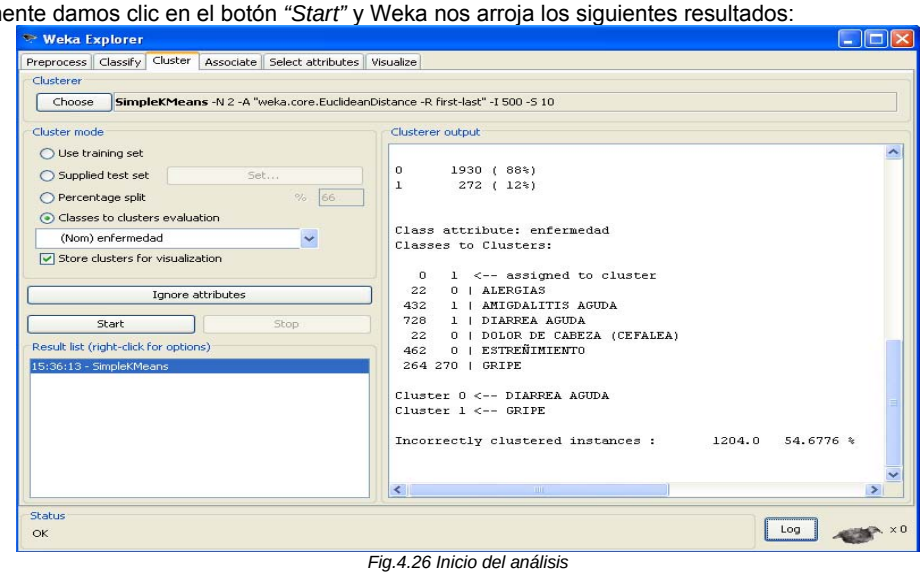


Fig.4.26 Inicio del análisis

Los resultados completos que nos arroja Weka son:

```
=== Run information ===
Scheme:       weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     enfermedades_junio_09-2-weka.filters.unsupervised.attribute.Remove-R1-2
Instances:    2202
Attributes:   3
              id_infante
              medicamento
Test mode:    Classes to clusters evaluation on training data
=== Model and evaluation on training set ===
kMeans
======
Number of iterations: 3
Within cluster sum of squared errors: 3862.0000000000005
Missing values globally replaced with mean/mode

Cluster centroids:
                                Cluster#
Attribute          Full Data          0          1
                      (2202)     (1930)     (272)
=============================================
id_infante            A1167      A1536      A176
medicamento             A34        A33       A34

Clustered Instances

0      1930 ( 88%)
1       272 ( 12%)

Class attribute: enfermedad
Classes to Clusters:

  0   1    <-- assigned to cluster
 22   0    | ALERGIAS
432   1    | AMIGDALITIS AGUDA
728   1    | DIARREA AGUDA
 22   0    | DOLOR DE CABEZA (CEFALEA)
462   0    | ESTREÑIMIENTO
264 270 | GRIPE

Cluster 0 <-- DIARREA AGUDA
Cluster 1 <-- GRIPE

Incorrectly clustered instances :          1204.0      54.6776 %
```

Información importante de lo arrojado por Weka es:

- Se detectaron 2 clústeres, el clúster 0 que corresponde a “DIARREA AGUDA” y el clúster 1 que corresponde a “GRIPE”:

Cluster 0 (88%)	**Cluster 1 (12%)**	<-- assigned to cluster
22	0	| ALERGIAS
432	1	| AMIGDALITIS AGUDA
728	1	| DIARREA AGUDA
22	0	| DOLOR DE CABEZA (CEFALEA)
462	0	| ESTREÑIMIENTO
264	270	| GRIPE

Clustered Instances
0 1930 (88%)
1 272 (12%)

∑ 100%)

- Podemos ver que porcentualmente la gripa no representa un gran problema, que solo afecta al 12% de los infantes, además de que por el mes en que se tomaron las muestras es normal la aparición de esta enfermedad.
- El punto que tenemos que tomar con más cuidado es la aparición tan recurrente de casos de “diarrea aguda”, ya que está siendo presentado por el 88% de nuestros infantes.

Reglas de asociación (descriptiva)

Similar a las correlaciones, tiene como objetivo identificar relaciones no explícitas entre atributos categóricos.

La formulación más común es del estilo “si el atributo X toma el valor d entonces el atributo Y toma el valor b”.

Las reglas de asociación no implican una relación causa-efecto, es decir, puede no existir una causa para que los datos estén asociados. Este tipo de tarea se utiliza frecuentemente en el análisis de la cesta de la compra, para identificar productos que son frecuentemente comprados juntos, información que puede usarse para ajustar los inventarios, para la organización física del almacén o en campañas publicitarias.

Tienen por objetivo descubrir reglas que muestran condiciones del tipo atributo-valor que ocurren frecuentemente en un conjunto de datos.

Ejemplo 5: Reglas de asociación

Justificación
Las enfermedades hereditarias son un conjunto de enfermedades genéticas caracterizadas por transmitirse de generación en generación, es decir de padres a hijos, en la descendencia.
Una de las enfermedades más comunes de heredar es la Diabetes, en cualquiera de sus múltiples manifestaciones o tipos de ella. La Hemofilia es otra de las enfermedades hereditarias reconocidas por todos y se debe a la deficiencia en alguno de los factores necesarios para la correcta coagulación de la sangre, lo que genera pérdidas de sangre superiores a las normales en, por ejemplo, hematomas y en procedimientos dentales. La Anemia Falciforme es otra enfermedad hereditaria muy común, que afecta directamente la hemoglobina, la proteína que forma se encuentra en los glóbulos rojos y su función principal es la de trasportar el oxígeno en ellos. Otras enfermedades hereditarias son las cardiovasculares. Éstas se manifiestan a cualquier edad y todos poseeos probabilidad de contraerlas. A modo de ejemplo, si en una familia varios integrantes han sufrido alguna anomalia, es muy alta la probabilidad que a nosotros nos suceda lo mismo.
Como los padecimiento mencionados existe un gran número de enfermedades hereditaria que se pueden presentar por factores genéticos.

Nuestra estancia
*Ya que se trata de un problema en México en el mundo difícil de prevenir, en la estancia se aplico el algoritmo de reglas de asociación para predecir l**as enfermedades hereditarias más comunes dentro de la estancia.***
El query que utilizamos para obtener la vista minable desde nuestro datamart llamado Expediente es:

```
SELECT DISTINCT N.nombre,M.curp,E.nombre,P.nombre, EX.num_enfermedad_antecedente,s.sano
FROM enfermedad E,parentesco P, estado_de_salud S, expediente EX, menor M, nivel N
WHERE id_año                =1998
AND   E.num_enfermedad =Ex.num_enfermedad_antecedente
AND   EX.id_parentesco   =P.id_parentesco
AND   EX.id_nivel           =M.id_nivel
AND   N.id_nivel            =M.id_nivel
```

Tomamos una muestra de la vista minable, tenemos que luce así:

Nivel_nombre	CURP	Nombre_enfermedad	Nombre_parentesco	num_enfermedad	Sano
PREESCOLAR	AEPD950618MDFREN00	ASMA	Padre	9	N
PREESCOLAR	BADP950629MDFHLT00	HIPERTENSION ARTERIAL	Padre	21	N
PREESCOLAR	BADP950629MDFHLT00	HIPERTENSION ARTERIAL	Padre	21	S
PREESCOLAR	BADR970927HDFRLL01	DIABETES	Padre	22	N
PREESCOLAR	BADR970927HDFRLL01	DIABETES	Padre	22	S
LACTANTE	BADR970927HDFRLL01	CARDIOVASCULAR	Abuelo materno	23	N
LACTANTE	BADR970927HDFRLL01	CARDIOVASCULAR	Abuelo materno	23	S
MATERNAL	BADR970927HDFRLL01	CARDIOVASCULAR	Abuelo materno	23	N
MATERNAL	BADR970927HDFRLL01	CARDIOVASCULAR	Abuelo materno	23	S
PREESCOLAR	BALR940304MDFROQ01	HIPERTENSION ARTERIAL	Madre	21	N
PREESCOLAR	BALR940304MDFROQ01	HIPERTENSION ARTERIAL	Madre	21	S
PREESCOLAR	BAMM940220MDFRRR07	HEMOFILIA	Abuelo materno	24	N
PREESCOLAR	BAMM940220MDFRRR07	HEMOFILIA	Abuelo materno	24	S
PREESCOLAR	BEGJ940223MDFRNL04	ASMA	Abuelo materno	9	N
PREESCOLAR	BEGJ940223MDFRNL04	ASMA	Abuelo materno	9	S
LACTANTE	CACP970430MDFLBL02	ASMA	Padre	9	N
MATERNAL	CACP970430MDFLBL02	ASMA	Padre	9	N
PREESCOLAR	CACP970430MDFLBL02	ASMA	Padre	9	N
PREESCOLAR	CACP970430MDFLBL02	ASMA	Padre	9	S
PREESCOLAR	CAMA940211HDFSRD00	HEMOFILIA	Abuela paterna	24	N
PREESCOLAR	CAMA940211HDFSRD00	HEMOFILIA	Abuela paterna	24	S

Tabla característica de atributos:

Atributo	Tabla	Tipo	Total	Nulos	Dist	Media	Desv.E.	Moda	Min	Max
Nivel	Nivel	nominal	205	0	3	*	*	PREESCOLAR	MATERNAL	PREESCOLAR
CURP	Menor	nominal	205	0	205	*	*	*	*	*
Nombre_enfermedad	Enfermedad	nominal	205	0	7	*	*	Diabetes	Falciforme	Diabetes
Nombre_Parentesco	Parentesco	nominal	205	0	4			Abuelo paterno	Abuela materna	Abuelo paterno
Num_enfermedad	expediente	integer	205	0	7	*	*	22	24	22
Sano	Estado_salud	nominal	205	0	2	*	*	N	S	

Siendo así vamos a utilizar el modelo de clasificación con la técnica basada en Reglas de asociación

Procedimiento

- Utilizando **RAPIDMINER, importamos el archivo Reglas_Asociaion.xls**

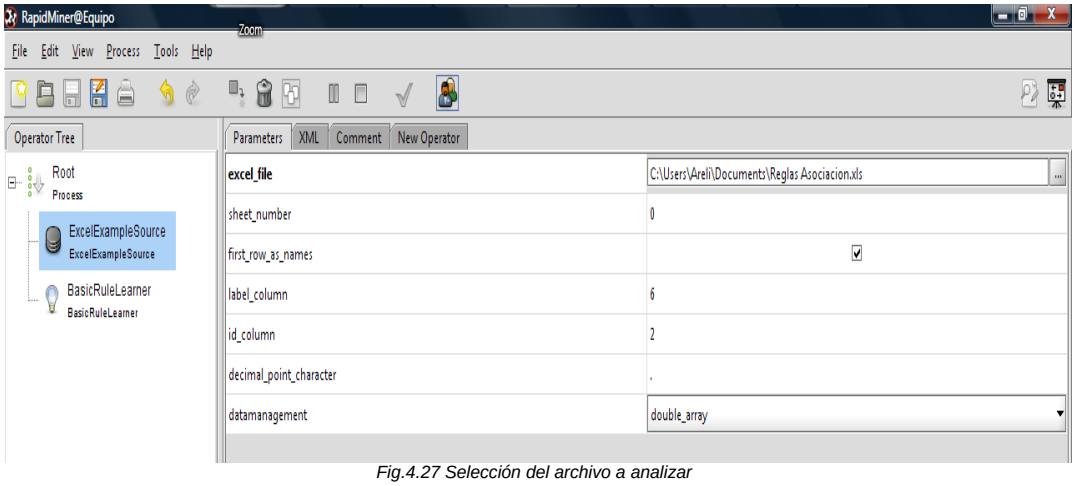
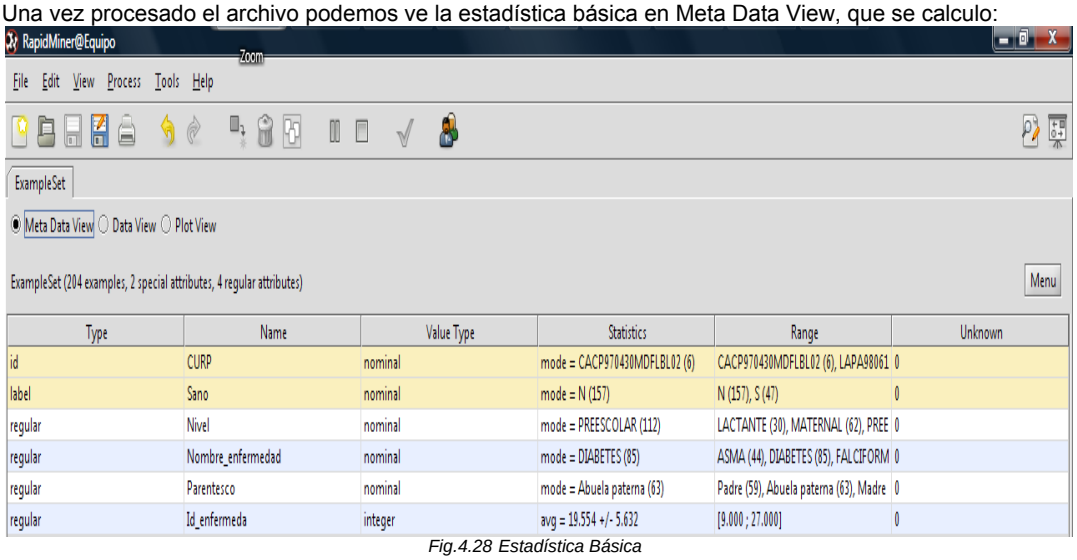


Fig.4.27 Selección del archivo a analizar

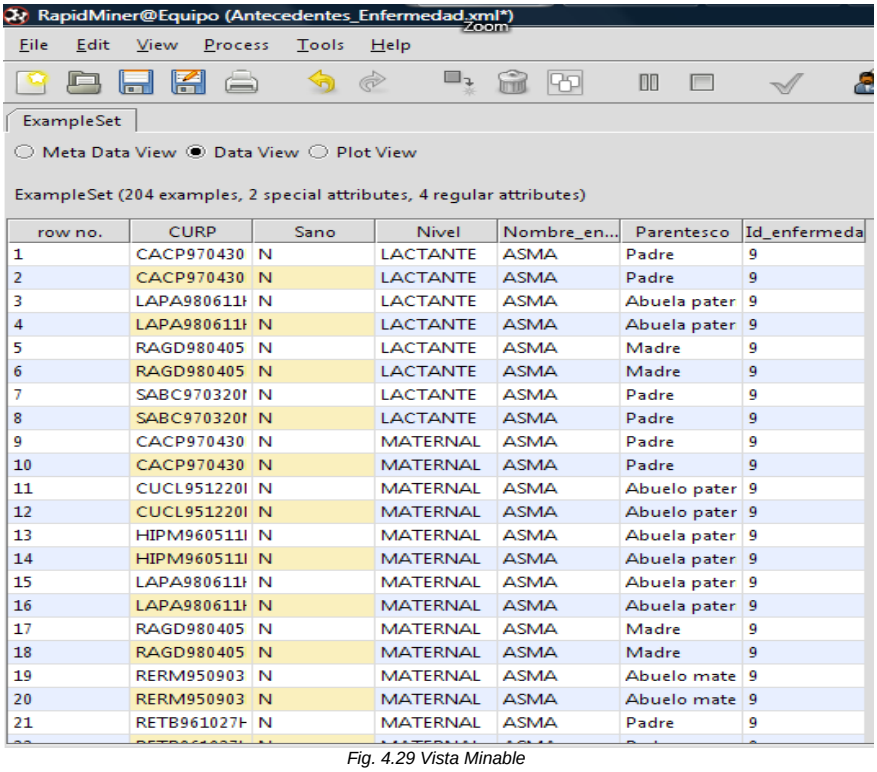
Una vez procesado el archivo podemos ve la estadística básica en Meta Data View, que se calculo:



Type	Name	Value Type	Statistics	Range	Unknown
id	CURP	nominal	mode = CACP970430MDFLBL02 (6)	CACP970430MDFLBL02 (6), LAPA98061	0
label	Sano	nominal	mode = N (157)	N (157), S (47)	0
regular	Nivel	nominal	mode = PREESCOLAR (112)	LACTANTE (30), MATERNAL (62), PREE	0
regular	Nombre_enfermedad	nominal	mode = DIABETES (85)	ASMA (44), DIABETES (85), FALCIFORM	0
regular	Parentesco	nominal	mode = Abuela paterna (63)	Padre (59), Abuela paterna (63), Madre	0
regular	Id_enfermeda	integer	avg = 19.554 +/- 5.632	[9.000 ; 27.000]	0

Fig.4.28 Estadística Básica

La vista minable que cargamos en la opción Data View



row no.	CURP	Sano	Nivel	Nombre_en...	Parentesco	Id_enfermeda
1	CACP970430	N	LACTANTE	ASMA	Padre	9
2	CACP970430	N	LACTANTE	ASMA	Padre	9
3	LAPA980611H	N	LACTANTE	ASMA	Abuela pater	9
4	LAPA980611H	N	LACTANTE	ASMA	Abuela pater	9
5	RAGD980405	N	LACTANTE	ASMA	Madre	9
6	RAGD980405	N	LACTANTE	ASMA	Madre	9
7	SABC970320I	N	LACTANTE	ASMA	Padre	9
8	SABC970320I	N	LACTANTE	ASMA	Padre	9
9	CACP970430	N	MATERNAL	ASMA	Padre	9
10	CACP970430	N	MATERNAL	ASMA	Padre	9
11	CUCL951220I	N	MATERNAL	ASMA	Abuelo pater	9
12	CUCL951220I	N	MATERNAL	ASMA	Abuelo pater	9
13	HIPM960511I	N	MATERNAL	ASMA	Abuela pater	9
14	HIPM960511I	N	MATERNAL	ASMA	Abuela pater	9
15	LAPA980611H	N	MATERNAL	ASMA	Abuela pater	9
16	LAPA980611H	N	MATERNAL	ASMA	Abuela pater	9
17	RAGD980405	N	MATERNAL	ASMA	Madre	9
18	RAGD980405	N	MATERNAL	ASMA	Madre	9
19	RERM950903	N	MATERNAL	ASMA	Abuelo mate	9
20	RERM950903	N	MATERNAL	ASMA	Abuelo mate	9
21	RETB961027H	N	MATERNAL	ASMA	Padre	9

Fig. 4.29 Vista Minable

Además de también observar algunas graficas de distribución de algunos atributos de nuestra vista por ejemplo:

- La siguiente Gráfica muestra según el parentesco del infante cuantos de ellas ya tienen la enfermedad hereditaria , N (rojo) indica los que no están sanos y S (azul) indica cuantos si están sanos

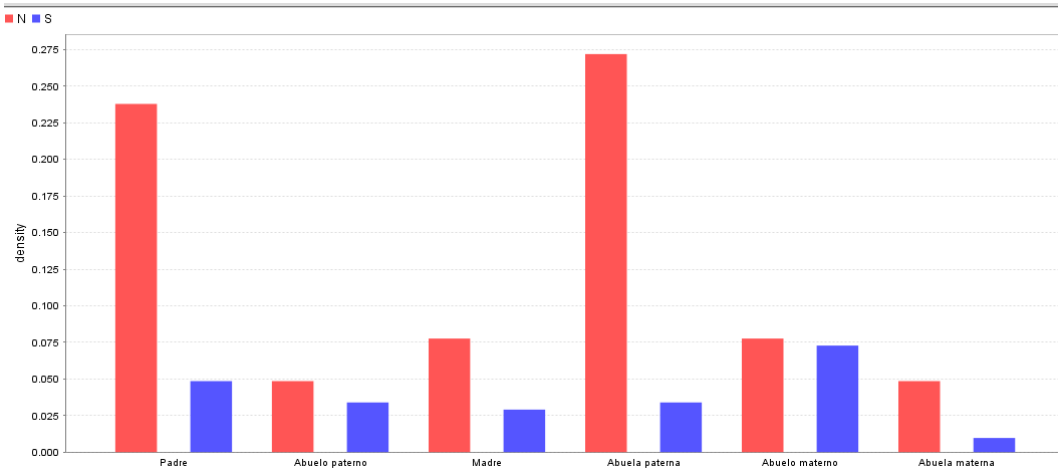


Fig 4.30 Parentesco vs Estado de Salud

- Esta grafica muestra las enfermedades que se presentan en cada nivel, donde podemos observar también que la diabetes es la enfermedad hereditaria más común

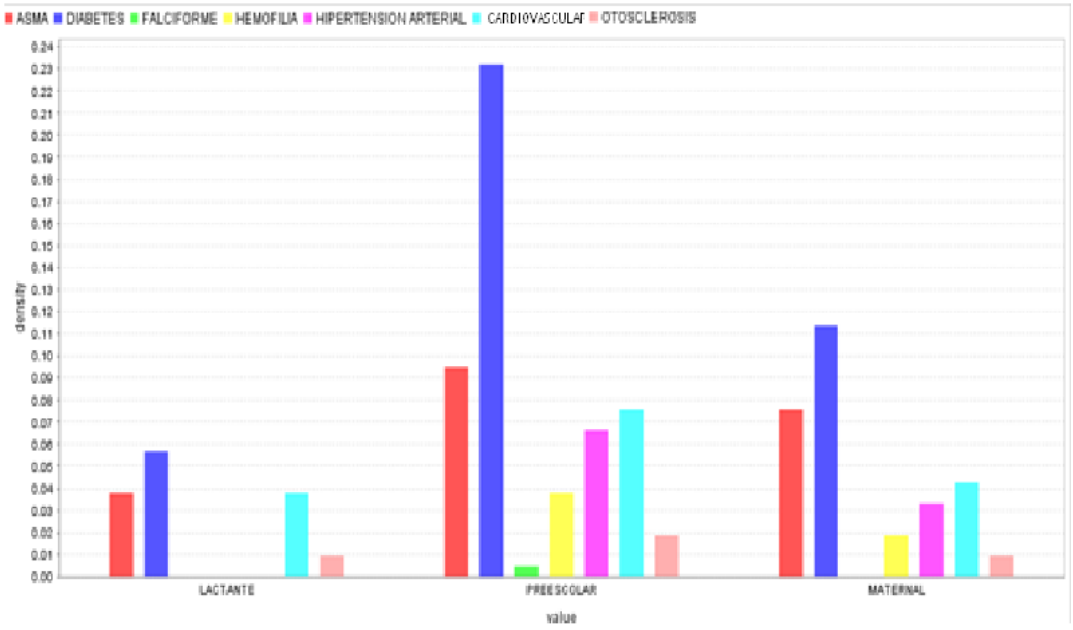


Fig. 4.31 Nivel Vs Enfermedad

Por ultmo esta grafica muestra las de que parte del arbol genealogico son mas frecuentes algunas enfemedades
Por ejemplo:

- El Asma es heredada por los infantes con mayores proporciones del padre
- La diabetes es heredada en un mayor número de casos por la abuela paterna

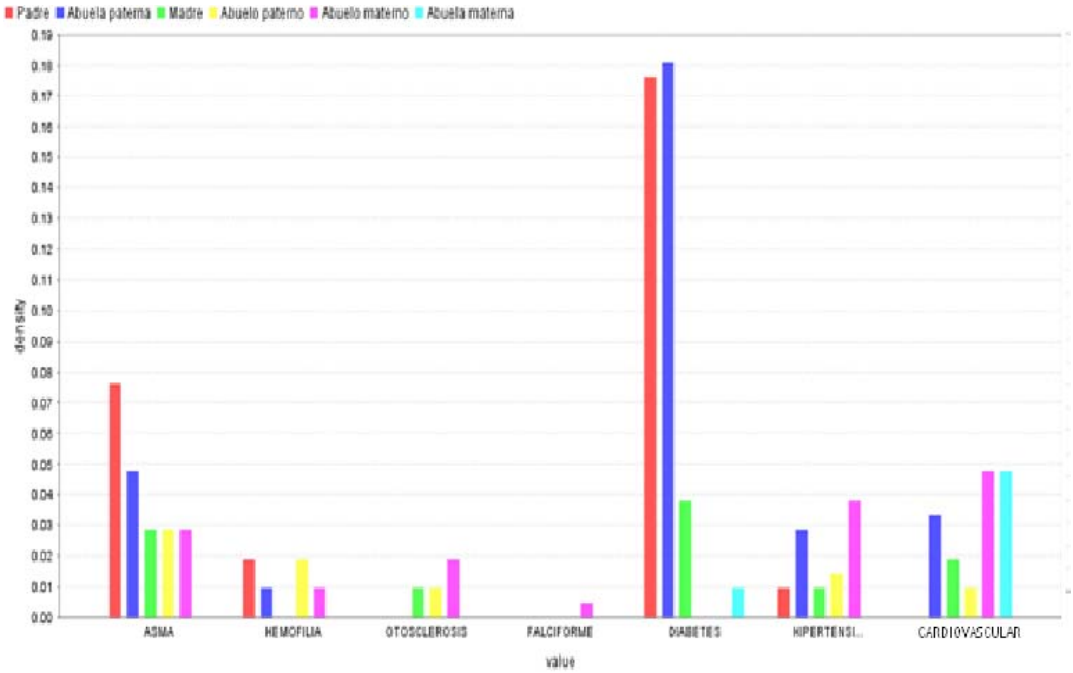


Fig. 4.32 Enfermedad vs Parentesco

Ahora para procesar nuestra vista con ayuda del algoritmo de Reglas de Asociación: BasicRuleLearner

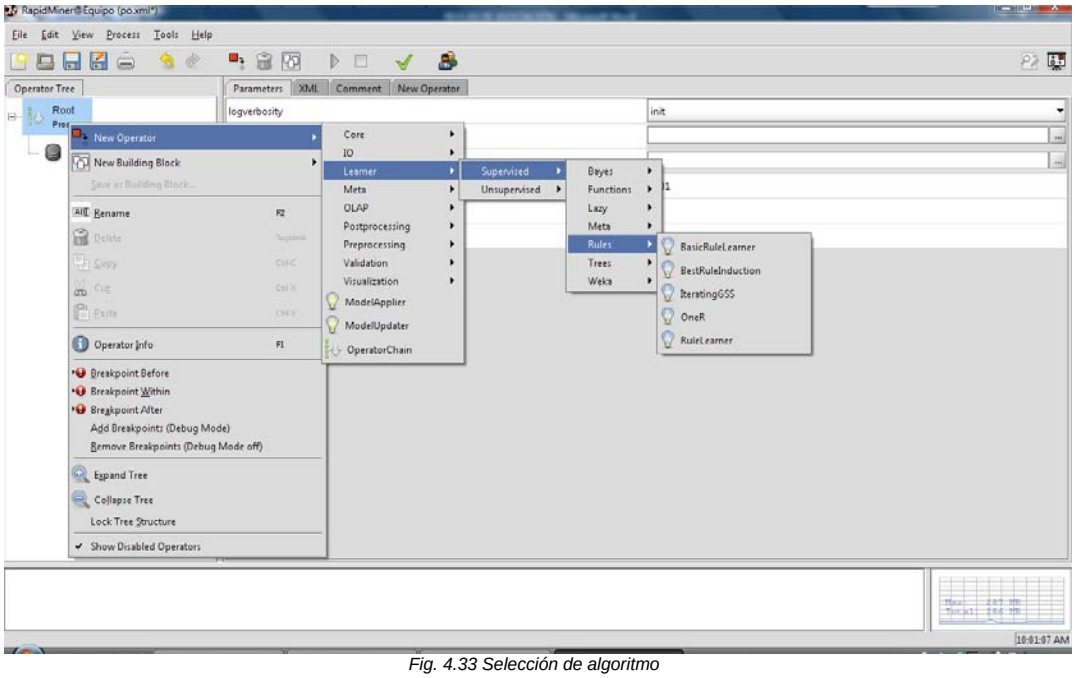


Fig. 4.33 Selección de algoritmo

Una vez seleccionado damos clic en ejecutar 

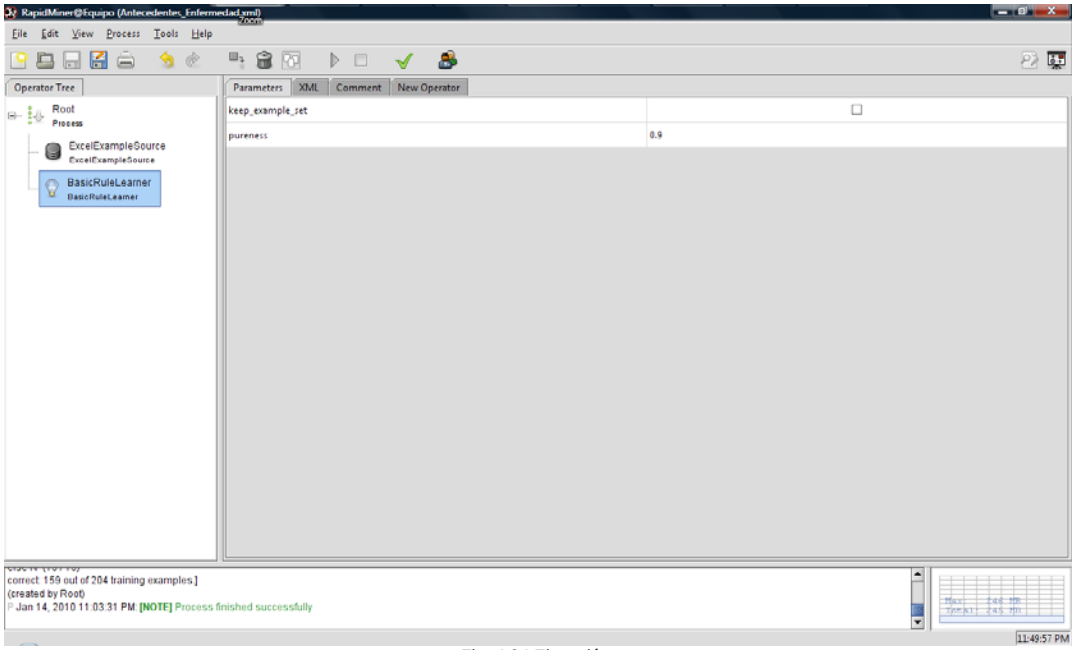


Fig. 4.34 Ejecución

Esta es la salida que nos nuestra una vez ejecutado el algoritmo:

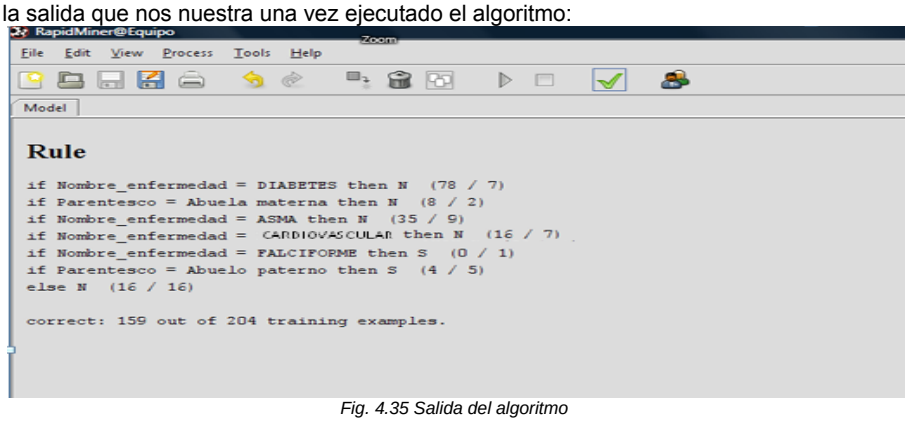


Fig. 4.35 Salida del algoritmo

Con lo cual podemos observar que la Diabetes es la enfermedad hereditaria más común entre los infantes de la estancia y también la que se hereda con mayor frecuencia, también observamos que el abuelo paterno es la persona de donde provienen el mayor número de enfermedades hereditarias de nuestra estancia

Conclusión:

Dados los resultados obtenidos, ya que las enfermedades hereditarias se recomienda tener en cuenta los factores de riesgo de cada una de las enfermedades, por ejemplo si a un infante ya le dio Diabetes, se le debe suministrar una dieta diferente, para tratar de evitar su futuro padecimiento y así con cada una de las enfermedades.

Correlaciones (descriptiva)

Se usan para examinar el grado de similitud de los valores de dos variables numéricas, para medir la correlación lineal es con el coeficiente de correlación r, el cual es un valor real comprendido entre -1 y 1. Si es 0 no hay correlación, cuando r es positivo, las variables tienen un comportamiento similar (ambas crecen o decrecen al mismo tiempo) y cuando r es negativo si una variable crece la otra decrece.
Ejemplo: Un inspector de incendios obtiene correlaciones negativas entre el empleo de aisladores y la frecuencia de incendios.

Cuando se estudian dos variables (X, Y) o tres variables (X, Y, Z) es importante obtener una medida de la dependencia o medida de la relación entre esas variables.

Para estudiar y medir esta relación, el primer paso consistirá en recoger los datos que muestren los correspondientes valores de las variables consideradas y en representarlas después mediante un diagrama de dispersión (el alumno puede consultar este tipo de gráficos en la práctica relativa al cálculo de estadísticos descriptivos y gráficos). Esta representación gráfica es la que más se utiliza en el estudio de la dependencia de dos o tres variables y resulta muy útil como análisis previo a la ejecución de procedimientos de correlación.

Ejemplo 6: Correlaciones

Justificación

La estatura humana y peso varían de acuerdo con genética, *nutrición*, *dieta*, ejercicio y las condiciones de vida presentes antes de la edad adulta, cuando el crecimiento se detiene, constituyen el determinante ambiental.

La mayoría de las personas sabemos cuando estamos excedidas de peso y cuando no. Es una cuestión de conocer nuestros cuerpos y cómo funcionan y cuándo están bien o mal. La estatura tiene mucho que ver con el peso corporal de cada persona.

Nuestra estancia

El 5 % de los niños no desarrollan una estatura y peso óptimos en México razón por la cual hemos decidido hacer un estudio que de alguna manera pueda reducir este porcentaje.
Para esto asociaremos dos variables que tenemos en nuestro OLAP de Expediente, estas son Peso y Estatura.

6. Asociar el peso con la estatura de los infantes.

Para realizar el análisis entre dichos atributos necesitamos una vista que contenga:
El Nivel, estatura, y su peso promedio de un periodo del infante.
Query utilizado para obtener la vista minable:

SELECT distinct curp, peso, estatura, genero, id_nivel
FROM menor;
Tomando una muestra de la vista minable, tenemos que luce así:

CURP	PESO	ESTATURA	GENERO	ID_NIVEL
AAAA571116MDFLBN07	19.3	1.1	F	8
AAAE620531MJCDRS01	9.4	.8	F	3
AAAE620630HJCBRR02	9.1	.8	M	3
AAAE630521HJCLGR00	5.4	.6	M	1
AAAG070105HDFLVR09	11.3	.8	M	4
AAAG070412HJCLVR09	9.4	.7	M	3
AAAH780106HDFLCG00	18.5	1.1	M	8
AAAM820518MDFRRR02	7.3	.6	F	2
AABJ690507HJCNLH09	17.5	1.1	M	8
AABV790612HMCLDC03	10.5	.8	M	3
AACA590622HJCRLL09	17.4	1.0	M	7
AACA780812HDFLSL09	17.4	1.1	M	8
AACI710905HJCMRC09	15.5	1.0	M	7

Tabla característica de atributos:

Atributo	Tabla	Tipo	Total	Nulos	Dist	Media	Desv.E.	Moda	Min	Max
Id_nivel	Menor	numérico	1500	0	8	5	1.82356985	5	2	8
curp_infante	Menor	Nominal	1500	0	210	*	*	*	*	*
Peso	Menor	Numérico	1500	0	20	10.33	1.734	17.4	5.2	20.5
Estatura	Menor	numérico	1500	0	4	.88	1.98759	1.1	.5	1.5
Genero	Menor	nominal	1500	0	4	*	*	*	*	*

Procedimiento

Para este ejemplo Utilizaremos **Statistical Package for the Social Sciences (SPSS)** se trata de un programa estadístico informático muy utilizado en empresas para investigación de mercado, con el cuál podremos observar la asociación que estamos buscando; entre el consumo de alimentos en cada nivel y la estatura de los infantes.

Para esto lo primero que haremos es, una vez abierto nuestro entorno de SPSS, abrimos el asistente para la importación de texto adjuntando nuestro archivo estaturaPeso.csv en el que se encuentra nuestra vista minable.
Una vez seleccionado continuaremos con el llenado de un formulario que consta de 6 pasos y que nos permitirá especificar las características de nuestra vista minable, como se muestra en la Fig. 4.36:
En el primer paso seleccionamos **no** pues no contamos con un formato definido anteriormente.

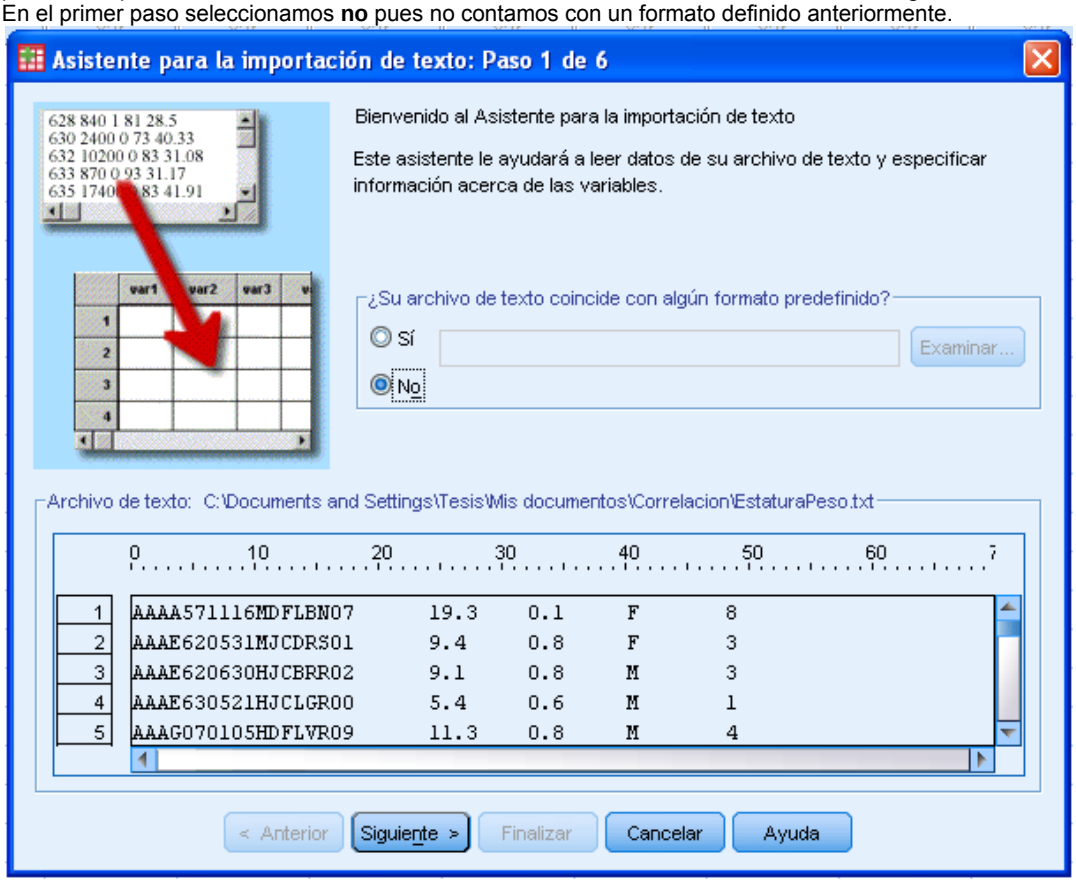


Fig. 4.36 Vista minable

Luego seleccionamos el delimitador y especificamos donde empieza el archivo:

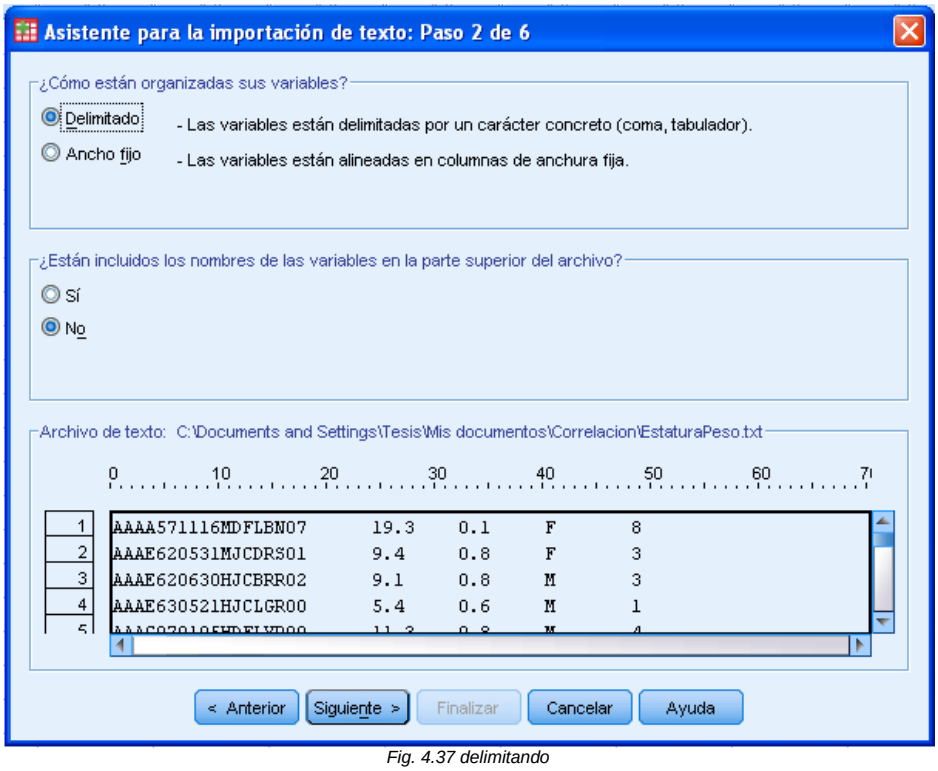


Fig. 4.37 delimitando

A continuación, seleccionamos la opción que dice: **cada línea representa un caso**, pues para nuestro ejemplo se trata de una toma de medida de estatura cada caso en diferentes periodos, e **importamos los primero 1500 casos**.

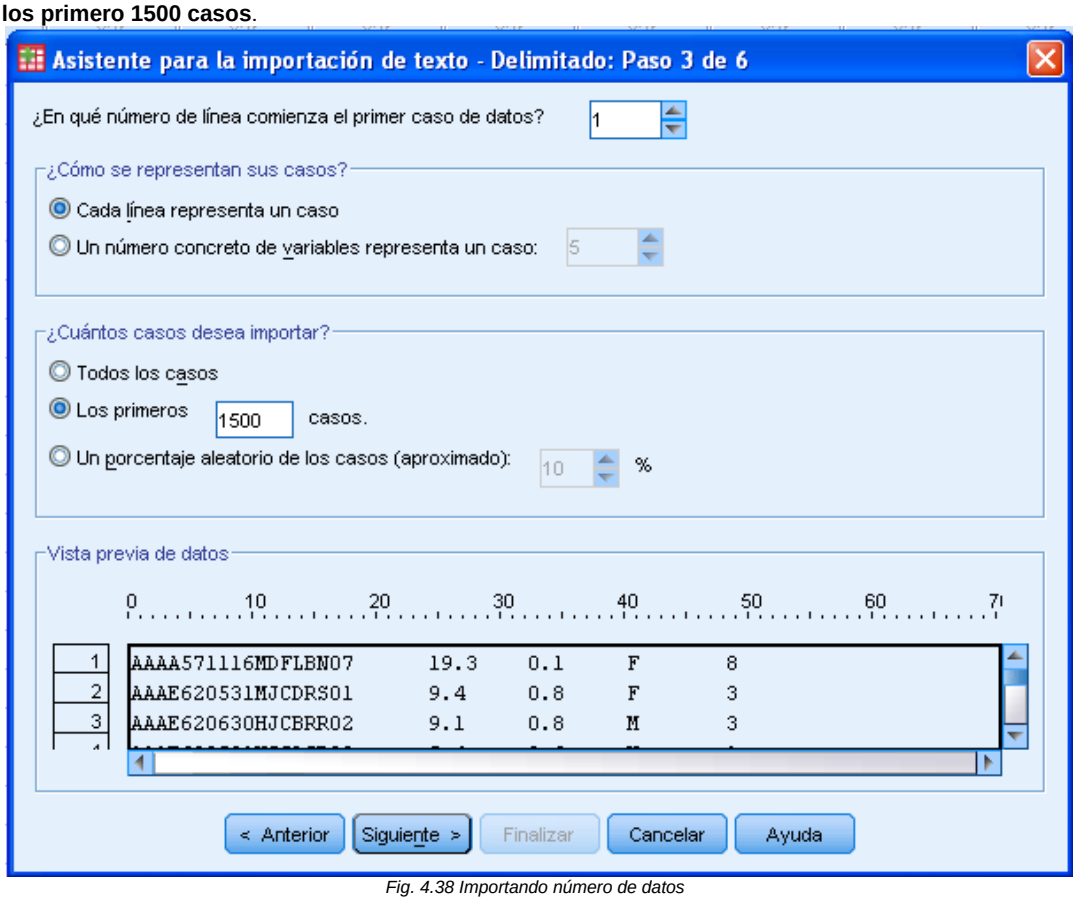


Fig. 4.38 Importando número de datos

Después seleccionamos **tabulador** como nuestro delimitador y **ningún calificador** de texto.

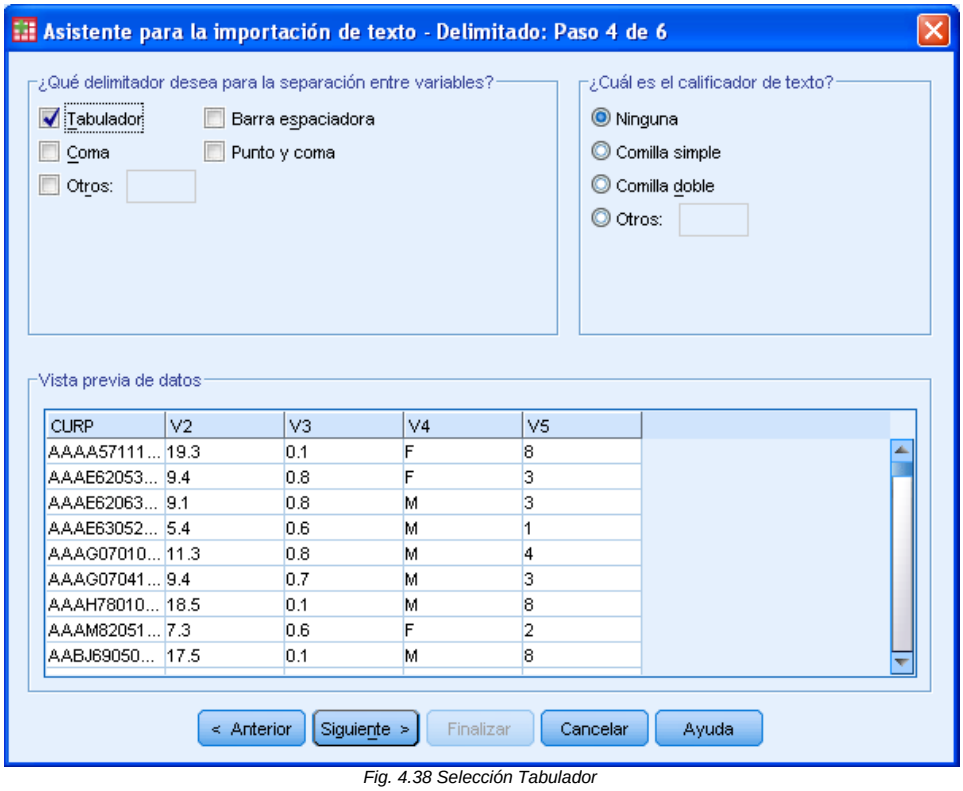


Fig. 4.38 Selección Tabulador

Ahora, **especificamos el formato** de datos para cada columna que compone nuestra vista minable, tal y como lo describimos en la tabla característica de atributos.

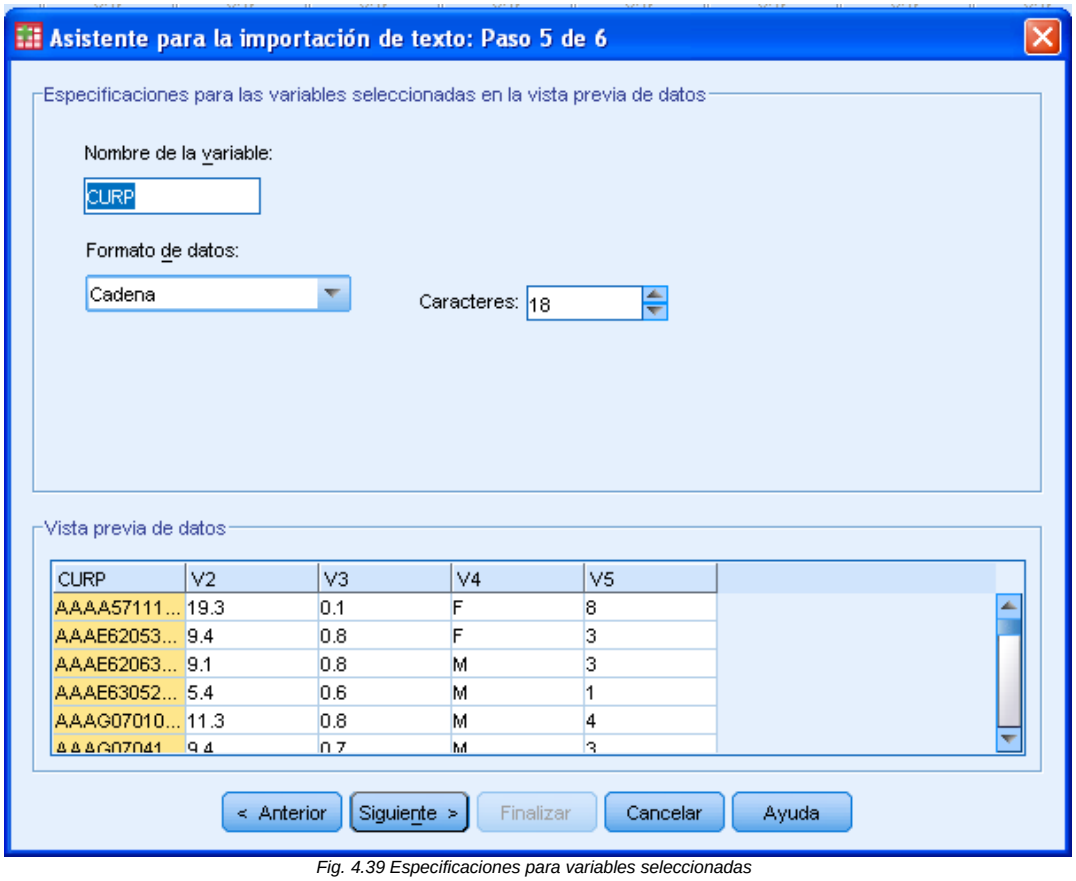


Fig. 4.39 Especificaciones para variables seleccionadas

Y en el último paso del asistente confirmamos y finalizamos:

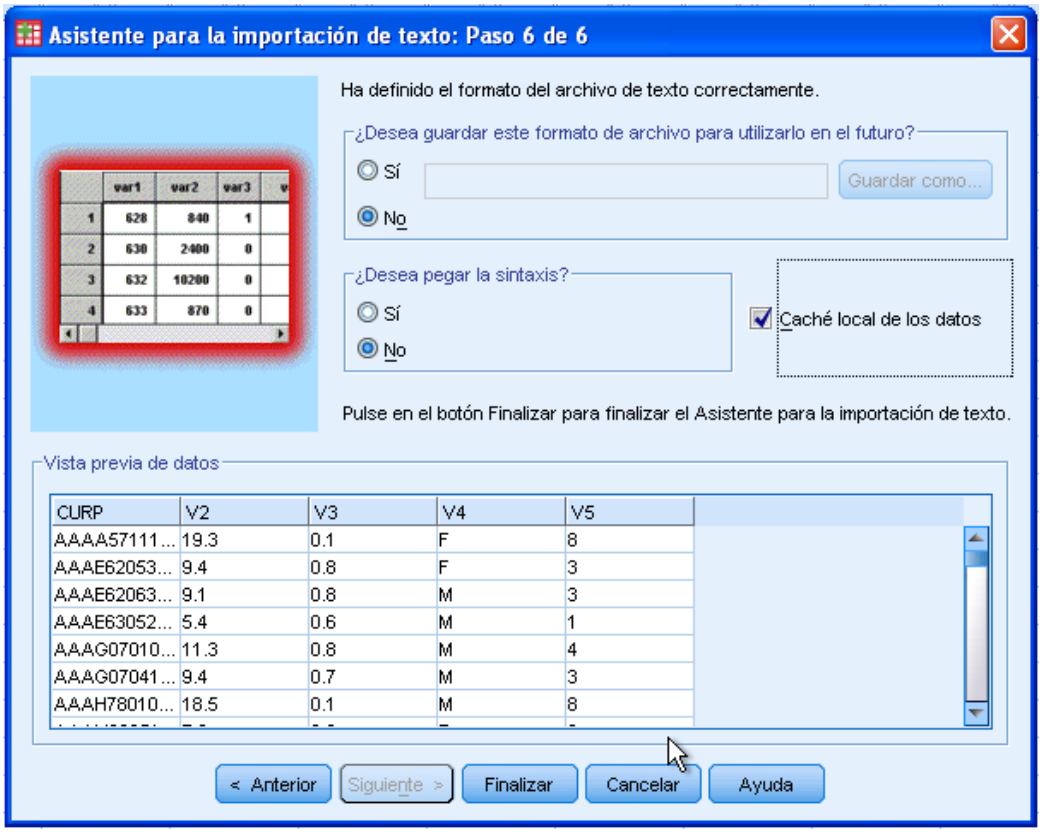


Fig. 4.40 Confirmación y finalización

Una vez hecho esto, nos mostrará la vista cargada correctamente, ahora procederemos a realizar el análisis, buscando una **correlación** entre nuestras variables peso y estatura.
Para esto en la barras de herramientas y en la pestaña **Analizar**, seleccionamos **correlaciones Bivariadas** Fig. 4.41:

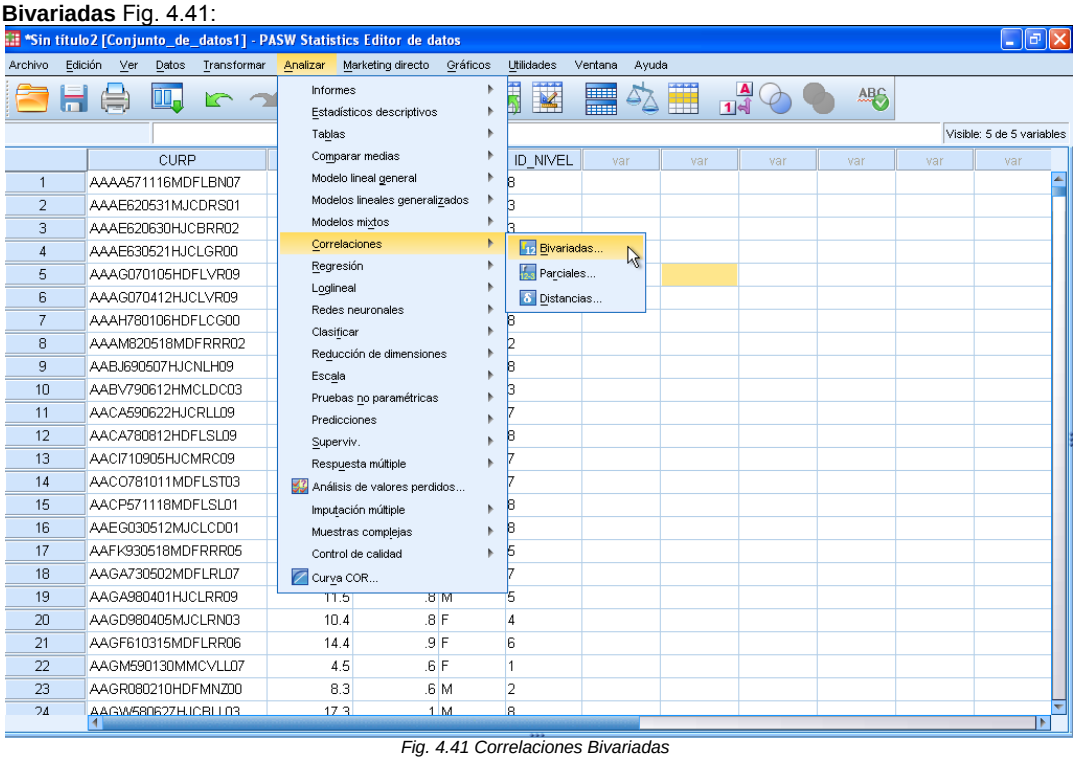


Fig. 4.41 Correlaciones Bivariadas

Nos muestra un ventana en la que seleccionamos la variable que queremos analizar, par este caso seleccionamos Peso y Estatura, y aceptamos.

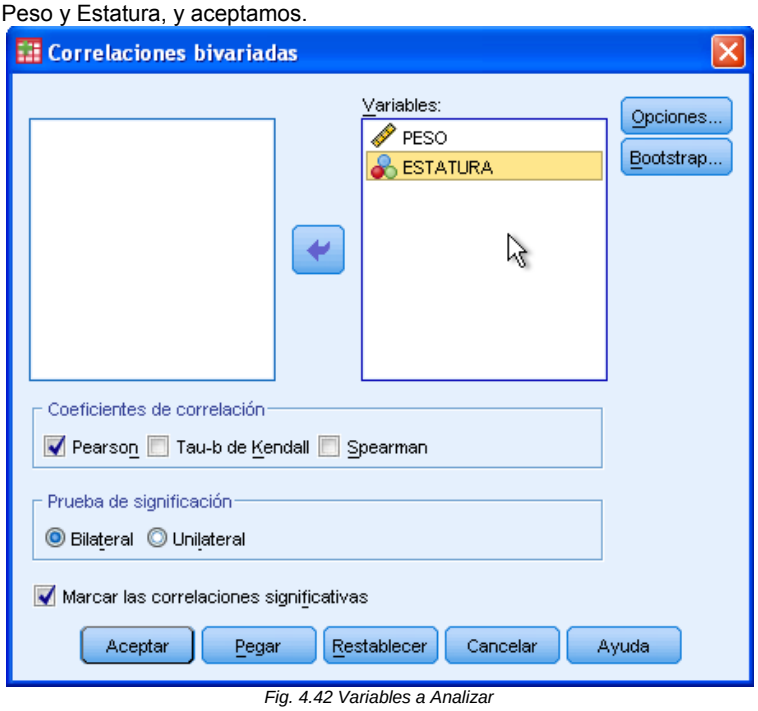


Fig. 4.42 Variables a Analizar

Una vez procesado el análisis nos muestra los resultados desglosado en una tabla como la siguiente:

Correlaciones

[Conjunto_de_datos1]

Correlaciones

		PESO	ESTATURA
PESO	Correlación de Pearson	1	0.805
	Sig. (bilateral)		.000
	N	1500	1500
ESTATURA	Correlación de Pearson	0.805	1
	Sig. (bilateral)	.000	
	N	1500	1500

**. La correlación es significativa al nivel 0,01 (bilateral).

Fig. 4.43 Correlaciones

Se trata de una matriz que correlaciona las variables que analizamos en este Peso y Estatura.
Como podemos observar se muestra una correlación de Pearson =**0.805**, entre peso y estatura, lo que nos indica que se trata de una correlación positiva.

Esto quiere decir que el Peso y Estatura del infante están estrechamente correlacionados, pues cuanto mayor Peso tenga el infante, mayor será su Estatura reportada mensualmente.

Conclusión

Sabiendo esto podemos ver que el peso y la estatura de los infantes están estrechamente relacionados y son factores determinantes en la salud de los infantes, por lo que proponemos crear campañas que verifique el crecimiento de cada infante, notificando a los padres o tutores por ejemplo su índice de Masa corporal y proponiendo dietas personalizadas que contribuyan al buen desarrollo de los infantes.

Además como información adicional, la Organización Mundial de la Salud estableció un índice para determinar cuándo se está excedido de peso o no. Se llama Índice de Masa Corporal. Es un cálculo que se puede realizar muy fácil. Aquí te presentamos una tabla que te ayudará mucho:

peso/(estatura*estatura).

30 o más: indica obesidad
25 a 29: indica sobrepeso
19 a 24: indica normal
Menos de 18: delgadez extrema

De esta manera determinaríamos la solución para cada caso y sería más factible realizarlo.

4.4 Evaluación de Modelos

En esta parte se abordara la última fase del proceso de extracción de conocimiento, evaluación, combinación, interpretación, difusión y uso de los modelos extraídos.
Las técnicas presentadas en esta parte permiten consolidar los modelos obtenidos en la fase anterior.

Técnicas de evaluación
Hemos presentado varias de las técnicas que permiten la generación de modelos a partir de una evidencia formada por un conjunto de datos.
Llegados a este punto, cabe preguntarse cómo podemos saber si un determinado modelo es lo suficientemente válido para nuestro propósito.

Los métodos de aprendizaje permiten construir modelos o hipótesis a partir de un conjunto de datos. En la mayoría de los casos es necesario evaluar la calidad de las hipótesis de la manera más exacta posible.
Por ejemplo, un error en la predicción de un modelo conlleva importantes consecuencias (por ejemplo detección de células cancerígenas), es importante conocer con exactitud el nivel de precisión de los modelos aprendidos.
Por lo tanto, la etapa de evaluación de modelos es crucial para la aplicación real de las técnicas de minería de datos.
Una opción es evaluar los modelos sobre un conjunto de datos diferente al conjunto de entrenamiento.
Otra aproximación, más realista por lo general, es la evaluación basada en costos
En este contexto, el mejor modelo es el modelo que comete errores con menor costo asociado, no el modelo que cometa menor número de errores.

Respecto a la primera opción, que consiste en separar el conjunto de datos que forma la evidencia en dos subconjuntos disjuntos como se muestra:

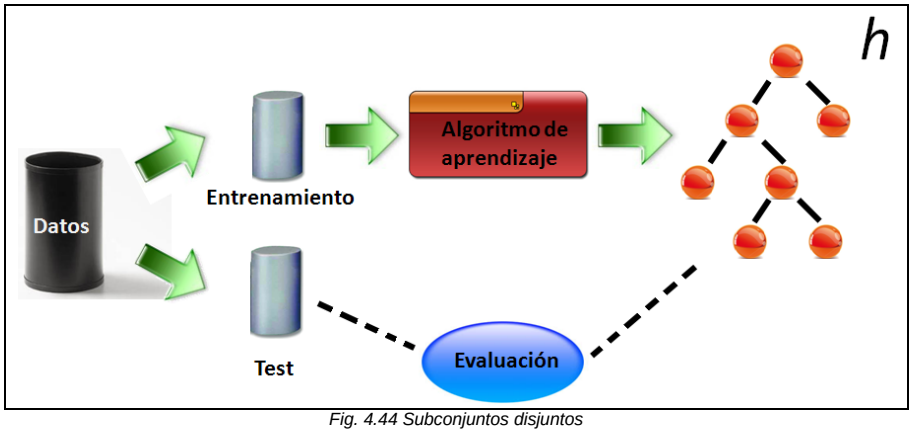


Fig. 4.44 Subconjuntos disjuntos

- El primer subconjunto, denominado de entrenamiento, se utiliza para el aprendizaje de las hipótesis.
- Al segundo subconjunto, denominado conjunto de prueba, sólo se emplea para calcular el error de muestra de las hipótesis construidas con los datos de entrenamiento.
Habitualmente, la partición se realiza de manera aleatoria, dejando el 50% de los datos para el entrenamiento y el 50% restante, para el subconjunto de prueba.

Evaluación mediante validación cruzada
Este método consiste en dividir el conjunto de la evidencia en "*k*" subconjuntos disjuntos de similar tamaño. Entonces, se genera una hipótesis utilizando el conjunto formado por la unión de *k-1* subconjuntos y el subconjunto restante se emplea para calcular un error de muestra parcial.
- Se repite "*k*" veces utilizando siempre un subconjunto diferente para estimar el error de muestra parcial.
- El error de muestra final se calcula como la media aritmética de los "k" errores de muestra parciales.
Comúnmente, se suelen utilizar 10 particiones evaluación mediante validación cruzada

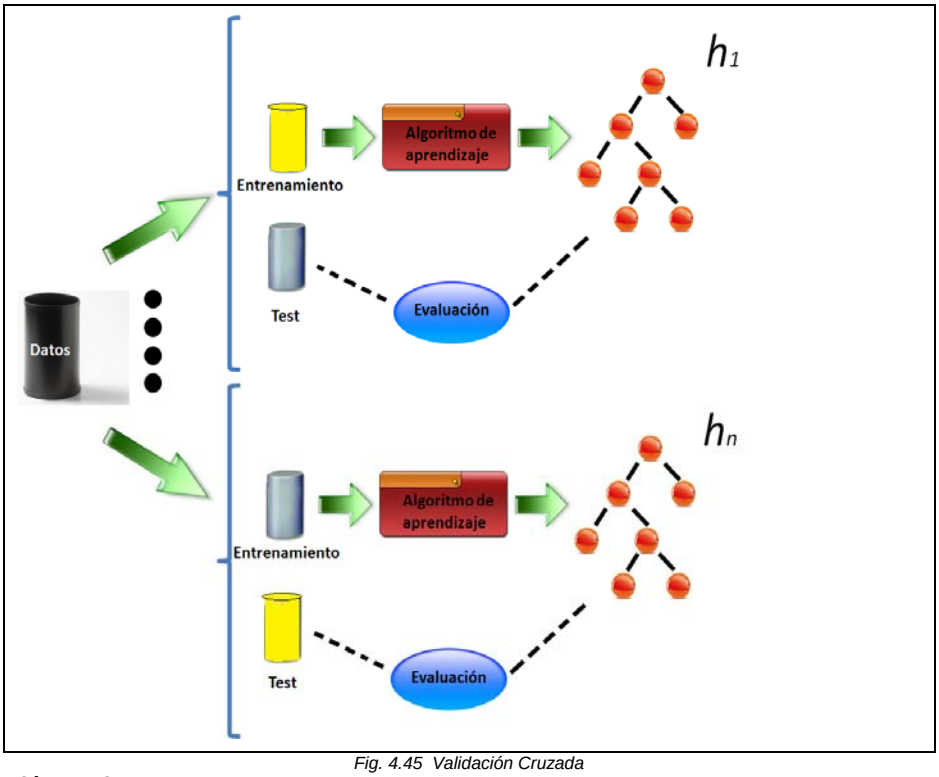


Fig. 4.45 Validación Cruzada

Evaluación por bootstrap

La evaluación por *"bootstrap"* se utiliza en casos en los que todavía se tienen menos ejemplos y está especialmente indicada en estos casos. La idea es en cierto modo similar a la "validación cruzada", aunque la forma de proceder es diferente.
Supongamos que tenemos "N" ejemplos. A partir de este conjunto realizamos un muestreo aleatorio. Esta muestra será el conjunto de entrenamiento.
Los ejemplos no elegidos por la muestra se reservan para el conjunto de test. Con estos conjuntos se entrena y evalúa un modelo.
El proceso anterior se repite un número de veces prefijado "k" y después se actúa como en el caso de la validación cruzada, promediando los errores.
Quizá lo interesante de este proceso es que las "k" repeticiones del proceso son independientes y esto es más robusto estadísticamente.

Evaluación de hipótesis basada en costo
Hemos presentado varias formas de evaluar hipótesis basadas en el porcentaje de error.
En muchos casos reales ocurre que diferentes errores tienen costos muy dispares.
El aprendizaje sensible al costo puede considerarse como una generalización más realista del aprendizaje predictivo.
En este contexto, la calidad de un determinado modelo se mide en términos de minimización de costos, en vez de en minimización de errores.

La Unidad de Cuidados Intensivos (UCI) de un hospital desea mejorar sus criterios de admisión, la probabilidad de cada uno de los casos es 0.5% ingreso a cuidado intensivo, 13% ingreso en observación, 86.5% No ingreso.
Desde el punto de vista de la precisión, podría considerarse como válido, sin embargo, este modelo sería inútil y muy peligroso, dado que se enviaría a casa pacientes en estado grave.
En este caso, sería más conveniente utilizar las técnicas de aprendizaje que obtienen modelos que minimizan los costos de aplicación. Considérese que se conocen, aproximadamente, los costos de cada clasificación errónea.

Matriz de costos
La manera más habitual de expresar estos costos en clasificación es mediante la denominada matriz de costos.
En esta matriz se expresan los costos de todas las posibles combinaciones que se pueden dar entre la clase estimada y la real.

	Real			
		Salida	**Observación**	**UCI**
Estimado	**Salida**	$0	$5,000	$500,000
	Observación	$300	$0	$50,000
	UCI	$800	$500	$0

Esta matriz se interpreta de la siguiente manera:
Si el sistema predice que un paciente debe ir a UCI, cuando en realidad no necesita tratamiento tiene un costo de 800 pesos. Nótese que todos los costos en el diagonal (es decir los aciertos) tiene valor 0.
Es bastante sencillo estimar el costo de un clasificador para un determinado conjunto de ejemplos si se dispone de la matriz de costos del problema. Para ello se utiliza la matriz de confusión.

Matriz de confusión
La matriz de confusión es una manera detallada de informar cómo distribuye los errores un determinado clasificador.
Por ejemplo, una posible matriz de confusión para una evidencia formada por 100 casos sería:

	Real			
		Salida	**Observación**	**UCI**
Estimado	**Salida**	71	3	1
	Observación	8	7	1
	UCI	4	2	3

Cada celda de la matriz de confusión enumera, para cada clase real, el número de casos en los cuales el clasificador ha predicho una clase.
A partir de una matriz de confusión se puede estimar directamente la precisión de un clasificador, dividiendo el número de acierto entre el número total de casos. Para este clasificador la precisión sería del 81% (71%+7%+3%).
Si se disponen de las matrices de costos y de confusión, el costo total (Costo =Σ Ci,j * Mi,j) que representan los posibles errores es de: $571600 (8x$300+4x$800 + 3x$5,000 + 2x$500 + 1x$500,000 + 1x$50,000).
En cambio, el costo del clasificador que siempre da el alta sería de $2560,000 con una precisión del 83%. (71 x$0 + 12x$5,000 + 5x$ 500,000).

En este ejemplo vemos que un clasificador con más error puede tener mucho menos costo.
Otra situación bastante frecuente en problemas de clasificación sobre Bases de Datos es que haya muchísima más proporción de algunas clases sobre otras.
Esta situación puede causar que la clase escasa se tome como ruido y sea ignorada.
Por ejemplo, si en un problema binario (si/no) sólo hay un 1 % de ejemplos de la clase no, la teoría "todo es de la clase sí" tendría un 99 por cierto de precisión.
Además suele ocurrir que clasificar un ejemplo de la clase minoritaria como la clase mayoritaria suele tener asociado costos más altos que la situación inversa.

Sobremuestreo
El sobremuestreo, consiste en duplicar ejemplos (tuplas) de las clases con menor proporción.
Esto evidentemente, cambia la proporción de las clases. Pero permite aprovechar a fondo los ejemplos de las clases más raras.
Debemos utilizar sobremuestreo cuando una clase es muy poco frecuente.

Submuestreo
Por otra parte, el submuestreo consigue efectos similares, pero en este caso filtrando los ejemplos (tuplas) de las clases con mayor proporción.

Evaluación de modelos de regresión
Cuando se desea evaluar la calidad de una hipótesis de regresión. Una de las medidas de evaluación más comúnmente usada se basa en sumar los errores cuadráticos.

Evaluación de modelos de agrupamiento
Este tipo de modelos son más difíciles de evaluar, ya que no existe una clase o un valor numérico donde medir las veces que el modelo aprendido predice correctamente.
Una opción es utilizar la distancia entre los grupos como medida de calidad del modelo de agrupamiento.
Cuanto mayor sea la distancia entre los grupos, significa que se ha efectuado una mejor separación, y por tanto mejor modelo.
Falta determinar qué se considera como distancia entre grupos; se han estudiado varias opciones y podemos destacar los siguientes:

- *Distancia media:* La distancia entre dos grupos se calcula como la media de las distancias entre todos los componentes de ambos grupos.

- *Distancia simple:* Se considera la distancia entre los vecinos más próximos de los dos grupos.
- *Distancia completa:* Similar a lo anterior, pero utilizando la distancia entre los componentes que se encuentren a más distancia.

La mejor evaluación de este tipo de modelos es saber si el modelo resultado de la fase de aprendizaje tiene un comportamiento útil cuando se utilice en su área de aplicación.

Evaluación de reglas de asociación
Para evaluar una regla de asociación, se suele trabajar con dos medidas para conocer la calidad de la regla: Cobertura y Confianza.

- La cobertura (también denominada soporte) de una regla se define como el número de instancias en las que la regla se puede aplicar.
- La confianza (también llamada precisión) mide el porcentaje de veces que la regla se cumple cuando se puede aplicar.

Otros criterios de evaluación
Existen otras aproximaciones que evalúan modelos basándose en criterios tales como:

- *Interés:* Intenta medir la capacidad del modelo de suscitar la atención del usuario del modelo.
Existen patrones evaluados que pueden no tener ningún interés.
Por ejemplo una regla de asociación como "si el paciente ingresa en maternidad entonces el sexo del paciente es mujer", es muy válida con las medidas vistas (soporte y precisión), pero no sirve de nada, porque no aporta nada nuevo e interesante.

- *Novedad:* Mide la capacidad de un modelo de sorprender al usuario con respecto al conocimiento previo que tenía sobre determinado problema.
- *Compresibilidad o Inteligibilidad.* Se divide en dos partes: Simplicidad y Aplicabilidad.
    - *Simplicidad:* Establecer el tamaño o complejidad del modelo.
    - Aplicabilidad: Capacidad de ser utilizado con éxito en el contexto real donde va ser aplicado.
- La comprensibilidad
La comprensibilidad del modelo es una cuestión mucho más subjetiva, ya que un modelo puede ser poco comprensible para un usuario y muy comprensible para otra. A pesar de ello, podemos encontrar algunas medidas útiles. Cuando el modelo se expresa como un conjunto de reglas, cuanto más cortas sean más comprensibles.
Esta medida puede calcularse contando las condiciones de las reglas y el número de reglas del modelo. Sin embargo, la longitud no es el único factor que influye en la comprensibilidad del modelo.
En general, cuando más alto es el nivel de abstracción mayor será la compresibilidad.
La discretización de atributos puede servir para obtener generalizaciones más comprensibles.
No obstante, tenemos que tener en cuenta que esta transformación de atributos continuos en discretos puede hacer que perdamos detalles importantes sobre los datos y que la precisión del modelo se vea afectada negativamente por ello.
Del mismo modo, y como se ha comentado, ciertos modelos no son comprensibles por el tipo de técnica utilizada, por ejemplo, las redes neuronales, aunque el resultado puede ser muy preciso.
Resumiendo, en muchos casos hay que alcanzar un consenso entre comprensibilidad y precisión.

Criterios más significativos
La correcta elección del método de evaluación es un paso determinante del éxito final de todo el proceso de minería de datos. Una evaluación realizada de manera incorrecta puede provocar que seleccionemos un modelo que presente bajas prestaciones en el entorno real de aplicación. Hemos de considerar las características propias de cada problema, adaptando la evaluación a esas características.

Ejemplos de evaluación

Ejemplo 1: Árboles de decisión

Utilizando el ***método mediante validación cruzada*:**

Primero dividimos nuestro conjunto de datos, en este caso será nuestra vista minable, en 10 subconjuntos compuestos por 68 instancias cada uno de los 9 subconjuntos y el último por 70 instancias para nuestro ejemplo.

Formamos una hipótesis formada por los 9 subconjuntos de 68 instancias, y el subconjunto de 70 instancias, lo emplearemos para un error de la muestra parcial.
Las respuestas que dieron estos nuevos infantes a la encuesta fueron:

Para esto utilizamos Weka, como lo hicimos en nuestro ejemplo para cada subconjunto
Tenemos:

$$\text{Error}_{\text{parcial1}} = \mathbf{1.3325}$$

Repetimos 10 veces utilizando siempre el subconjunto diferente para estimar el error de la muestra parcial.

A continuación se muestra la tabla que muestra los errores parciales:

Muestras	Errores Parciales
1	1.3325
2	1.0022
3	1.5325
4	1.8705
5	1.9001
6	0.9925
7	1.0322
8	1.8375
9	1.0897
10	1.0984

Sabiendo que el error de muestra final se calcula como la media aritmética de los 10 errores de muestra parciales.

Tenemos que el $\text{Error}_{\text{MuestraFinal}}$ = **1.36881**

Una vez que tenemos nuestro árbol de decisión, pasaremos a la tarea de clasificar algunos infantes de nuestra estancia que no han sido reportados en nuestro OLAP.

La muestra de estos nuevos infantes fue:

curp_infante	consumo	nombre	nivel	desnutrición
ROCC041219MJCSSR06	CM	PREESCOLAR	6	no
UIUL060616MDFRRZ00	NC	MATERNAL	4	no
VACG050913HJCSRL00	CM	PREESCOLAR	6	no
GOBM060412HJCNNR09	NC	MATERNAL	4	yes
HECY041019MJCRBN00	PR	PREESCOLAR	6	yes
BAEM030203HJCRSR09	NC	PREESCOLAR	8	yes
SACB040413MJCNRL07	CM	PREESCOLAR	7	no
CACP060404MJCLBL01	PR	MATERNAL	5	yes
MACL040904MDFTSC01	NC	PREESCOLAR	6	yes

VAOI041103HDFLLV00	CM	PREESCOLAR	6	no

Ahora aplicamos el árbol de decisión en donde el nivel de infantes más propenso, a padecer una desnutrición es:

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

consumo = CM: no (181.0/3.0)
consumo = NC
|   nivel <= 4: no (14.0)
|   nivel > 4: yes (150.0/1.0)
consumo = PR
|   nivel <= 4: no (12.0)
|   nivel > 4: yes (156.0/4.0)
consumo = CT: no (179.0/2.0)

Number of Leaves  :         6

Size of the tree :  9
```

Como podemos observar los niveles más propensos son >=4.

Conclusión

Apoyados en nuestro Árbol de decisión podríamos determinar la edad en la que los infantes están más expuestos a presentar algún problema nutricional y podríamos proponer a la estancia organizar una serie de campañas en las que se fomente el desarrollo nutricional de los infantes, teniendo mayor énfasis en dichos niveles y de esta manera disminuir el porcentaje de desnutrición en cada estancia infantil.

Ejemplo 2: Árboles de decisión

Utilizando el ***método mediante validación cruzada*:**

Primero dividimos nuestro conjunto de datos, en este caso será nuestra vista minable, en 10 subconjuntos compuestos por 60 instancias cada uno de los 9 subconjuntos y el último por 62 instancias par nuestro ejemplo.

Formamos una hipótesis formada por los 9 subconjuntos de 60 instancias, y el subconjunto de 62 instancias, lo emplearemos para un error de la muestra parcial.
Para esto utilizamos Weka, como lo hicimos en nuestro ejemplo para cada subconjunto
Tenemos:

$$Error_{parcial1} = 17.5684$$

Repetimos 10 veces utilizando siempre el subconjunto diferente para estimar el error de la muestra parcial.

A continuación se muestra la tabla que muestra los errores parciales:

Muestras	Errores Parciales
1	20.0024
2	19.1734
3	16.9733
4	17.4722
5	16.0155
6	19.6566
7	17.6784
8	20.7845
9	19.0967
10	17.5553

Sabiendo que el error de muestra final se calcula como la media aritmética de los 10 errores de muestra parciales.

Tenemos que el $Error_{MuestraFinal}$ = **18.44083**

Analizando el árbol obtenido ahora para una nueva muestra de datos tenemos lo siguiente:

Descripción	nombre	consumo	nivel_nombre	nivel
jugos_aguas	Tamarindo_A	CT	MATERNAL	6
jugos_aguas	Tuna_A	CT	PREESCOLAR	6
jugos_aguas	Uva_J	CT	PREESCOLAR	6
Sopas	Calabaza	CT	MATERNAL	3
Sopas	Pasta	CT	MATERNAL	3
Sopas	Calabaza	CT	PREESCOLAR	7
Sopas	Verduras	CT	PREESCOLAR	6
Sopas	Zanahoria	CT	PREESCOLAR	6
variaciones_leche	Arroz	CT	LACTANTE	2
variaciones_leche	Atole	CT	LACTANTE	2
variaciones_leche	Avena	CT	MATERNAL	3
variaciones_leche	Cajeta	CT	MATERNAL	3

En donde el resultado fue:

descripcion = jugos_aguas
| nivel_nombre = MATERNAL
| | nivel <= 4: Tamarindo (3.0/1.0)
| nivel_nombre = PREESCOLAR:
| | nivel <= 6:Tuna (3.0/1.0)
| | nivel > 6: Tuna (3.0/1.0
descripcion = sopas
| nivel <= 7
| | nivel <= 3: Calabaza (5.0/1.0)
| | nivel > 2: Pasta (5.0/1.0)
| nivel > 7: Calabaza (5.0/1.0)
| | nivel > 2: Verduras (5.0/1.0)
| nivel > 7: zanahorias(5.0/1.0)
descripcion = variaciones_leche
| nivel_nombre = LACTANTE
| | nivel <= 2: **Arroz (14.0/2.0)**
| | nivel > 2: Atole (13.0/7.0)
| nivel_nombre = MATERNAL
| | nivel <= 3
| | | nivel <= 1:Avena(3.0)
| | | nivel > 1: Cajeta (2.0)

Siguiendo el árbol generado podemos determinas que alimento prefieren los infantes según su nivel.

Conclusión:

Como podemos observar conforme crecen los infantes se acentúan más y se define el gusto por ciertos alimentos, por lo cual debemos desarrollar el hábito de llevar una dieta adecuada que cubra todos los nutrientes necesarios a los infantes, pues esto será la base para su desarrollo.
Es ahora cuando podemos influir en el gusto de los infantes por ciertos alimentos como las verduras motivándolos, sin obligarlos pues esto genera una actitud negativa, de esta manera disminuiremos el porcentaje de enfermedades en los infantes.

Ejemplo 3: Árboles de decisión

Evaluación

Una vez que tenemos nuestro árbol de decisión, pasaremos a la tarea de clasificar algunos infantes de nuestra estancia que no han sido revisados por los médicos.

Las respuestas que dieron estos nuevos infantes a la encuesta fueron:

curp_infante	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
AUPV080405HDFGDC09	B	B	A	B	A	A	A	A	A	A	B	A	A	A	A	B	B	B	A	B	A
AAMG080305HDFLRB09	A	B	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
RIGA080512HDFVNR04	A	A	A	B	A	A	A	A	B	B	B	A	A	A	A	B	B	A	A	B	A
AYVE080409HDFZYM01	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
GEMC080316MMCRRR03	B	A	B	B	B	B	A	B	B	B	B	B	A	A	A	B	B	A	A	B	B
NECM080511MDFRRO04	A	A	B	A	B	B	B	A	B	B	A	B	A	A	A	A	A	A	A	A	A
AESM080318MMCSNY05	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	A	B	B
ROPL080523MDFXZC06	A	A	B	A	B	B	B	A	B	B	A	A	A	A	A	A	A	A	A	A	A
AAGR080210HDFMNZ00	B	B	A	B	A	A	A	A	A	A	B	A	A	A	A	B	B	B	A	B	A
MAHE080319HDFRRR01	A	A	B	A	B	B	B	B	B	B	A	B	B	B	B	A	A	A	B	A	B
PAVA080319HDFLLR02	A	A	A	A	A	A	B	A	B	B	A	B	A	A	A	A	A	A	A	A	A
BUAM080322HDFNLG06	B	B	A	A	A	A	A	A	A	A	A	A	B	B	B	A	A	B	A	A	A
EICG080512MMCLHR04	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
MORG080407MDFNDB03	B	A	A	B	A	A	A	A	B	B	B	B	A	A	A	B	B	A	A	B	A
MAVD080421MDFLLL01	A	B	A	A	A	A	A	A	A	A	A	A	B	B	B	A	A	B	A	A	A
VAPI080506MDFZRR02	A	B	A	A	A	A	B	A	B	B	A	B	A	A	A	A	A	A	A	A	A
NASL080625MDFVTT05	A	A	B	A	B	B	B	A	B	B	A	B	A	A	A	A	A	A	B	A	A
DEHJ070822HMCLRS08	B	B	A	B	A	A	A	A	B	B	B	A	B	B	B	B	B	B	A	B	B
MACE070704HMCRHN09	A	B	A	B	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
OOIJ070801HMCRNN01	B	B	A	B	A	A	A	A	A	A	B	A	A	A	A	B	B	B	A	B	A

Una vez que aplicamos el árbol de decisión llegamos al diagnóstico de cada infante el cual fue:

curp_infante	diagnóstico
AUPV080405HDFGDC09	Peso Normal
AAMG080305HDFLRB09	Sobrepeso
RIGA080512HDFVNR04	Peso normal
AYVE080409HDFZYM01	Sobrepeso
GEMC080316MMCRRR03	Bajo peso
NECM080511MDFRRO04	Obesidad
AESM080318MMCSNY05	Obesidad
ROPL080523MDFXZC06	Obesidad
AAGR080210HDFMNZ00	Obesidad
MAHE080319HDFRRR01	Peso normal
PAVA080319HDFLLR02	Sobrepeso
BUAM080322HDFNLG06	Bajo peso
EICG080512MMCLHR04	Sobrepeso
MORG080407MDFNDB03	Bajo peso
MAVD080421MDFLLL01	Bajo peso
VAPI080506MDFZRR02	Sobrepeso
NASL080625MDFVTT05	Sobrepeso
DEHJ070822HMCLRS08	Peso normal
MACE070704HMCRHN09	Peso normal
OOIJ070801HMCRNN01	Peso normal

Conclusión

Apoyados con este árbol de decisión, podemos ayudar a que los padres y los médicos detecten de una manera fácil y con alto índice de confiabilidad a los infantes que sean propensos a presentar problemas de sobrepeso, ayudando a que sean tratados lo más pronto posible para evitar más alteraciones de la salud.

Evaluación ejemplo 4: Clustering

Apoyándonos en los resultados obtenidos, se decidió aplicar tareas de limpieza integral en el área de cocina de la estancia, además de una campaña de información sobre higiene enfocada a padres e hijos. Posteriormente al desarrollo de estas dos actividades, la estancia siguió recabando datos hasta tener 712 registros nuevos (un 33.33% de los usados en el experimento original) a los que nuevamente se les realizara un proceso de clustering para ver si han surgido cambios a los resultados anteriores. Cabe destacar que estos datos fueron recabados durante la época invernal.
Siguiendo los pasos vistos con anterioridad:

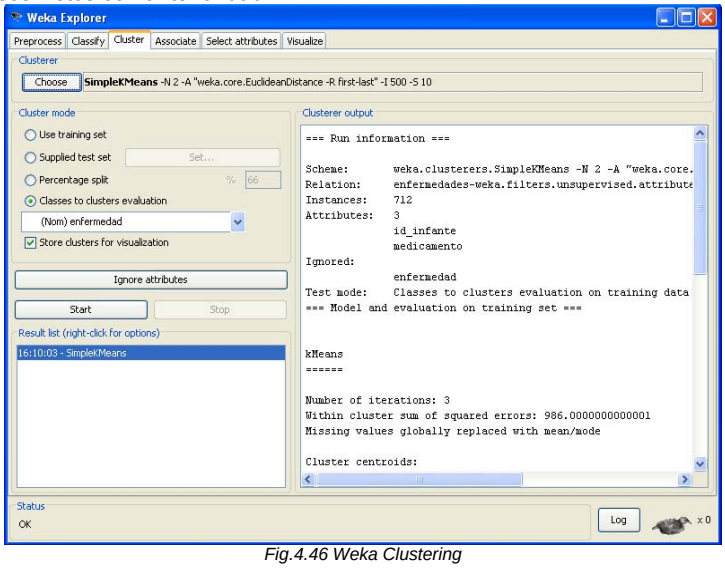


Fig.4.46 Weka Clustering

Se obtuvieron los siguientes resultados:

```
=== Run information ===
Scheme:       weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     enfermedades-weka.filters.unsupervised.attribute.Remove-R1-2
Instances:    712
Attributes:   3
              id_infante
              medicamento
Test mode:    Classes to clusters evaluation on training data
=== Model and evaluation on training set ===
kMeans
======
Number of iterations: 3
Within cluster sum of squared errors: 986.0000000000001
Missing values globally replaced with mean/mode

Cluster centroids:
                                Cluster#
Attribute       Full Data          0          1
                  (712)        (496)      (216)
==============================================
id_infante          A1184         A5        A56
medicamento           A33        A33        A34

Clustered Instances

0      496 ( 70%)
1      216 ( 30%)

Class attribute: enfermedad
Classes to Clusters:

  0   1    <-- assigned to cluster
 85   0  | ESTREÑIMIENTO
127   1  | AMIGDALITIS AGUDA
 22   0  | DIARREA AGUDA
 44   0  | DOLOR DE CABEZA (CEFALEA)
218 215 | GRIPE

Cluster 0 <-- AMIGDALITIS AGUDA
Cluster 1 <-- GRIPE

Incorrectly clustered instances :    370.0          51.9663 %
```

Ejemplo 5 Reglas de Asociación

Realizamos nuevamente el estudio de algunos expedientes médicos de infantes y se muestra a continuación una parte de la vista minable utilizada para esta carga incremental

Nivel	CURP	Nombre_enfermedad	Parentesco	Id_enfermeda	Sano
LACTANTE	HEPJ980605HDFRXL08	OTOSCLEROSIS	Abuelo materno	26	S
PREESCOLAR	RERP941017MDFVYM04	CARDIOVASCULAR	Abuela paterna	23	S
PREESCOLAR	RERP941017MDFVYM04	CARDIOVASCULAR	Abuela paterna	23	S
PREESCOLAR	TOPG950201HDFREB01	CARDIOVASCULAR	Abuela materna	23	S
PREESCOLAR	TOPG950201HDFREB01	CARDIOVASCULAR	Abuela materna	23	S
PREESCOLAR	ZAMM940720HDFMXN03	FALCIFORME	Abuelo materno	27	S

Una vez procesada esta muestra el resultado fue:

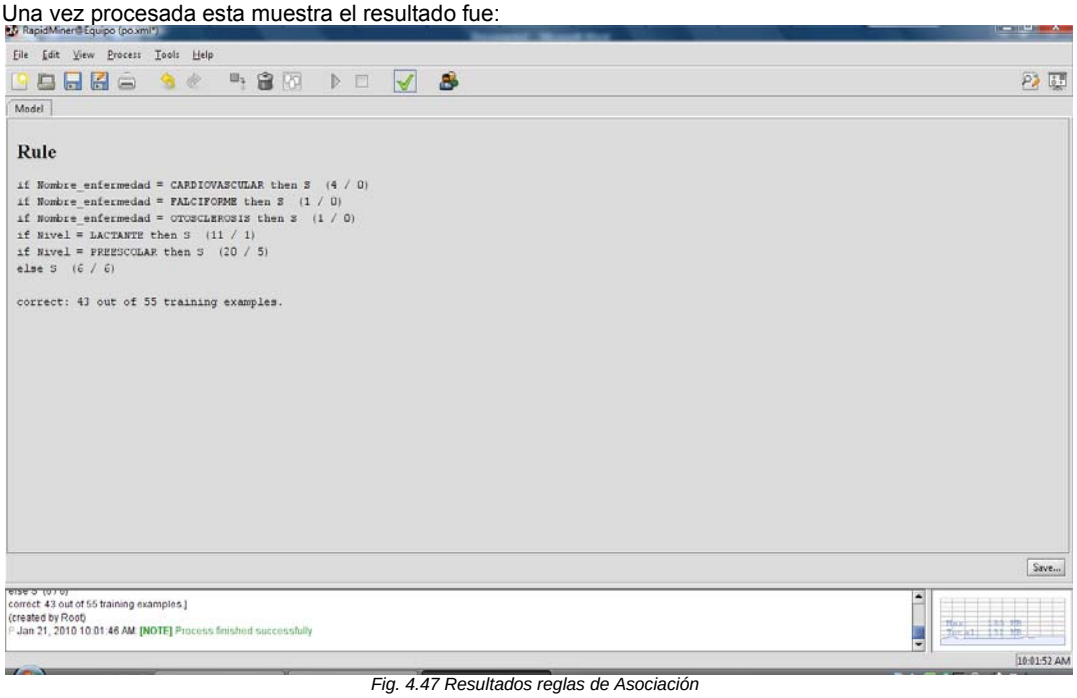


Fig. 4.47 Resultados reglas de Asociación

Conclusión:
Apoyados en este método podemos identificar de manera más clara a lo infantes que puedan presentar alguna enfermedad hereditaria y contar con los materiales necesarios en caso de un incidente.

Ejemplo 6: Correlación.

Evaluación
Una vez que tenemos matriz de correlación, continuaremos a la tarea de clasificar algunos infantes de nuestra estancia que no han sido registrados en nuestra OLAP.

La tabla de registro que llenaron quedó así:

CURP	PESO	ESTATURA	GENERO	ID_NIVEL
COGM041010HDFRRN01	7.3	.6	M	2
COGN681115MDFRMN08	18.3	1.1	F	8
COHA680802HDFRRR00	18.4	1.1	M	8
COHA830120HDFBRR07	5.5	.6	M	1
COHL781012MJCMRC01	19.5	1.1	F	8
COHM680802MDFRRR00	18.5	1.1	F	8
COHS940320HDFLRG09	16.3	1.1	M	7
COHS950620HJCLRR03	16.5	1.1	M	7
COLM940220HJCOOR03	19.3	1.1	M	8
COLM951210HDFOOR02	12.3	.8	M	5
COLS780923MDFSYR04	17.5	1.1	F	8
COMA790110HDFNEN04	19.3	1.1	M	8
COMA821222HMCRNL00	11.4	.8	M	4
COML690930MDFRRZ05	18.3	1.1	F	8
COMP930511HDFRJB05	4.5	.6	M	1
COMZ620324MDFSNR00	4.4	.6	F	1
COPA600802MDFRLL09	18.5	1.1	F	8

Utilizando **Statistical Package for the Social Sciences (SPSS)**, procedemos como lo hicimos anteriormente buscando una **correlación** entre nuestras variables Consumo y Estatura.
El análisis nos muestra el siguiente resultado:

Correlaciones

[Conjunto_de_datos1]

Correlaciones

		PESO	ESTATURA
PESO	Correlación de Pearson	1	.799**
	Sig. (bilateral)		.000
	N	1500	1500
ESTATURA	Correlación de Pearson	.799**	1
	Sig. (bilateral)	.000	
	N	1500	1500

**. La correlación es significativa al nivel 0,01 (bilateral).

Fig. 4.48 Correlaciones

Conclusión

Tomando como base la matriz de correlación que obtuvimos, podemos observar que la nuevamente la correlación fue positiva, lo que nos índica que el peso y estatura están estrechamente relacionados y por supuesto son un factor determinante en la salud de los infantes, por lo que podemos proponer campañas en la estancia que fomenten una cultura de chequeo periódico, en la que se revisé cada infante y se pueda tratar para evitar que adquieran enfermedades como desnutrición o tensión arterial alta, y difundir esta cultura tanto dentro como fuera de la estancia, para de esta manera disminuir el número de casos de infantes que padecen alguna enfermedad debito a su bajo o alto índice de masa corporal.

CONCLUSIÓN CAPITULO 4:

Es uno de los capítulos más interesantes y útiles pues, se trata de la fase más importante en el proceso de extracción de conocimiento ya que nos permite no sólo describir y explicar los sucesos específicos en nuestro sistema sino que además nos permite predecir hechos, que a la hora de la toma de decisiones son muy importantes de considerar.

También nos dimos cuenta que cierta información que creímos que solo servía como datos históricos podría darnos respuestas a preguntas que ni siquiera habíamos planteado.

Conclusiones Generales y Recomendaciones

Este trabajo nos ayudo a reforzar nuestros conocimientos poniendo en práctica la teoría vista en las asignaturas: base de datos, base de datos avanzadas, depósitos de datos y minería de datos.

Esperando que este trabajo sirva de guía o referencia rápida a los alumnos de próximas inscripciones en el modulo de Base de Datos, y para sus proyectos de cada asignatura, ya que le material de apoyo para dichas asignaturas es muy escaso, y esperamos que dicho material les facilite a los alumnos la comprensión de cada uno de los temas involucrados en esta tesis.

Y por ultimo cumplimos el objetivo primordial de nuestra tesis el cual le la el titulo a la misma **“Enlazar los programas de estudio por medio de una aplicación de 4 asignaturas del Módulo de Bases de Datos”.**

BIBLIOGRAFÍA Y MESOGRAFÍA

- **Case method**
  Richard Barker
  Editorial Addison-Wesley

- **Introducción a los sistemas de base de datos**
  C.J. Date
  Editorial Addison-Wesley

- **Oracle manual del administrador**
  Kevin Loney
  Editorial Oracle Press

- **Oracle Data Warehousing**
  Michael J. Corey , Michael Abbey
  Editorial Oracle Press

- **Introducción a la minería de datos**
  José Hernández Orarlo, M. José Ramírez Quintana, César Ferri Ramírez
  Editorial Prentice Hall

- **Tutorial básico de minería de datos**
  http://msdn.microsoft.com/es-es/library/ms167167.aspx

- **Tutorial pentaho**
  http://pentaho.almacen-datos.com/