VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA
DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES
"ARAGÓN"

"Business Intelligence con SQL Server 2005"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACION

P R E S E N T A:

CHRISTIAN EDUARDO BARRADAS VILLEGAS

ASESOR: MTI. Omar Mendoza González

MÉXICO 2008

INTRODUCCIÓN

CAPÍTULO I DIPLOMADO DE ADMINISTRACIÓN DE BASES DE DATOS

1.1 Módulo I SISTEMAS DE INFORMACIÓN Y EL MODELADO DE DATOS RELACIONAL

- 1.1.1. Sistemas de Información
- 1.1.2. Componentes del sistema de Información
- 1.1.3. Modelo de Datos
- 1.1.4. Modelo jerárquico
- 1.1.5. Modelo de red
- 1.1.6. Modelo relacional
- 1.1.7. Modelo orientado a objetos
- 1.1.8. Reglas de Codd
- 1.1.9 Diagrama Entidad – Relación
- 1.1.10 Entidades
- 1.1.11 Atributos
- 1.1.12 Dominios
- 1.1.13 Claves
- 1.1.14 Interrelaciones
- 1.1.15 Restricciones en las interrelaciones
- 1.1.16 Normalización
- 1.1.17 Reglas de integridad

1.2 Módulo II SISTEMAS MANEJADORES DE BASES DE DATOS RELACIONALES (SGBD)

- 1.2.1 SGBD
- 1.2.2 Función del SGBD
- 1.2.3 Arquitectura Cliente/Servidor
- 1.2.4 Esquema de seguridad en el SGBD
- 1.2.5 Componentes de un SGBD
- 1.2.6 DDL o Lenguaje de Definición de Datos
- 1.2.7 DML o Lenguaje de Manipulación de Datos
- 1.2.8 DCL o Lenguaje de Control de Datos
- 1.2.9 DD o Diccionario de Datos
- 1.2.10 Principales Funciones del DD
- 1.2.11 Arquitectura
- 1.2.12 Usuarios en una Base de Datos
- 1.2.13 Transact-SQL

1.3 Módulo III SQL (Structured Query Language)

- 1.3.1 Tipos de datos del sistema
- 1.3.2 Tablas
- 1.3.3 Creación de Tablas
- 1.3.4 Modificación de Tablas
- 1.3.5 Llaves e Índices
- 1.3.6 Creación de índices
- 1.3.7 Manipulación de datos
- 1.3.8 Join
- 1.3.9 Cláusula SELECT
- 1.3.10 Cláusula FROM
- 1.3.11 Cláusula WHERE

CAPÍTULO II SQL SERVER 2005 BUSINESS INTELLIGENT

2.1 SQL SERVER INTEGRATION SERVICES

- 2.1.1 Usos típicos de Integration Services
- 2.1.2 Limpiar y normalizar datos
- 2.1.3 Generar Business Intelligence en un proceso de transformación de datos
- 2.1.4 Automatizar las funciones administrativas y la carga de datos
- 2.1.5 Arquitectura de Integration Services
- 2.1.6 Servicio Integration Services
- 2.1.7 Modelo de objetos de Integration Services
- 2.1.8 Tiempo de ejecución de Integration Services
- 2.1.9 Flujo de datos de Integration Services

2.2 SQL SERVER ANALYSIS SERVICES

- 2.2.1 Arquitectura de Analysis Services
- 2.2.2 Arquitectura de servidor (Analysis Services)
- 2.2.3 Arquitectura de cliente (Analysis Services)
- 2.2.4 Conceptos de Analysis Services
- 2.2.5 Objetos de Analysis Services

2.3 SQL SERVER REPORTING SERVICES

- 2.3.1 Presentación de Reporting Services
- 2.3.2 Información general de componentes de Reporting Services
- 2.3.3 Diagrama de la arquitectura de Reporting Services
- 2.3.4 Implementar Reporting Services
- 2.3.5 Planear una implementación de Reporting Services
- 2.3.6 Implementación estándar
- 2.3.7 Implementación escalada
- 2.3.8 Implementación en un clúster con equilibrio de carga de red (NLB)
- 2.3.9 Implementación en un clúster de conmutación por error de SQL Server
- 2.3.10 Diagrama de implementación escalada
- 2.3.11 Métodos para crear un informe
- 2.3.12 Crear informes con el generador de informes
- 2.3.13 Crear informes con el diseñador de informes

CONCLUSIONES

BIBLIOGRAFÍA

INTRODUCCIÓN

La Inteligencia de Negocios o Business Intelligence (BI) se puede definir como el proceso de analizar los bienes informáticos o datos acumulados en una empresa y extraer una cierta inteligencia o conocimiento de ellos. Dentro de la categoría de bienes se incluyen las bases de datos de clientes, información de la cadena de suministro, ventas personales y cualquier actividad de marketing o fuente de información relevante para la empresa.

BI apoya a los tomadores de decisiones proporcionándoles información correcta, en el momento y lugar correcto, lo que les permite tomar mejores decisiones de negocios. La información adecuada en el lugar y momento adecuado incrementa efectividad de cualquier empresa.

La tecnología de BI no es nueva, ha estado presente de varias formas por lo menos en los últimos 20 años, comenzando por generadores de reportes y sistemas de información ejecutiva en los 80's…". Entiéndase como sinónimos de tecnología de BI los términos aplicaciones, soluciones o software de inteligencia de negocios.

Tal vez ayude a comprender mejor el concepto por medio de un ejemplo. Una franquicia de hoteles a nivel nacional que utiliza aplicaciones de BI para llevar un registro estadístico del porcentaje promedio de ocupación del hotel, así como los días promedio de estancia de cada huésped, considerando las diferencias entre temporadas. Con esta información ellos pueden:

- Calcular la rentabilidad de cada hotel en cada temporada del año
- Determinar cual es su segmento de mercado
- Calcular la participación de mercado de la franquicia y de cada hotel
- Identificar oportunidades y amenazas

Estas son sólo algunas de las formas en que una empresa u organización se puede beneficiar por la implementación de software de BI, hay una gran variedad de aplicaciones o software que brindan a la empresa la habilidad de analizar de una forma rápida por qué pasan las cosas y enfocarse a patrones y amenazas.

CAPÍTULO I
DIPLOMADO DE ADMINISTRACIÓN DE BASES DE DATOS

1.1 MÓDULO I SISTEMAS DE INFORMACIÓN Y EL MODELADO DE DATOS RELACIONAL

1.1.1 Sistemas de Información

Un sistema de información se define como un conjunto de procedimientos interrelacionados que forman un todo, es decir, obtiene, procesa, almacena y distribuye información (datos manipulados) para apoyar la toma de decisiones y el control en una organización.

Un sistema de información contiene información de sus procesos y su entorno. Como actividades básicas produce la información que se necesita: entrada, procesamiento y salida. La retroalimentación consiste en entradas devueltas para ser evaluadas y perfeccionadas. Proporciona la información necesaria a la organización o empresa, donde y cuando se necesita.

Tipos: Transaccionales, de apoyo a las decisiones y estratégicos.
Sistema de Información: es el conjunto de procesos que, operando sobre una colección de datos estructurada de acuerdo a una empresa, recopila, elabora y distribuye (parte de) la información necesaria para la información de dicha empresa y para las actividades de dirección y control correspondientes, apoyando al menos en parte, la toma de decisiones necesarias para desempeñar las funciones y procesos de negocios de la empresa de acuerdo a su estrategia.

1.1.2 Componentes del sistema de Información

- Herramientas tecnológicas (hardware, software).
- Procedimientos.
- El recurso humano que interactúa con el sistema de información, el cual esta formado por las personas que utilizan el sistema, también conocidos como usuarios.

Un sistema de información realiza cuatro actividades básicas: entrada, almacenamiento, procesamiento y salida de información.

Los sistemas de Información que logran la automatización de procesos operativos dentro de una organización, son llamados frecuentemente sistemas transaccionales, ya que su función primordial consiste en procesar transacciones tales como pagos, cobros, pólizas, entradas, salidas, etc. Por otra parte, los sistemas de Información que apoyan el proceso de toma de decisiones son los sistemas de Soporte a la toma de decisiones, sistemas para la toma de decisión de Grupo, sistemas expertos de soporte a la toma de Decisiones y sistema de Información para ejecutivos. El tercer tipo de sistema, de acuerdo con su uso u objetivos que cumplen, es el de los sistemas estratégico, los cuales se desarrollan en las organizaciones con el fin de lograr ventajas competitivas, a través del uso de la tecnología de información.

1.1.3 Modelo de Datos

Modelo: Es una representación de la realidad que contiene las características generales de algo que se va a realizar. En base de datos, esta representación la elaboramos de forma gráfica.

Modelo de datos: Es una colección de herramientas conceptuales para describir los datos, las relaciones que existen entre ellos, semántica asociada a los datos y restricciones de consistencia.

Los cuatro modelos de datos más ampliamente aceptados son:

- Modelo jerárquico.
- Modelo de red.
- Modelo relacional.
- Modelo orientado a objetos

1.1.4 Modelo jerárquico

Es similar al modelo de red en cuanto a las relaciones y datos, ya que estos se representan por medio de registros y sus ligas. La diferencia radica en que están organizados por conjuntos de árboles en lugar de gráficas arbitrarias.

En este tipo de modelos la organización se establece en forma de árbol, donde la raíz es un nodo ficticio. Así tenemos que, una base de datos jerárquica es una colección de árboles de este tipo.

1.1.5 Modelo de red

Este modelo representa los datos mediante colecciones de registros, sus relaciones se representan por medio de ligas o enlaces, los cuales pueden verse como punteros.
Una base de datos de red como su nombre lo índica, esta formado por una colección de registros, los cuales están conectados entre sí por medio de enlaces.

1.1.6 Modelo relacional

El trabajo publicado por Codd en ACM (1970) presentaba un nuevo modelo de datos que perseguía una serie de objetivos, que se resumen en las siguientes líneas:

Independencia física. El modo en el que se almacenan los datos no influye en su manipulación lógica y por tanto, los usuarios que acceden a esos datos no tienen que modificar sus programas por cambios en el almacenamiento físico.

Independencia lógica. El añadir, eliminar o modificar objetos de la base de datos no repercute en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).

Flexibilidad. En el sentido de poder presentar a cada usuario los datos de la forma en que éste prefiera.

Uniformidad. Las estructuras lógicas de los datos presentan un aspecto uniforme, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.

Sencillez. Las características anteriores, así como unos lenguajes de usuario muy sencillos, producen como resultado que el modelo de datos relacional sea fácil de comprender y de utilizar por parte del usuario final.

El modelo Relacional se divide en 3 partes:

- Estructura de los datos.
- Integridad de los datos.
- Manipulación de los datos.

1.1.7 Modelo orientado a objetos

El modelo de datos orientado a objetos es una adaptación a los sistemas de bases de datos. Se basa en el concepto de encapsulamiento de datos y código que opera sobre estos en un objeto. Los objetos estructurados se agrupan en clases.

El propósito de los sistemas de bases de datos es la gestión de grandes cantidades de información. Las primeras bases de datos surgieron del desarrollo de los sistemas de gestión de archivos. Estos sistemas primero evolucionaron en bases de datos de red o en bases de datos jerárquicas y, más tarde, en bases de datos relacionales.

La interfaz entre un objeto y el resto del sistema se define mediante un conjunto de mensajes. Un método, es un trozo de código para implementar cada mensaje. Un método devuelve un valor como respuesta al mensaje.

1.1.8 Reglas de Codd

1. Cualquier BDMS que proclame ser relacional, deberá manejar, completamente, las bases de datos por medio de sus capacidades relacionales.

2. Regla de información. Toda la información dentro de una base de datos relacional se representa de manera explícita a nivel lógico y exactamente de una sola manera, como valores en una tabla.

3. Regla del acceso garantizado. Se garantiza que todos y cada uno de los datos (valor atómico) en una base de datos relacional pueden ser leídos recurriendo a una combinación del nombre de la tabla, valor de la llave primaria y nombre de la columna.

4. El manejo sistemático de los valores nulos. En un SGBD totalmente relacional se soportan los valores nulos (que son distintos de una cadena de caracteres vacía o de una cadena con caracteres en blanco o de cero o cualquier otro número), para representar información faltante o no aplicable de una forma consistente, independientemente del tipo de dato.

5. Catálogo dinámico en línea basado en un modelo relacional. La descripción de la base de datos se representa en el nivel lógico de la misma forma que los datos ordinarios, de tal suerte que los usuarios autorizados puedan aplicar el mismo lenguaje relacional para consultarla, que aquél que emplean para con sus datos habituales.

6. Regla del sub-lenguaje de dato completo. Se debe contar con un sub-lenguaje que contemple la definición de datos, la definición de vistas, la manipulación de datos, las restricciones de integridad, la autorización, el inicio y fin de una transacción.

7. Regla de actualización de vistas. Todas las vistas que teóricamente sean actualizables deberán ser actualizadas por medio del sistema.

8. Inserción, actualización y eliminación de alto nivel. La posibilidad de manejar una relación base o una relación derivada como un sólo operador se aplica a la lectura, inserción, modificación y eliminación de datos.

9. Independencia física de los datos. Los programas de aplicación y la actividad en terminales no deberán ser afectados por cambios en el almacenamiento físico de los datos o en el método de acceso.

10. Independencia lógica de los datos. Los programas de aplicación y la actividad en terminales no deberán ser afectados por cambios de cualquier tipo que preserven la información y que teóricamente permitan la no afectación en las tablas base.

11. Independencia de la integridad. Las restricciones de integridad de una base de datos deberán poder definirse en el mismo sub-lenguaje de datos relacional y deberán almacenarse en el catálogo, no en los programas de aplicación.

12. Independencia de la distribución. Un SGBD relacional tiene independencia de distribución.

13. Regla de la no subversión. Si un sistema relacional tiene un lenguaje de bajo nivel (un sólo registro cada vez), ese bajo nivel no puede ser utilizado para suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

1.1.9 Diagrama Entidad – Relación

El modelo Entidad - Relación, es una técnica de diseño de bases de datos gráfica, que incorpora información relativa a los datos y la relación existente entre ellos, para poder así plasmar una visión del mundo real sobre un soporte informático y que se caracteriza fundamentalmente por:

- Sólo reflejar la existencia de los datos sin expresar lo que se hace con ellos.
- La independencia de la base de datos y de los sistemas operativos.
- La inclusión de todos los datos sin considerar las aplicaciones que se tendrán.

1.1.10 Entidades

Se puede definir como entidad a cualquier objeto, real o abstracto, que existe en un contexto determinado o que puede llegar a existir y del cual deseamos guardar información, por ejemplo, un profesor, un alumno o bien una materia. Las entidades las podemos clasificar en:

Fuertes: Son aquellas entidades que existen por sí mismas, es decir, la existencia de un ejemplar de la entidad no depende de la existencia de otros ejemplares en otra entidad, por ejemplo, la entidad "PROFESOR".

Débiles: Son aquellas entidades en las que su existencia depende de la existencia de ejemplares en otras entidades, por ejemplo, la existencia de la entidad "*PROFESOR*" depende de la existencia de la entidad "*ESCUELA*".

1.1.11 Atributos

Las entidades se componen de atributos que son cada una de las propiedades o características que tienen las entidades. Cada ejemplar de una misma entidad posee los mismos atributos, tanto en nombre como en número, diferenciándose cada uno de los ejemplares por los valores que toman dichos atributos. Si consideramos la entidad "PROFESOR" y definimos los atributos Nombre, Cursos, Teléfonos y Edad, podríamos obtener los siguientes ejemplares:

PROFESOR			
NOMBRE	**CURSOS**	**TELÉFONOS**	**EDAD**
Luís García	Algebra	555-55-55	80.500
Juan Antonio Álvarez	Diseño	666-66-66	92.479
Marta López	Base de Datos	777-77-77	85.396

Existen cuatro tipos de atributos:

1. *Obligatorios:* Aquellos que forzosamente deben tomar un valor.
2. *Opcional*: Aquellos atributos que pueden o no tener valores.
3. *Monoevaluado*: Aquel atributo que sólo puede tener un único valor.
4. *Multievaluado*: Aquellos atributos que pueden tener varios valores.

Dentro del diagrama, la entidad "*PROFESOR*" y sus atributos quedarían de la siguiente forma:

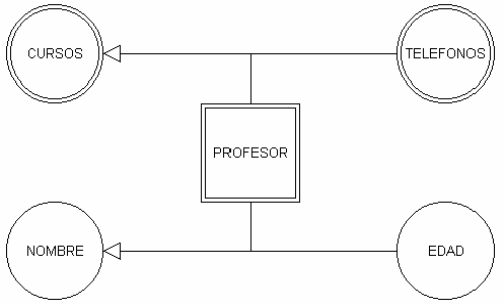

Figura 1.0. Representación de entidades y atributos

Existen atributos, llamados derivados, cuyo valor se obtiene a partir de los valores de otros atributos, por ejemplo, supongamos que la entidad "*PROFESOR*" tiene los atributos "*NOMBRE*", "*FECHA DE NACIMIENTO y* "*EDAD*"; el atributo "*EDAD*" es un atributo derivado por que se calcula a partir del valor del atributo "*FECHA DE NACIMIENTO*". Su representación gráfica es la siguiente:

Figura 1.1. Atributo derivado

En determinadas ocasiones es necesaria la descomposición de un atributo para definirlos en más de un dominio, podría ser el caso del atributo "*TELEFONO*" que toma valores del dominio "*PREFIJO*" y del dominio "NUMERO". Estos atributos se representan de la siguiente forma:

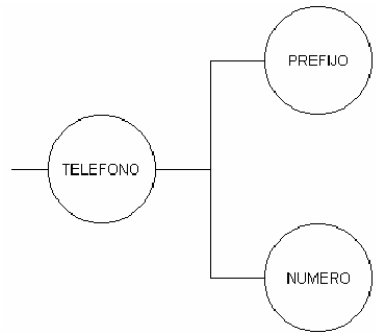

Figura 1.2. Descomposición de un atributo

1.1.12 Dominios

Se define dominio como un conjunto de valores que puede tomar un determinado atributo dentro de una entidad. Por ejemplo:

Atributo	Dominio
Fecha de Alta	Calendario Gregoriano
Teléfono	Conjunto de números de teléfonos
Cobro de Incentivos	SI / NO
Edad	16 - 65

De forma casi inherente al término dominio aparece el concepto restricción para un atributo. Cada atributo puede adoptar una serie de valores de un dominio restringiendo determinados valores. El atributo "*EDAD*" toma sus valores del dominio N (números naturales) pero se puede poner como restricción aquellos que estén en el intervalo (0-120), pero dentro de la entidad "*PROFESOR*" se podría restringir aun más el intervalo, puesto que la edad mínima para trabajar es de 16 años y la máxima de 65, por lo tanto el intervalo sería (16-65).

1.1.13 Claves

El Diagrama Entidad - Relación exige que cada entidad tenga un identificador, se trata de un atributo o conjunto de atributos que identifican de forma única a cada uno de los ejemplares de la entidad.

Un ejemplo de identificador es el atributo "RFC", que en la entidad "PROFESOR", identifica de forma única a cada uno de los profesores. Estos identificadores reciben el nombre de Clave Primaria o Primary Key (PK). Como ya se había mencionado antes, puede ser que existan más identificadores, a estos atributos se les conoce como Identificadores Candidatos (IC).

Los atributos identificadores de una entidad se representan en los diagramas de la siguiente forma:

Figura 1.3. Atributos identificadores.

1.1.14 Interrelaciones

Se entiende por interrelación a la asociación, vinculación o correspondencia entre entidades. Por ejemplo, entre la entidad "*PROFESOR*" y la entidad "*CURSO*" podemos establecer la relación "*IMPARTE*" por que el profesor imparte cursos.

Al igual que las entidades, las interrelaciones se pueden clasificar en regulares y débiles, esto de acuerdo al tipo de entidad que estén asociando, entidades regulares o entidades débiles, con otra de cualquier tipo. Las interrelaciones débiles se subdividen en dos grupos:

- *En existencia*: Cuando los ejemplares de la entidad débil no pueden existir si desaparece el ejemplar de la entidad regular del cual dependen.
- *En identificación*: Cuando además de ser una relación en existencia, los ejemplares de la entidad débil no se pueden identificar por sí mismos y exigen añadir el identificador principal de la entidad regular del cual dependen para ser identificados.

Las interrelaciones, dentro de los diagramas, se representan de la siguiente forma:

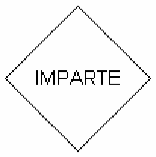

Figura 1.4. Interrelación regular.

Figura 1.5. Interrelacion débil.

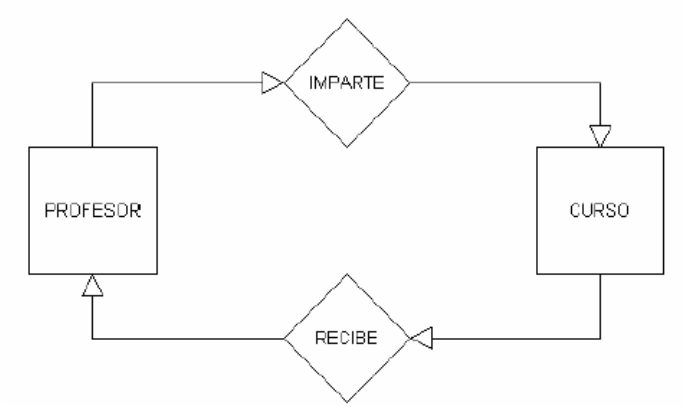

Figura 1.6. Interrelación regilar con entidades.

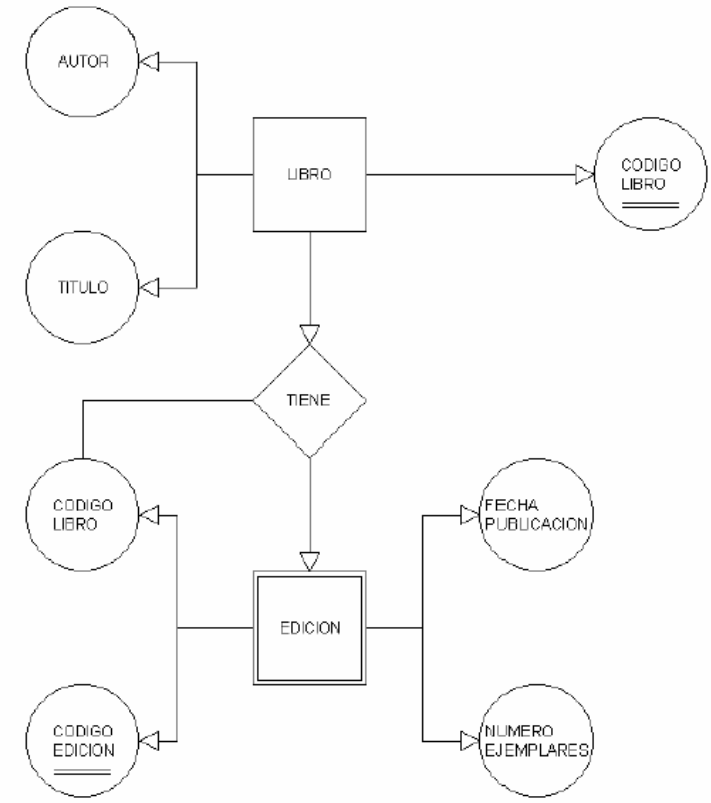

Figura 1.7. Interrelación en identidad

Figura 1.8. Interrelación en existencia

En cada interrelación se debe establecer el número máximo y mínimo de ejemplares de un tipo de entidad que pueden estar asociadas, mediante una determinada relación con un ejemplar de la otra entidad. Este valor máximo y mínimo se conoce como *cardinalidad* y según corresponda, se representa de la siguiente forma: (0,N), (N,0), (1,N), (N,1), (0,1), (1,0) ó (N,N). La cardinalidad se representa de la siguiente forma:

Figura 1.9. Representación de la cardinalidad.

En el diagrama anterior la cardinalidad "*CLIENTE*" - "*PEDIDO*" es 1:1 por que al formularnos la pregunta ¿cuántos clientes se pueden relacionar con un pedido? la respuesta es, uno como mínimo y uno como máximo, ya que un pedido es realizado por un único cliente y no cabe la posibilidad que el mismo pedido esté formulado por dos clientes distintos. La cardinalidad "*PEDIDO*" - "*CLIENTE*" es 1:N por que al formularnos la pregunta ¿cuántos pedidos se pueden relacionar con un cliente? la respuesta es, como mínimo un pedido pertenece a un cliente, pero varios pedidos pueden estar relacionados con el mismo cliente.

Existen ocasiones concretas en que las relaciones tienen atributos, es el caso del diagrama siguiente en donde los alumnos reciben cursos, y la interrelación posee los atributos de fecha de comienzo, fecha de finalización y calificación.

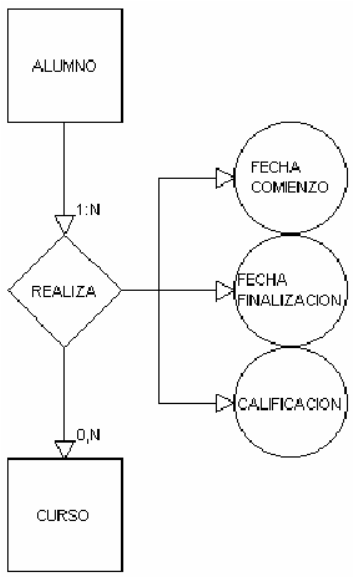

Figura 1.10. Relaciones con atributos.

A medida que se van estableciendo las interrelaciones hay que prestar especial atención a las interrelaciones cíclicas o redundantes, que son aquellas cuya eliminación no implica la pérdida de información. Pongamos como ejemplo en siguiente modelo entidad - relación:

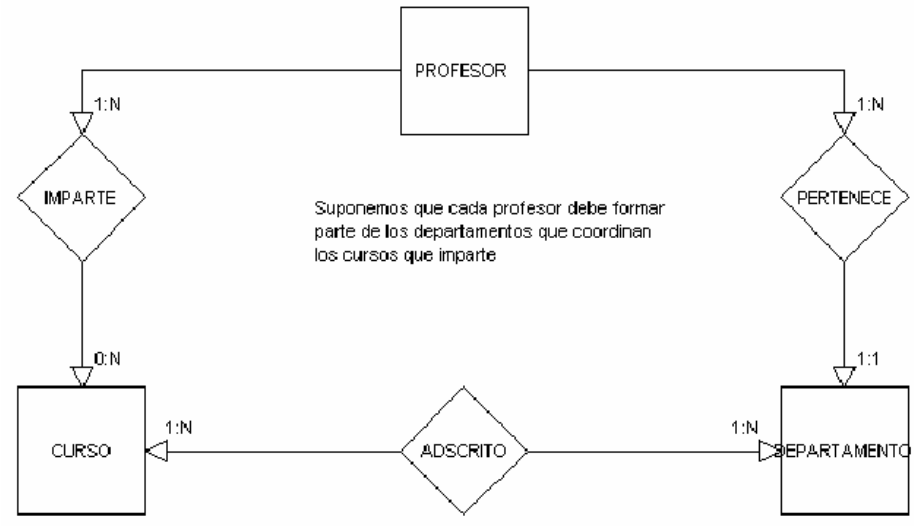

Figura 1.11. Interrelaciones cíclicas o redundantes.

Según se plantea el esquema la relación "*PERTECE*" se puede suprimir por que para saber a qué departamentos pertenece un profesor basta con saber que cursos imparte y conociendo los cursos averiguamos que departamentos están asociados a los cursos. En este caso se dice que: "*PERTENECE*" = "*IMPARTE*" + "*ADSCRITO*".

1.1.15 Restricciones en las interrelaciones

Restricción de Exclusividad: Esta restricción se da cuando cada ejemplar de la entidad presente sólo puede combinarse con ejemplares de una sola de las entidades restantes. Por ejemplo:

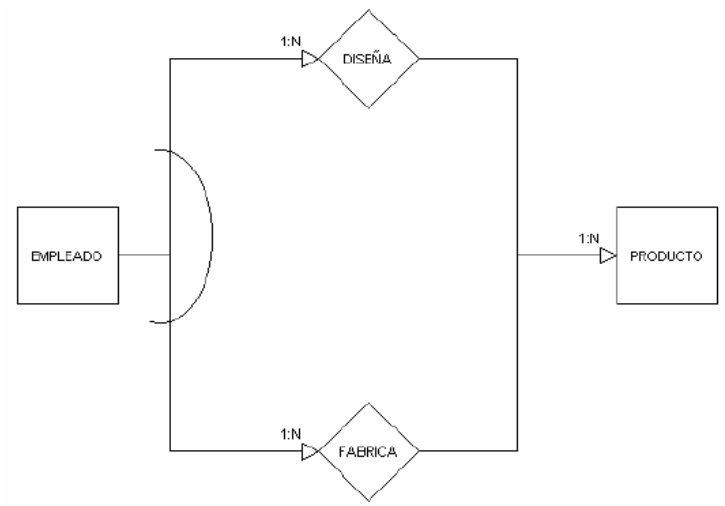

Figura 1.12. Restricción de exclusividad

Los empleados, en función de sus capacidades, o son diseñadores de productos o son operarios y los fabrican, no es posible que ningún empleado sea diseñador y fabricante a la misma vez.

Restricción de Exclusión: Se produce una restricción de exclusión cuando los ejemplares de las entidades sólo pueden combinarse utilizando una interrelación, este es el caso del siguiente ejemplo:

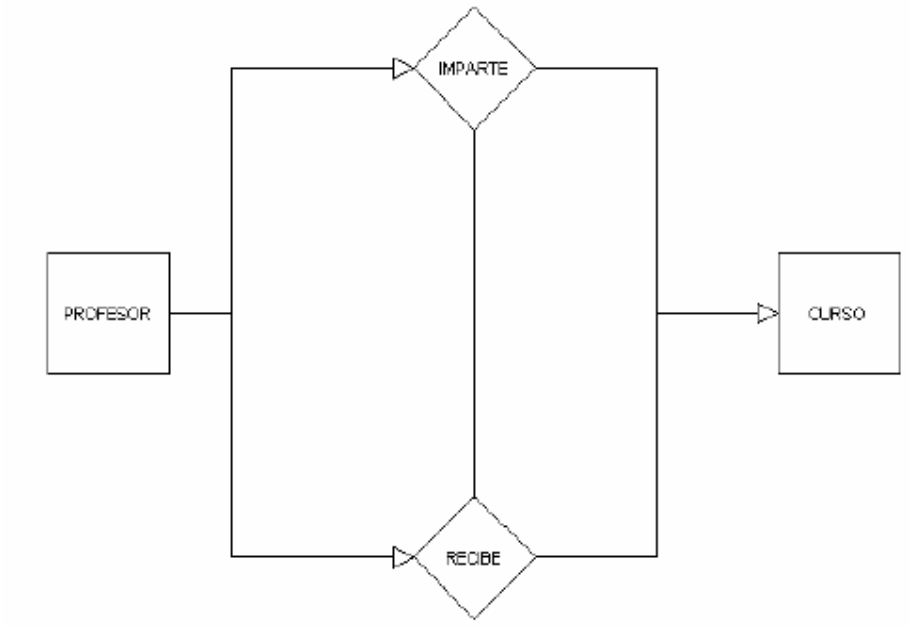

Figura 1.13. Restricción de exclusión.

Un profesor no puede recibir e impartir el mismo curso.

Restricción de Inclusividad: Se dice que una relación es de inclusividad cuando todo ejemplar de una entidad participa en ambas interrelaciones, por ejemplo:

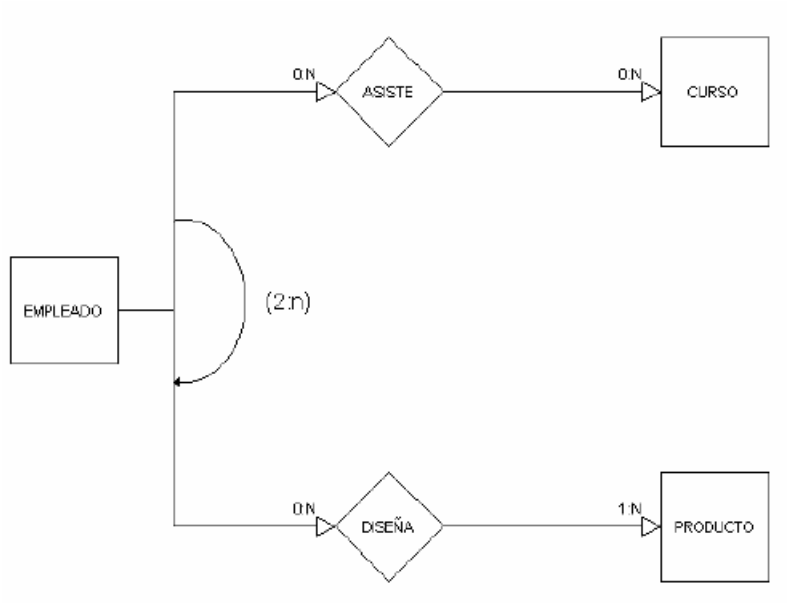

Figura 1.14. Restricción de inclusividad.

Para que un empleado pueda trabajar como diseñador de productos deber haber asistido, al menos, a dos cursos.

Restricción de Inclusión: Se establece cuando la asociación entre entidades esta condicionada con la existencia de una segunda interrelación, por ejemplo:

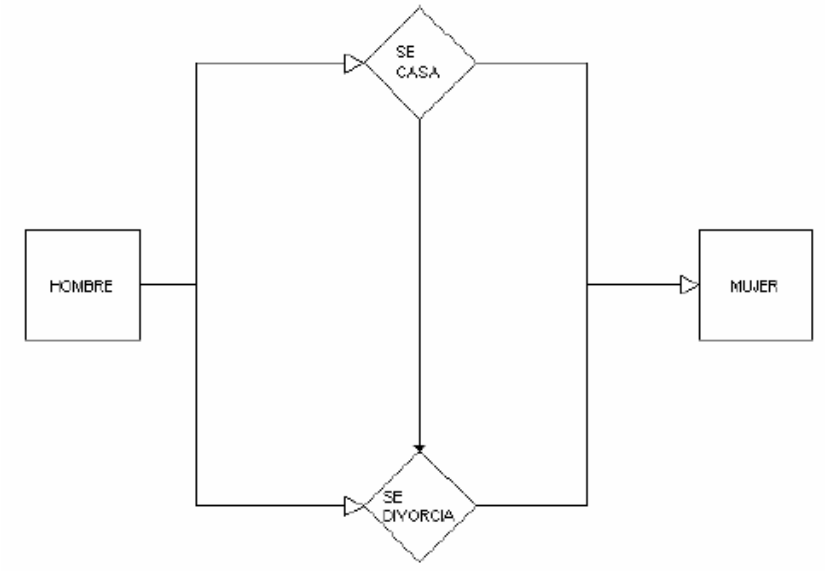

Figura 1.15. Restricción de inclusión.

Para que un hombre se divorcie de una mujer deben estar casados

1.1.16 Normalización

El proceso de cristalización de las entidades y sus relaciones en formatos de tabla usando los conceptos relacionales se llama proceso de normalización y consiste en agrupar a los campos de datos en un conjunto de relaciones o tablas que representan a las entidades, sus características y sus relaciones de forma adecuada.

La razón de la normalización es asegurar que el modelo conceptual de la base de datos funcionará. Esto no significa que una estructura no normalizada no funcionará, sino que puede causar algunos problemas cuando los programadores de aplicación traten de modificar la base de datos para insertar, actualizar o eliminar datos.

Las ventajas de la normalización son las siguientes:

- Evita anomalías en inserciones, modificaciones y borrados.
- Mejora la independencia de datos.
- No establece restricciones artificiales en la estructura de los datos.

Uno de los conceptos fundamentales en la normalización es el de dependencia funcional.

La dependencia funcional es una noción semántica donde cada dependencia es una clase especial de regla de integridad y representa una relación de uno a muchos.

La normalización se lleva a cabo en una serie pasos, cada paso corresponde
a una forma normal que tiene ciertas propiedades. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto y más fuerte y por lo tanto, son menos vulnerables a las anomalías de actualización. Las tres principales formas normales son las siguientes:

Primera Forma Normal (1FN): Una relación está en primera forma normal si, y sólo si, todos los dominios de la misma contienen valores atómicos, es decir, no hay grupos repetitivos. Si se ve la relación gráficamente como una tabla, estará en 1FN si tiene un solo valor en la intersección de cada fila con cada columna y tiene una clave primaria.

Segunda Forma Normal (2FN): Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no está en la clave primaria es completamente dependiente de la clave primaria.

Tercera Forma Normal (3FN): Una relación está en tercera forma normal si, y sólo si, está en 2FN y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria. La dependencia es transitiva si existen las dependencias siendo atributos o conjuntos de atributos de una misma relación.

Básicamente, las reglas de normalización están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.

1.1.17 Reglas de Integridad

Una vez definida la estructura de datos del modelo relacional, pasamos a estudiar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina *restricciones de dominios*. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados o instancias (las reglas se deben cumplir todo el tiempo). Estas reglas son la *regla de integridad de entidades* y la *regla de integridad referencial*. Antes de definirlas, es preciso conocer el concepto de *nulo*.

Nulos

Cuando en una tupla un atributo es desconocido, se dice que es *nulo*. Un nulo no representa el valor cero ni la cadena vacía, éstos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido.Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

Regla de integridad de entidades

La primera regla de integridad se aplica a las claves primarias de las relaciones base: *ninguno de los atributos que componen la clave primaria puede ser nulo.*

Por definición, una clave primaria es un identificador irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permite que parte de la clave primaria sea nula, se está diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se contradice la irreducibilidad.

Nótese que esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves alternativas.

Regla de integridad referencial

La segunda regla de integridad se aplica a las claves ajenas: *si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos*.

La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. La cuestión es ¿qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal? Existen dos opciones: *rechazar* la operación, o bien *aceptar* la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

Reglas de negocio

Además de las dos reglas de integridad anteriores, los usuarios o los administradores de la base de datos pueden imponer ciertas restricciones específicas sobre los datos, denominadas *reglas de negocio*.

Por ejemplo, si en una oficina de la empresa inmobiliaria sólo puede haber hasta veinte empleados, el SGBD debe dar la posibilidad al usuario de definir una regla al respecto y debe hacerla respetar. En este caso, no debería permitir dar de alta un empleado en una oficina que ya tiene los veinte permitidos.

Hoy en día aún existen SGBD relacionales que no permiten definir este tipo de restricciones ni las hacen respetar.

1.2 MÓDULO II SISTEMAS MANEJADORES DE BASES DE DATOS RELACIONALES (SGBD)

1.2.1 SGBD

Entre la base de datos física (es decir, los datos tal y como están almacenados en la realidad) y los usuarios del sistema, existe un nivel de software, denominado, manejador de bases de datos (MBD) o, en la mayoría de los casos, el sistema administrador de bases de datos SGBD.
Un SGBD es el conjunto de programas que permiten la definición, manipulación, explotación y control de acceso para una o varias bases de datos.

Algunas características de los SGBD son:

- Facilitan la integridad, seguridad y acceso de los datos.
- Los datos se almacenan como mínima redundancia.
- Las aplicaciones son independientes del almacenamiento físico de los datos.

Un SGBD debe permitir las siguientes condiciones en una base de datos:

- Los datos han de estar almacenados juntos.
- Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento.
- Los datos son compartidos por diferentes usuarios y programas de aplicación; existe un mecanismo común para la inserción, actualización, borrado y consulta de los datos.
- Los procedimientos de actualización y recuperación, comunes, y bien determinados, habrán de ser capaces de conservar la integridad, seguridad y confidencialidad del conjunto de datos.
- Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes SGBD.

1.2.2 Función del SGBD

Conceptualmente lo que sucede en un SGBD cuando un usuario realiza alguna petición, se presenta lo siguiente:

El usuario solicita alguna petición a la base de datos empleando algún sublenguaje de datos por ejemplo (SQL).

El SGBD interpreta esa solicitud y la analiza.

El SGBD inspecciona en orden el esquema externo de ese usuario, la correspondencia externa/conceptual asociada, el esquema conceptual, la correspondencia conceptual/interna y la definición de la estructura de almacenamiento.

El SGBD ejecuta las operaciones necesarias sobre la base de datos almacenada y devuelve una respuesta al usuario.

1.2.3 Arquitectura Cliente/Servidor

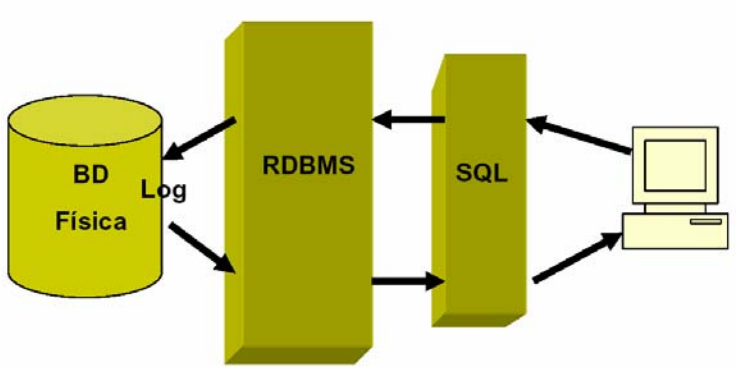

Figura 2.0. Arquitectura Cliente/Servidor.

1.2.4 Esquema de seguridad en el SGBD

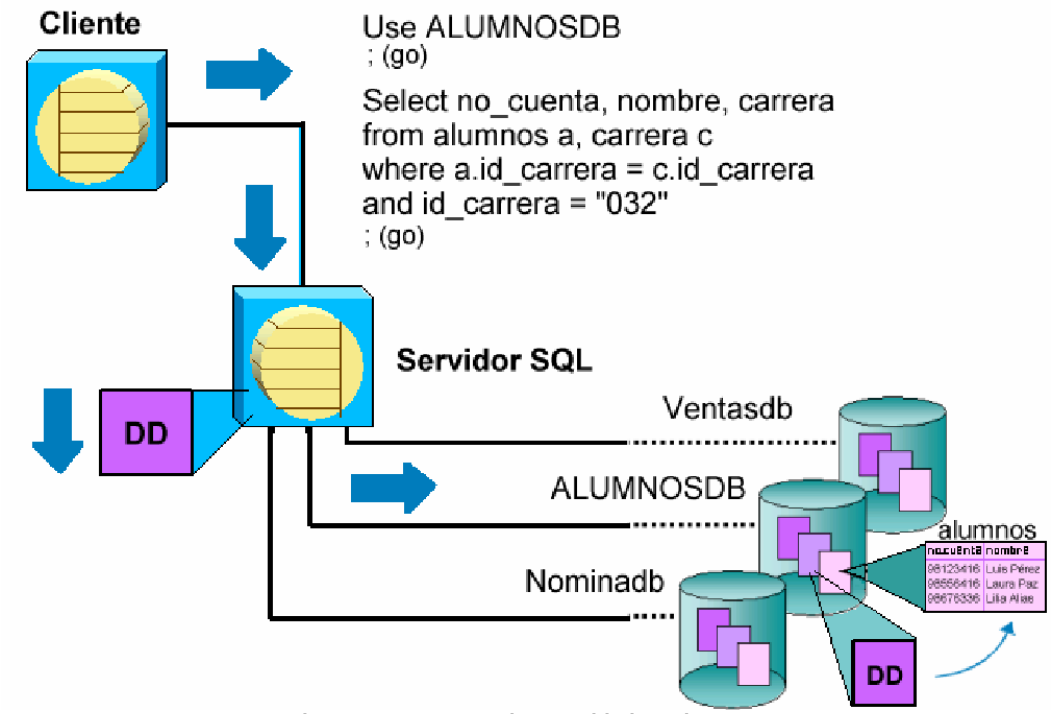

Figura 2.1. Esquema de seguridad en el SGBD.

Dentro del Sistema Manejador de Base de Datos (SGBD) podemos encontrar un acceso multicapas, como el que se muestra a continuación:

El usuario final debe tener una cuenta válida dentro de la capa del servidor (SGBD). *Seguridad a Nivel Servidor.*

El usuario final debe ser un usuario válido dentro de la capa de la base de datos. *Seguridad a Nivel de Base de Datos.*

El usuario final deberá tener permiso dentro de la capa de los datos.

Seguridad a Nivel de Permisos sobre Objetos y Comandos.

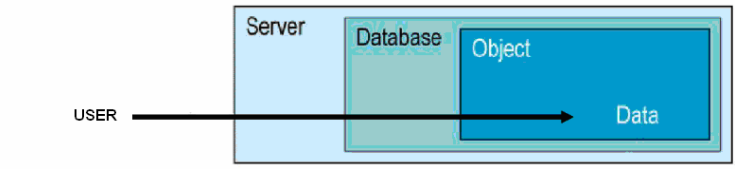

Figura 2.2. Nivel de permisos.

1.2.5 Componentes de un SGBD

1.2.6 DDL o Lenguaje de Definición de Datos

Se utiliza para crear, eliminar o modificar tablas, índices, vistas, triggers, procedimientos; es decir, nos permite definir la estructura de la base de datos mediante comandos como crear (*Create*), eliminar (*Drop*), o alterar (*Alter*).

-**create**. Utilizado para crear nuevas bases de datos, tablas, campos, índices, vistas, defaults, reglas, procedimientos, procedimientos, triggers.
-**alter**. Utilizado para modificar la estructura de una tabla para agregar campos o constrainst.
-**drop**. Utilizado para eliminar bases de datos, tablas, campos, índices, vistas, defaults, reglas, procedimientos, procedimientos, triggers.

1.2.7 DML o Lenguaje de Manipulación de Datos

Se utiliza para realizar la consulta y edición de la información contenida en la base de datos, esto implica: seleccionar, insertar, borrar, modificar datos.

Los DML se distinguen por sus sublenguajes de recuperación subyacentes; se pueden distinguir dos tipos de DML, el procedural y el no procedural. La principal diferencia entre ambos es que en los lenguajes procedurales se tratan los registros individualmente, mientras que en uno no procedural se opera sobre un conjunto de registros.

Las instrucciones relacionadas con este componente son:

-**select**. Permite realizar consultas a la base de datos.
-**insert**. Empleado para agregar registros a una tabla.
-**update**. Utilizado para modificar los valores de los campos de una tabla.
-**delete**. Utilizado para modificar los valores de los campos de una tabla.

1.2.8 DCL o Lenguaje de Control de Datos

Se utiliza para la definición de los privilegios de control de acceso y edición a los elementos que componen la base de datos (seguridad), es decir, permitir o revocar privilegios.

Los permisos a nivel base de datos pueden otorgarse a usuarios para ejecutar ciertos comandos dentro de la base o para que puedan manipular objetos y los datos que puedan contener estos.

Las instrucciones relacionadas con este componente son:

-**grant**. Permite otorgar permisos a los usuarios sobre los objetos definidos en la base de datos, así como las operaciones a utilizar sobre ellos.

-**revoke**. Permite revocar permisos sobre los objetos definidos en la base de datos y las operaciones sobre los mismos.

1.2.9 DD o Diccionario de Datos

El contenido del diccionario puede considerarse como “datos acerca de los datos” (los cuales comúnmente reciben el nombre de metadatos), es decir, definiciones de otros objetos de la base de datos **[Administración de Base de datos 2005].**

En particular, todos los diversos esquemas (externo, conceptual e interno), se almacenan físicamente en el diccionario, tanto en forma fuente como en forma objeto. Un diccionario amplio incluirá también las referencias cruzadas que indican, por ejemplo que partes de datos utiliza cada programa, que informes necesita cada departamento, etc. De hecho, el diccionario puede integrarse a la base de datos que describe, y, por tanto, incluir su propia descripción.

Debe ser posible consultar el diccionario de la misma manera que cualquier otra base de datos, de modo que, por ejemplo, el DBA pueda describir con facilidad que programas tienen probabilidad de ser afectados por un cambio propuesto al sistema.

1.2.10 Principales Funciones del DD

- Describe todos los elementos en el sistema.
- Los elementos se centran en los datos.
- Comunica los mismos significados para todos los elementos del sistema.
- Documenta las características del sistema.
- Facilita el análisis de los detalles para evaluar las características y determinar cómo deben realizarse los cambios.
- Localiza errores y omisiones del sistema.

1.2.11 Arquitectura

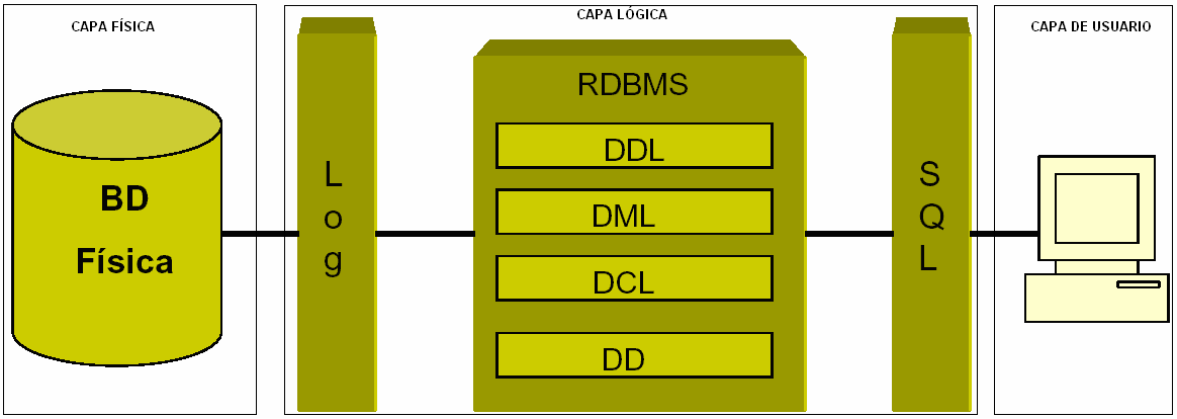

Figura 2.3. Arquitectura del SGBD

1.2.12 Usuarios en una Base de Datos

El programador de aplicaciones, quien se encarga de escribir los programas de aplicación que utilizan la base de datos.

El usuario final, el cual tiene acceso a los datos de la base a través de alguna aplicación desarrollada o utilizando una interfaz incluida como parte integral de los programas del SGBD.

El DBA (database administrator, Administrador de la base de datos).

Roles o Funciones
Los roles o perfiles sirven como medio para conceder privilegios sobre todo el sistema a un usuario que los requiera.

Estos permisos pueden verse reflejados sobre objetos o sobre el mismo sistema. En los SGBD ya vienen predeterminados algunos, sin embargo no en todos se pueden crear nuevos roles.

DBA (database administrator, administrador de la base de datos).

DBO (database owner, Dueño de la base de datos).

SSO (Security System Oficer, Oficial del sistema de seguridad).

Oper (Operador)

Administrador de Base de Datos

Lleva a cabo tareas administrativas inconexas aplicaciones específicas. El administrador del sistema no es necesariamente único; el rol puede ser otorgado a cualquier número de cuentas individualmente y para ello las funciones del DFB deben estar centralizadas o muy bien coordinadas. Las tareas del DBA incluyen:

- Decidir el contenido en estructura de la base de datos.
- Crear la estructura de almacenamiento y los métodos de acceso.
- Administrar y controlar la seguridad física y lógica de la base de datos.
- Monitoreo del comportamiento y crecimiento de la base de datos.
- Procedimientos de respaldo y depuración de la base de datos.
- Salvaguardar la documentación, respaldos y diccionario de datos tanto de la base datos como de sus aplicaciones.
- Procedimientos de contingencia y recuperación de la base de datos.
- Modificar la base de datos o la descripción de la organización física.
- Otorgar permisos de acceso y prioridades a los diferentes usuarios.
- Especificar las limitaciones de integridad.
- Ser el enlace con el usuario (desarrolladores, gente de sistemas y en muy pocos casos con usuarios de aplicaciones).
- Instalación del servidor SQL.
- Diagnóstico de problemas del sistema, así como la corrección de los mismos.
- Otorgar y revocar roles en el servidor.
- Configuración del servidor SQL.
- Control de procesos en el servidor.

SSO (Oficial del sistema de Seguridad)

El Oficial del Sistema de Seguridad es responsable de la seguridad del sistema y tiene las siguientes funciones.

- Crear cuentas de usuario.
- Bloquear cuentas.
- Otorgar y revocar los roles de sso y oper.
- Cambiar password a cualquier cuenta en el sistema.
- Poner intervalos de expiración a los passwords.
- Manejar el sistema de auditoria.

Oper (Operador)

El operador del sistema es responsable de respaldar y recuperar todas las Bases de Datos del Sistema.

En Oracle tenemos:

OSOPER: Permite al usuario llevar a cabo el inicio y apagado del servidor, respaldo y mantenimiento de bases de datos.

OSDBA: Administrador del sistema, tiene todos los privilegios del sistema, sólo este role permite crear bases de datos y recuperarlas.

En SQL Server tenemos:

sysadmin: Administrador del sistema (sa odba).

serveradmin: Se usa para conceder a un usuario la autoridad para realizar cambios en la configuración del servidor.

processadmin: Permite gestionar los procesos activos en SQL Server.

dbcreator: Proporciona a un usuario la capacidad de crear y modificar bases de datos en el sistema.

diskadmin: Este se proporcionar por lo regular con el anterior y sirve para manejar los dispositivos de almacenamiento (archivos).

1.2.13 Transact-SQL

Para comunicarse con el servidor SQL y manipular objetos almacenados en él, los programas cliente y los procedimientos almacenados usan una variedad de SQL llamado: Transact-SQL (T-SQL). Este está basado en el estándar ANSI SQL 89, pero además contiene extensiones importantes que permiten flexibilidad y facilidad en el lenguaje (estructuras de control de flujo, variables locales, procedimientos almacenados y triggers).

Transact-SQL contiene algunas adiciones al estándar, como las siguientes:

- Contiene estructuras de control de flujo de programas, declaración de variables, paso de parámetros, entre otras. Estas permiten crear, por ejemplo, procedimientos almacenados y triggers.
- Permite la definición de tipos de datos por parte del usuario.
- Amplía la función de la instrucción SELECT.
- Agrega una gran variedad de funciones para el manejo de datos tipo carácter, fecha y numéricos, a las contenidas en el estándar SQL.

1.3 MÓDULO III SQL (Structured Query Language)

SQL (Structure Query Language; Lenguaje Estructurado de Consulta) es un lenguaje de consulta para bases de datos, adoptado como estándar de la industria en 1986. Desde entonces se han realizado revisiones al estándar para incorporar nueva funcionalidad conforme la industria de las bases de datos lo va requiriendo. Una de las revisiones más importantes fue la de 1992, conocida como ANSI SQL92.

Los componentes del SQL son:

DDL (Data Definition Language; Lenguaje de Definición de Datos). Permite crear, modificar y eliminar estructuras de datos como: tablas, bases de datos, índices, etc.

DML (Data Manipulation Language; Lenguaje de Manipulación de Datos). Permite consultar, insertar, modificar y eliminar datos de las tablas.

DCL (Data Control Language; Lenguaje de Control de Datos). Permite establecer los privilegios de acceso a los datos, en otras palabras, establece la seguridad de la base de datos.

1.3.1 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 cierta información.

1.3.2 Tablas

Las tablas son el elemento fundamental que compone a una base de datos relacional porque todo gira en torno a ellas. Las tablas son estructuras de almacenamiento que albergan la información en forma de registros (renglones) que deben de ser identificados de manera única y esto se logra a través de una llave primaria.

1.3.3 Creación de Tablas

La sintaxis básica para la creación de una tabla es la siguiente:

```
CREATE TABLE <nombre_tabla>
(
<nombre_campo1> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
<nombre_campo2> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
<nombre_campo3> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>],
…
<nombre_campoN> <tipo_de_dato> [NULL | NOT NULL] [DEFAULT <val_predeterminado>]
)
```

1.3.4 Modificación de Tablas

La sintaxis varía de un SGBD, debido a las diferentes características soportadas, sin embargo todos ellos utilizan la cláusula ALTER TABLE para realizar las modificaciones posibles, a una tabla.

La sintaxis general para agregar un campo es la siguiente:

```
ALTER TABLE <nombre_tabla>
 add <nombre_campo> <tipo_dato> [DEFAULT <val_predeterminado>] [NOT NULL | NULL]
```

Para eliminar una tabla y los datos que contiene, así como sus índices, triggers y permisos se utiliza la siguiente instrucción:

```
DROP TABLE <nombre_tabla>
```

1.3.5 Llaves e Índices

Un índice es una estructura de almacenamiento físico que permiten recuperar datos de una manera muy eficiente.

En un esquema relacional, cada registro dentro de una tabla debe de ser identificado de manera única, y esto se logra a través de una llave primaria, a la cual se le genera de manera automática un índice, que ayuda a ser más eficiente el proceso de consulta de la información. También existen las llaves foráneas, que son las columnas que hacen referencia a la llave primaria de otra tabla. A través de ellas se establecen relaciones entre tablas.

Al crear una tabla se puede especificar que campo o campos forman parte de la llave primaria.

Ejemplo:

```
CREATE TABLE pais
(
  id_pais numeric(3) NOT NULL PRIMARY KEY,
  nombre varchar(100)
)
```

En el ejemplo anterior, la tabla solamente consta de un campo que forma la llave primaria “id_pais”, por ello es posible utilizar la cláusula PRIMARY KEY. Pero cuando la llave primaria consta de dos o más campos la sintaxis anterior no es útil, por lo cual se emplea la siguiente sintaxis:

```
CREATE TABLE pedido
(
  id_cliente numeric(10) NOT NULL,
  id_producto numeric(10) NOT NULL,
  fecha date,
  CONSTRAINT pedido_pk PRIMARY KEY (id_cliente, id_producto)
 )
```

Las llaves foráneas se pueden crear dentro de la definición de la tabla, o una vez que esta ya existe, se puede utilizar la cláusula ALTER TABLE para agregar esta restricción. A diferencia de las llaves primarias, las llaves foráneas no generan un índice, por lo que de ser necesario se deberá crear con la cláusula CREATE INDEX.

Ejemplo:

```
CREATE TABLE resultado
 (
  id_pais1          NUMBER(3) NOT NULL,
  id_pais2          NUMBER(3) NOT NULL,
  resultado         VARCHAR2(15) NOT NULL,
  CONSTRAINT pais_01_fk FOREIGN KEY (id_pais1) REFERENCES pais(id_pais),
  CONSTRAINT pais_02_fk FOREIGN KEY (id_pais2) REFERENCES pais(id_pais)
);
```

1.3.6 Creación de índices

Un índice es una estructura de almacenamiento físico que ocupa un espacio. Los índices ayudan al Servidor SQL a localizar datos y son transparentes para el usuario. El propósito principal de un índice es proporcionar un acceso más rápido a los datos, aunque en algunos casos su propósito es asegurar que el contenido de un campo sea único.

Para crear índices se utiliza el siguiente comando:

CREATE [UNIQUE] INDEX <*nombre_índice*>
ON <*nombre_tabla*>(<*campo*>,...)

Para eliminar un índice se utiliza la siguiente instrucción:

DROP INDEX <*nombre_índice*>

1.3.7 Manipulación de datos

La mayor parte del trabajo con SQL girará entorno a cuatro comandos:

Select. Permite seleccionar (recuperar) información de una tabla.
Insert. Permite agregar información a una tabla.
Delete. Permite eliminar información de una tabla.
Update. Permite actualizar información que existe en una tabla.

1.3.8 Join

La operación Join es en esencia un producto cartesiano entre dos tablas, donde se selecciona los renglones que satisfagan las condiciones indicadas que establecen la relación entre las tablas involucradas. Esta es una operación muy común en las bases de datos relacionales.

1.3.9 Cláusula SELECT

Esta cláusula indica que la instrucción a ejecutar es una consulta proyección y selección a la base de datos. SELECT nos permite indicar el nombre de los campos que queremos mostrar en la consulta. En caso de querer mostrar todos los campos de una tabla se emplea el comodín asterisco: *.

Es posible utilizar seudónimos (alias) para cambiar el nombre de las columnas mostradas en una consulta, esto puede servir para hacer más legible los resultados mostrados, o porque así lo requiere alguna aplicación. Para colocar los seudónimos es necesario especificarlo mediante la cláusula AS, de la siguiente forma: <nombre_campo> AS <otro_nombre>.

Ejemplo: Seleccionar el nombre del empleado de la tabla empleado y el nombre de departamento de la tabla departamento.

SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento

1.3.10 Cláusula FROM

La cláusula FROM sirve para indicar las tablas de las cuales se desea mostrar la información. Cuando una consulta involucra dos o más tablas, es indispensable establecer las relaciones que existen entre ellas (join) mediante una cláusula WHERE.

Ejemplo: Seleccionar el nombre del empleado de la tabla empleado y el nombre de departamento de la tabla departamento.

```
SELECT empleado.nombre, departamento.nombre
FROM empleado, departamento
WHERE departamento.id_departamento = empleado.id_departamento
```

1.3.11 Cláusula WHERE

La cláusula WHERE permite delimitar los registros que serán mostrados en la consulta, a través de criterios o condiciones. Es posible utilizar los operadores lógicos: OR , AND y NOT para combinar expresiones y refinar el criterio de consulta.

Para escribir condiciones de manera adecuada es muy importante recordar que:

Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.

Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

El valor NULL es un valor especial por lo cual se debe tener sumo cuidado cuando se desee utilizar condiciones con NULL. La única forma de comparar contra un valor NULL es utilizar el operador IS o IS NOT.

```
SELECT * FROM empleado WHERE comision = NULL;    --incorrecto
SELECT * FROM empleado WHERE comision is NULL;    --correcto
```

1.3.12 Cláusula GROUP BY

En la cláusula GROUP BY se indica el o los campos por los cuales se desea agrupar un conjunto de registros. Comúnmente esta agrupación va acompañada con una serie de funciones que realizan ciertas operaciones sobre el valor de los campos indicados. Estas funciones son conocidas como funciones de agregación o agrupación:

Función	Acción
COUNT(*)	Regresa el número de registros encontrados
COUNT(<*campo*>)	Regresa el número de registros cuyo valor del campo especificado no es nulo
SUM(<*campo*>)	Suma los valores de la columna especificada
AVG(<*campo*>)	Promedia los valores del campo especificado

MIN(<*campo*>)	Regresa el valor mínimo del campo especificado
MAX(<*campo*>)	Regresa el valor máximo del campo especificado

1.3.13 Cláusula HAVING

Esta cláusula es el equivalente a la cláusula WHERE, es decir, especifica un criterio o condición, pero la diferencia radica en que se ocupa únicamente cuando se desea especificar una función de agregación en la condición.

1.3.14 Cláusula ORDER BY

Esta cláusula nos permite indicar los campos por los cuales se desea ordenar la información mostrada. Es posible indicar si el orden es descendente o ascendente, de manera predeterminada es ascendente. Algunos SGBD permiten realizar este ordenamiento especificando, en lugar del nombre del campo, la posición de este.

1.3.15 Inserción de datos

A través de la instrucción INSERT de SQL, se introduce la información a una tabla.
La estructura general de este comando es la siguiente:

```
INSERT INTO <tabla> [(<nombreCampo1>, <nombreCampo2>, <nombreCampo3> ...)]
{VALUES (<valorCampo1>, <valorCampo2>, <valorCampo3> ... ) | <Expresión select> }
```

1.3.16 Cláusula INSERT

Esta cláusula permite indicar que la operación a realizar es la inserción de un registro.

1.3.17 Cláusula INTO

Esta cláusula permite indicar la tabla en donde se realizará dicha inserción. Únicamente se puede especificar una tabla a la vez. Después del nombre de la tabla puede o no ir el nombre de los campos donde se va insertar información, esto es opcional, pero es muy recomendable no omitirlos, por que le resta legibilidad a la instrucción.

Si no se especifica el nombre de los campos que se van a insertar, el SGBD identifica que se desea insertar información en cada uno de los campos, en el orden definido por la estructura de la tabla.

Por ejemplo, tomando a la tabla empleado cuya estructura define 11 campos con el siguiente orden: id_empleado, nombre, apellido_paterno, apellido_materno, edad, sexo, sueldo, porcentaje_comision, fecha_contratación, id_departamento, id_cargo

1.3.18 Cláusula VALUES

Esta cláusula permite especificar los valores a insertar para cada uno de los campos involucrados en la sentencia.

Para especificar los valores a insertar de manera adecuada es muy importante recordar que:

Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples.

Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico.

Ejemplo:

Insertar el registro de un nuevo empleado con clave 20, llamada Lorena Aguilar, de 19 años de edad, con un sueldo de 7,000 pesos, teniendo cargo de empleado, entrando el 26 de Abril del 2004 al departamento de recursos humanos.

```
INSERT INTO empleado (id_empleado id_departamento, id_cargo, nombre, edad, sexo, sueldo, fecha_contratacion)
VALUES (20, 2, 1, 'Lorena Aguilar', 'M', 7000.00, '2002-04-26')
```

En este ejemplo se puede ver que el orden de los campos especificados se encuentran en un orden diferente al de la estructura física de la tabla (cuyo orden es: id_empleado, nombre, edad, sexo, sueldo, porcentaje_comision, fecha_contratación, id_departamento, id_cargo) y además se omite el campo de porcentaje comisión ya que este solo se emplea para los vendedores y admite valores nulos.

1.3.19 Eliminación de registros

La instrucción delete elimina registros de la tabla indicada con la posibilidad de indicar un criterio, en caso de omitirlo, se eliminan todos los registros de la tabla.
La sintaxis es la siguiente:

```
DELETE FROM <nombre_tabla>
[ WHERE <condición>]
```

1.3.20 Cláusula DELETE

La cláusula DELETE permite indicar que la operación a realizar es una eliminación de registros

La cláusula FROM permite especificar la tabla de donde se desea eliminar registros.

Nota: En algunos SGBD, la cláusula FROM se puede emplear cuando es necesario especificar varias tablas que se encuentran involucradas en la condición de borrado.

La cláusula WHERE permite delimitar el conjunto de registros a eliminar, si no se especifica esta condición, se eliminaran todos los registros que contenga la tabla especificada. Para mayor detalle de las condiciones consulte la sección II.2.3.

1.3.21 Actualización de datos.

Para modificar o actualizar los valores de los registros de una tabla se utiliza el comando UPDATE. Si no se especifica una condición con la cláusula WHERE, todos los registros que existan en la tabla serán actualizados.
La sintaxis es la siguiente:

```
UPDATE <nombre_tabla>
SET <campo1> = <valor1>, <campo2> = <valor2>, ....
[ WHERE <condición>  ]
```

1.3.22 Cláusula UPDATE

La cláusula UPDATE es la que indica que la operación a ejecutar es una actualización. Después de la cláusula se especifica el nombre de la tabla en donde se encuentra la información que deseamos modificar. Sólo se puede especificar una tabla a la vez.

Ejemplos:

Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.

```
UPDATE empleado
SET sueldo = 27000, edad = 27
WHERE id_empleado = 12
```

1.3.23 Cláusula SET

Esta cláusula permite especificar los campos que se desean modificar y su nuevo valor. La cláusula se coloca una sola vez aunque sean varios campos los que se deseen modificar.
Ejemplos:
Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.

```
UPDATE empleado
SET sueldo = 27000, edad = 27
WHERE id_empleado = 12
```

La cláusula WHERE permite especificar un criterio para delimitar el conjunto de registros a modificar.
Ejemplos:

Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.

```
UPDATE empleado
set sueldo = 27000, edad = 27
WHERE id_empleado = 12
```

1.3.24 Vistas

Una vista es una tabla qué no ocupa espacio de almacenamiento para la información que contiene, porque su estructura e información está definida a través de la ejecución de una instrucción SELECT. Las vistas tiene dos usos: el primero es para simplificar el acceso a datos que se ocupan frecuentemente y que requieren una sentencia de SQL muy compleja para dicho acceso; y el segundo es con fines de seguridad, que permitan mantener ocultas ciertas columnas.

La sintaxis para crear una vista es:

CREATE VIEW <*nombre_vista*> AS <*instrucción SELECT*>
Para eliminar una vista se utiliza la siguiente instrucción:
DROP VIEW <*nombre_vista*>

1.3.25 Funciones de utilidad

Aunque el estándar SQL92 define las funciones con las que debe contar un SGBD, desafortunadamente en la práctica, no todos los SGBD cumplen con el estándar.

1.3.26 Manejo de Transacciones

Los SGBD deben de implementar un mecanismo a través del cual, los cambios a realizar en la información, a través de una instrucción INSERT, DELETE o UPDATE, no son efectuados hasta que el usuario lo indique explícitamente. Una transacción puede comprender una o más instrucciones SQL.

Una transacción es atómica, porque todas las instrucciones de SQL deben completarse con éxito, o ninguna de ellas. Una vez que el SGBD determina que la transacción fue exitosa es necesario que la información sea almacenada de manera permanente. Una base de datos transaccional garantiza que todas las operaciones realizadas en una transacción sean guardadas en almacenamiento permanente antes de que ésta sea reportada como completada, previniendo así, perdida de información por fallas del equipo, por ejemplo en un corte del suministro de energía.

Cuando múltiples usuarios realizan transacciones de manera concurrente, cada uno de ellos no debe ver los cambios incompletos realizados por los demás. En el momento que transacción finaliza adecuadamente y es almacenada permanentemente, los cambios se vuelven visibles para todos los demás usuarios.

1.3.27 Commit y Rollback

El comando commit permite indicar que los cambios a realizar dentro de una transacción sean llevados a cabo de manera permanente, y el comando rollback permite deshacer los cambios. El SGBD reserva un espacio de almacenamiento, donde registra todas las instrucciones de SQL que se deben de ejecutar. De esta manera es posible deshacer los cambios realizados por operaciones UPDATE, DELETE o INSERT.

Las transacciones se inician de distinta manera, aunque similar. Por ejemplo en PostgreSQL, una transacción es iniciada mediante la cláusula BEGIN y en Sybase se emplea BEGIN TRANSACTION, seguida de las instrucciones de SQL a realizar y finalmente se emplea la cláusula COMMIT para indicar que las modificaciones deben realizarse de manera permanente o en caso que se desee cancelar la operación, se ocupa la cláusula ROLLBACK.
Ejemplo:

```
BEGIN;
UPDATE cuenta SET saldo = saldo - 1000.00
   WHERE cliente = 'Fernanda';

UPDATE cuenta SET saldo = saldo + 1000.00
   WHERE cliente = 'Alfredo';
COMMIT;
```

1.3.28 Estructuras de Control de Flujo

Este tipo de estructuras se emplea en el desarrollo de programas, que en un SGBD puede ser un procedimiento almacenado. Los SGBD comerciales son los que se han desarrollado más en el aspecto de programación en la base de datos, sin embargo las instrucciones aunque similares, difieren entre uno y otro.

1.3.29 IF-THEN-END IF

La sintaxis en Sybase, MS SQL Server:

IF <*expresión*>
<*bloque A de instrucciones*>
ELSE
<*bloque B de instrucciones*>

1.3.30 IF-THEN-ELSE-END IF

La sintaxis en Sybase, MS SQL Server:

IF <*expresión 1*>
<*bloque A de instrucciones*>
ELSE IF <*expresión 2*>
<*bloque B de instrucciones*>
ELSE IF <*expresión 3*>
<*bloque C de instrucciones*>
ELSE
<*bloque D de instrucciones*>

1.3.31 FOR LOOP

La sintaxis en Oracle es la siguiente:

FOR <*variable*> IN <*valor inicial*>..<*valor final*> LOOP
  <*lista_de_instrucciones*>
END LOOP;

Este ciclo iterativo inicializa la variable especificada y ejecuta las instrucciones cíclicamente hasta que la variable alcanza el valor final.

1.3.32 WHILE LOOP

Esta estructura se utiliza cuando se desea realizar una o un grupo de instrucciones repetidamente. La sintaxis en Sybase y MS SQL Server es:

WHILE <*expresión*>
        <*instrucciones a ejecutar mientras la expresión sea verdadera*>
[BREAK]
[CONTINUE]

Donde break se utiliza para salir del ciclo en el punto donde esta palabra se encuentra y continué se utiliza para regresar directamente a evaluar nuevamente la expresión.

Ejemplo:

```
WHILE (SELECT AVG(precio) FROM productos) < 1000
BEGIN
        IF (SELECT MAX(precio) FROM productos) <= 1500
        BEGIN
                PRINT "Actualizando Precios"
                UPDATE productos SET precio = precio * 1.02
                CONTINUE
        END
        ELSE
        BEGIN
                PRINT "Llego al límite establecido"
                BREAK
        END
END
```

1.3.33 Procedimientos almacenados

Un procedimiento almacenado es un conjunto de comandos de SQL que pueden ser interpretados y almacenados en el servidor. Una vez realizado esto, los clientes no necesitan volver a teclear todas las instrucciones sino únicamente hacer referencia al procedimiento. Esto mejora el rendimiento del servidor, ya que la instrucción de SQL solamente es revisada una sola vez y menos información debe ser enviada entre el cliente y el servidor.

El lenguaje que se emplea para programar los procedimientos almacenados, varía de un SGBD a otro, y existen algunos que permiten programar en más de un lenguaje.

1.3.34 Declaración de variables

Las variables son elementos fundamentales en la programación de procedimientos. Las variables deben ser declaradas al inicio del programa, antes de utilizarlas, para ello existe la cláusula DECLARE en Sybase, Ms SQL Server y Oracle.

Las variables locales sólo existen durante la ejecución del procedimiento donde son declaradas; cuando éste termina, las variables locales son eliminadas.

Sintaxis para declarar una variable:

```
DECLARE  <@variable> <tipo de dato> <(tamaño)> [, .....]
```

1.3.35 Creación de procedimientos Almacenados

Para crear procedimientos almacenados se utiliza la siguiente instrucción en Sybase y MS SQL Server:

```
CREATE PROCEDURE <nombre_procedimiento>
[(@<nombre_parámetro> <tipo_de_dato>)]
AS
<instrucciones SQL>
RETURN
```

Para ejecutar un procedimiento almacenado se utiliza la siguiente instrucción:

```
[EXEC]  <nombre_procedimiento> [<parámetro>, ...]
```

Para eliminar un procedimiento almacenado, se dispone de la instrucción DROP PROCEDURE. La sintaxis es la siguiente:

```
DROP PROCEDURE <nombre_procedimiento>
```

1.3.36 Paso de parámetros

El uso de parámetros incrementa la flexibilidad de un procedimiento almacenado. Éstos se definen desde la creación del procedimiento y deben ser proporcionados por el usuario al momento de ejecutarlo.

Un ejemplo en Sybase de un procedimiento con parámetros es el siguiente:

```
CREATE PROCEDURE sueldo_empleado
@sueldo numeric(8,2)
AS
SELECT nombre, sueldo
FROM    empleado
WHERE  sueldo <= @sueldo
ORDER BY nombre
RETURN
```

Para ejecutar el procedimiento se utiliza la siguiente instrucción:

```
EXEC sueldo_empleado 10000
```

1.3.37 Triggers

Un trigger es un procedimiento almacenado que es invocado cuando un evento en particular ocurre en una tabla. Por ejemplo, se puede ejecutar (disparar) un procedimiento almacenado cada vez que se borre, actualice o inserte un registro. Los procedimientos almacenados que se invocan tiene la restricción de que no pueden manejar parámetros ni ser invocados directamente.

En Sybase y MS SQL Server un trigger al dispararse crea dos tablas temporales las cuales sólo pueden ser accesadas dentro del trigger, y poseen la misma estructura de la tabla a la que está ligada. Estas tablas son: Inserted y Deleted.

Tabla	Contenido	En Triggers
Inserted	Contiene los registros que se van a agregar a la tabla, como resultado de los comandos Insert y Update	Insert, Update
Deleted	Contiene los registros que se van a eliminar de la tabla, como resultado de los comandos Delete y Update	Delete, Update

Para crear un trigger se utiliza la siguiente sintaxis:

Sybase/Ms SQL Server	Oracle
CREATE TRIGGER <*nombre_trigger*> ON <*nombre_tabla*> FOR [INSERT | UPDATE | DELETE] AS <*instrucciones SQL*> ... RETURN	CREATE [OR REPLACE] TRIGGER <*nombre_trigger*> {BEFORE|AFTER} {INSERT|DELETE|UPDATE} [OF COLUMN <*nombre_columna*>] ON <*nombre_tabla*> [FOR EACH ROW] <*codigo pl/sql*>

La cláusula TRIGGER indica que el objeto que se desea crear es un TRIGGER. Inmediatamente después de esta cláusula, se coloca el nombre del trigger.

1.3.38 Cláusula BEFORE y AFTER

El trigger es una acción que se ejecuta cuando se da un determinado evento, por ejemplo una inserción. Con las cláusulas BEFORE y AFTER se especifica en que momento debe de activarse la acción, si antes de la realización del evento o después. A un lado de cualquiera de estas dos cláusulas, se especifica el evento que va disparar la acción, pudiendo ser INSERT, DELETE o UPDATE que corresponde a una inserción, eliminación ó actualización de datos, respectivamente.

En el caso de una actualización es posible especificar que únicamente cuando la modificación afecte a cierta columna se dispare la acción. Esto se realiza empleando la cláusula OF COLUMN seguida del nombre de la columna.

1.3.39 Cláusula ON

Permite especificar en que tabla se va a asociar el disparador de acción. Después de la cláusula ON se coloca el nombre de la tabla.

1.3.40 Cláusula FOR EACH_ROW

Esta cláusula permite especificar que la acción se ejecutara a nivel renglón. Es decir por cada registro afectado por la inserción, modificación o eliminación se realizara la acción especificada en el trigger.

La sintaxis para eliminar un trigger es muy simple, únicamente se utiliza la instrucción:

DROP TRIGGER <*nombre_trigger*>

1.3.41 Cursores

Los cursores son apuntadores que tienen acceso de manera individual a un registro dentro de un conjunto de resultados, provenientes de una consulta a la base de datos.

Los cursores se emplean dentro procedimientos almacenados cuando se necesita realizar operaciones complejas de selección o actualización de datos, o cuando se necesita realizar operaciones que afecte registro por registro.

Uso de Cursores

Al utilizar cursores se deben de seguir los siguientes pasos:

- Declaración del cursor
- Apertura del cursor
- Recorrer los registros del cursor (realizar procesos)
- Cerrar el cursor

Declaración del cursor

Sybase/MS SQL Server	Oracle
DECLARE <*nombre_cursor*> CURSOR FOR <*instrucción Select*> [FOR {READ ONLY | UPDATE [OF <*campos*>] }]	DECLARE CURSOR <*nombre_cursor*> IS <*instrucción Select*>;

Apertura del cursor

Sybase/MS SQL Server	Oracle
OPEN <*nombre_cursor*>	OPEN <*nombre_cursor*>;

Recorrer los registros

Sybase/MS SQL Server	Oracle
FETCH <*nombre_cursor*> INTO <*variables*>	FETCH <*nombre_cursor*> INTO <*variables*>;

Podremos recuperar filas mientras la consulta SELECT tenga filas pendientes de recuperar. Para saber cuándo se debe detener el ciclo de instrucciones FETCH, se realiza lo siguiente:

Sybase/MS SQL Server	Oracle
Se evalúa la variable @@sqlstatus/@@fetch_status respectivamente que contiene los siguientes valores: 0/0 Se logró el acceso al registro 1/-1 No se logró el acceso al registro 2/-2 No hay más registros	Se realiza la evaluación de alguno de los siguientes atributos: <*nombre_cursor*>%FOUND BOOLEAN Retorna si la última fila recuperada fue válida <*nombre_cursor*>%ISOPEN BOOLEAN Retorna si el cursor está abierto <*nombre_cursor*>%NOTFOUND BOOLEAN Retorna si la última fila fue inválida <*nombre_cursor*>%ROWCOUNT NUMBER Retorna el número de filas recuperadas

Así lo acción más típica es recuperar filas mientras queden alguna por recuperar en el servidor

Sybase/MS SQL Server	Oracle

FETCH <*nombre_cursor*> INTO <*@variables*>
WHILE (@@sqlstatus = 0)
BEGIN
<*instrucciones para procesar cada registro*>
/* Avanza al siguiente registro */.
FETCH <*nombre_cursor*> INTO <*@variables*>
END
	LOOP
	FETCH <*nombre_cursor*> INTO <*variables*>;
	EXIT WHEN <*nombre_cursor*>%NOTFOUND;
	<*instrucciones para procesar cada registro*>
	END LOOP;

Cierre del cursor

Sybase/MS SQL Server	Oracle
DEALLOCATE [CURSOR] <*nombre_cursor*>	CLOSE <*nombre_cursor*>;

1.4 MÓDULO IV PROGRMACIÓN DE CLIENTES

1.4.1 PHP (Hypertext Pre-processor)

Es un lenguaje de programación usado frecuentemente para la creación de contenido para sitios web con los cuales se puede programar la generación de paginas HTML dinámicas y los códigos de fuente. PHP es un acrónimo recursivo que significa "PHP Hypertext Pre-processor" (inicialmente PHP Tools, o, Personal Home Page Tools), y se trata de un lenguaje interpretado usado para la creación de aplicaciones para servidores, o creación de contenido dinámico para sitios web. Últimamente también para la creación de otro tipo de programas incluyendo aplicaciones con interfaz gráfica usando las librerías GTK+.

Conexiones

Las conexiones persistentes son enlaces SQL que no se cierran cuando la ejecución del script termina. El comportamiento de estas conexiones es el siguiente.

Cuando se invoca una conexión de este tipo, PHP comprueba si existe una conexión de este mismo tipo o por el contrario, se trata de una nueva conexión. En el caso de que exista, se procede a su uso, y en el caso de que no exista, la conexión se crea. Dos conexiones se consideran iguales cuando están realizadas sobre el mismo servidor, con el mismo usuario y la misma contraseña.

Pero en realidad, estas conexiones permanentes, no proporcionan ningún tipo de función adicional frente a conexiones temporales, debido a la forma en que los servidores Web funcionan.

Aún así se utilizan debido a la eficiencia, debido al tiempo de establecimiento de la conexión, y debido a que si tienes una sola conexión sobre el servidor, irá mucho más rápido que si tienes 10 conexiones temporales, puesto que la carga que soporta es diferente.

1.4.2 JSP (Java Server Pages)

En el campo de la informática, es una tecnología para crear aplicaciones Web basadas en java.

Es un desarrollo de la compañía Sun Microsystems y su funcionamiento se basa en scripts, que utilizan una variante del lenguaje java.

La **JSP** es una tecnología Java que permite a los programadores generar contenido dinámico para Web, en forma de documentos HTML, XML o de otro tipo. Las JSP's permiten al código Java y a algunas acciones predefinidas ser incrustadas en el contenido estático del documento Web.

En las **JSP** se escribe el texto que va a ser devuelto en la salida (normalmente, código HTML) incluyendo código java dentro de él, para poder modificar o generar contenido dinámicamente. El código java se incluye dentro de las marcas de etiqueta <% y %>; a esto se le denomina scriptlet.

El problema es comunicar un programa o aplicación con una base de datos y más que comunicar se pretende que el programa o aplicación realice una serie de procesos u operaciones con la base de datos o mejor aun con el conjunto de tablas que contiene una base de datos.

La primera nota a recordar es que una base de datos puede estar físicamente en el servidor y en algún fólder o directorio del disco duro de dicha maquina servidor

Sin embargo también es necesario conocer que así como existen servidores de paginas (web server), servidores de correo (mail server), servidores de ftp (ftp server), etc, también existen servidores de bases de datos (database server), los mas comunes son el sqlserver de microsoft, oracle, mysql, etc, estos servidores también pueden crear, administrar y procesar una base de datos.

El modo de comunicarse entre nuestro programa o aplicación y la base de datos (ya sea física o un servidor de base de datos) implica que ambos manejen un lenguaje de programación común, es decir no se puede mandar una instrucción en csharp, o en basic o pascal a la base de datos y además esperar que esta ultima la entienda (para entender esto, una razón muy sencilla es que la base de datos tendría que conocer o comprender todos los lenguajes de programación), para resolver este problema de comunicación es que se usa un lenguaje común de bases de datos que tanto los lenguajes de programación existentes como las bases de datos entienden, este lenguaje común de bases de datos es el SQL (structured query languaje) o lenguaje estructurado de consultas.

Objetos:

Objeto JDBCODBCDRIVER: Objeto que se utiliza para traducir las instrucciones del lenguaje SQL a las instrucciones del lenguaje original de la base de datos.

Objeto CONNECTION: objeto que se utiliza para establecer una conexión o enlace a la base de datos.

Objeto RESULTSET: Es la representación en memoria de una de las tablas de la base de datos en disco, se puede entender como una tabla virtual, recordar que generalmente todos los procesos que se realicen con la tabla (insertar registros, eliminar registros, etc.) se realizaran realmente contra un resulset y no provocaran ningún cambio en la tabla física en disco, resulset tiene un conjunto de métodos muy útiles y muy usados para el proceso de los renglones de la tabla virtual.

Objeto STATEMENT: Este objeto y sus dos métodos executequery (solo para select de sql) y executeupdate (solo para insert, update y delete de sql) son los métodos que se utilizaran para comunicarse con la tabla física en disco.

1.4.3 ASP (Active Server Pages)

Es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

La tecnología ASP está estrechamente relacionada con el modelo tecnológico de su fabricante. Intenta ser solución para un modelo de programación rápida ya que programar en ASP es como programar en Visual BASIC, por supuesto con muchas limitaciones ya que es una plataforma que no se ha desarrollado como lo esperaba Microsoft.

Lo interesante de este modelo tecnológico es poder utilizar diversos componentes ya desarrollados como algunos controles ActiveX. Otros problemas que han hecho evolucionar esta tecnología es el no disponer de información "que oriente a quienes desean aprenderla y resulta muy costosa en tiempo descubrir aquí y allá toda la información para volverla altamente útil".

ASP ha pasado por cuatro iteraciones mayores, ASP 1.0 (distribuido con IIS 3.0), ASP 2.0 (distribuido con IIS 4.0), ASP 3.0 (distribuido con IIS 5.0) y ASP.NET (parte de la plataforma .NET de Microsoft). Las versiones pre-.NET se denominan actualmente (desde 2002) como ASP *clásico*.

En el último ASP clásico, ASP 3.0, hay seis objetos integrados disponibles para el programador, Application, ASPError, Request, Response, Server y Session. Cada objeto tiene un grupo de funcionalidades frecuentemente usadas y útiles para crear páginas web dinámicas.

Las páginas pueden ser generadas mezclando código de scripts del lado del servidor (incluyendo acceso a base de datos) con HTML. Por ejemplo

Conexión con DSN

La conexión con DSN es la más cómoda, pero sólo se puede utilizar si tenemos acceso al panel de control de la máquina servidor. Por supuesto si estamos construyendo una intranet tenemos el server a nuestro alcance y a su panel de control.

Si simplemente estamos aprendiendo ASP y usamos el PWS (Personal Web Server) o el IIS 4 de NT también disponemos de esta comodidad.

Veamos como se realiza la conexión a una base de datos de Microsoft Access:

Creamos nuestra base de Datos en Microsoft Access y la guardamos. Luego vamos a *Inicio* > *Configuración* > *Panel de Control* y allí elegimos *Fuentes de Datos ODBC*

Al ingresar nos encontramos con una pantalla que es el administrador de orígenes de datos ODBC. En la solapa DSN de Usuario presionamos el botón *Agregar*. Luego seleccionamos *Microsoft Access Driver (*.mdb)* y presionamos *Finalizar*. Ahora se hará la conexión ODBC. Presionamos el botón *Seleccionar* y elegimos nuestra Base de Datos e ingresamos el nombre de la base en el primer campo.

Por último el botón Aceptar

Si todo salio bien debería aparecer el nombre de nuestra Base de Datos en la solapa DSN de usuario y ya tendremos hecha nuestra conexión ODBC a BS.

Ahora debemos conectar la base de datos en la pagina ASP

```
<%
'Definimos la variable para la conexión.
Dim Conex
Set Conex = Server.CreateObject ("ADODB.Connection")
'y ya estamos conectados a nuestra base de datos.
Conex.Open "nombre de la BD"
'aqui abrimos la tabla. ...
%>
```

Figura 3.0. Ejemplo de conexión ODBC.

1.5 MÓDULO V FUNDAMENTOS DE SISTEMAS OPERATIVOS

1.5.1 Conceptos básicos

1.5.2 Componentes principales de una computadora

EL núcleo de todos los sistemas de computadoras es el hardware que trabaja con el software del sistema para desempeñar varias tareas.

El hardware de la computadora esta formado de varios componentes diferentes, como el CPU, memoria y disco, cada una con un propósito especifico. Para que estos componentes trabajen juntos como un equipo, ellos requieren ser administrados por un sistema operativo.

El sistema operativo es una colección de programas y archivos con la función primaria de dar instrucciones a la computadora que hacer con el hardware.

1.5.3 Hardware

Los principales cuatro componentes de hardware de una computadora es la memoria RAM, el CPU, los dispositivos de entrada y salida, y el disco duro u otros dispositivos de almacenamiento.

1.5.4 Memoria RAM

La memoria RAM es la memoria principal de la computadora, también conocida como memoria física. Los programas y datos deben ser cargados dentro de la memoria física para que el sistema las procese. El enunciado “El sistema tiene 1GB de memoria” se refiere a la cantidad de memoria física o RAM que el sistema tiene instalada.

Un programa de software reside en el disco duro y cuando es activado, una imagen o copia del programa es cargado en la memoria RAM.

El programa se queda en la RAM el tiempo que sea necesario. Cuando el programa no es requerido por un tiempo, se pueden sobrescribir copias de otros programas. Si el sistema reinicia o experimenta una baja de energía, los datos de la memoria física son borrados.

1.5.5 CPU

La unidad central de procesamiento es el chip lógico de la computadora que ejecuta las instrucciones recibidas de la memoria física, Esas instrucciones son almacenadas en lenguaje binario.

1.5.6 Dispositivos de entrada y salida

Los componentes de entrada y salida leen las entradas de un dispositivo como el teclado a la memoria, y escribe la salida de la memoria al dispositivo.
Los dispositivos de entrada incluyen el teclado y el ratón, El monitor, la impresora y los dispositivos de cinta son los dispositivos de salida primarios.

1.5.7 Disco duro

El disco duro es un dispositivo de almacenamiento magnético en el cual los archivos, directorios y las aplicaciones son almacenadas.

1.5.8 Sistema operativo

El sistema operativo interpreta las instrucciones del usuario o de las aplicaciones y dice a la computadora que hacer. Manipula las entradas y las salidas, guarda el seguimiento de los datos almacenados en el disco y se comunica con periféricos como el monitor, disco duro, impresoras, modems, etc.

Los tres principales componentes del sistema operativo Linux son:

- El kernel
- El Shell
- El árbol de directorios

1.5.9 El kernel

El kernel es el núcleo del sistema operativo. Es el programa maestro que administra todos los recursos de la computadora, incluyendo:

- El sistema de archivos
- Administración de dispositivos
- Administración de procesos
- Administración de la memoria

1.5.10 El shell

El shell es una interfaz entre el usuario y el kernel. La función primaria de el shell es ser un intérprete de instrucciones o comandos. Esto es, el shell acepta las instrucciones que se ingresan, los interpreta y después las ejecuta. El shell por default en Linux es el BASH (Bourne Again Shell).

1.5.11 Árbol de directorios

El árbol de directorios se refiere al conjunto de directorios que donde se almacena toda la información del sistema operativo.

1.5.12 Responsabilidades del administrador

El administrador del sistema es el responsable de que las operaciones diarias del sistema se encuentren funcionando, por lo que es necesario que el administrador domine las tareas básicas de administración. Entre las tareas mas comunes tenemos las siguientes:

- Administración de cuentas de usuario y grupos.
- Mantenimiento de la seguridad del sistema.
- Configurar dispositivos.
- Instalar y particionar unidades de disco.
- Manejar y administrar sistemas de archivos.
- Calendarizar tareas relacionadas al sistema.
- Configurar archivos de inicialización del sistema.
- Instalación del sistema operativo.
- Administrar paquetes de software y parches.
- Realizar tareas de respaldo y recuperación.
- Administración de la recuperación del sistema.

1.5.13 Administración de Windows 2000

Windows 2000 es un sistema operativo con varios propósitos, con un soporte integrado para cliente/servidor y redes parejas. Se ha diseñado la familia de productos Windows 2000 para aumentar la fiabilidad, proporcionar mayores niveles de disponibilidad del sistema y conseguir dimensionabilidad de una pequeña red a una gran red entre empresas. Windows 2000 incorpora tecnologías que reducen el costo total de licencia permitiendo a las organizaciones aumentar el valor de sus inversiones existentes mientras disminuyen los costes totales de informática. Además, Windows 2000 incorpora un amplio soporte de Internet y aplicaciones, y ha sido construido a partir del éxito conseguido con Windows NT como un sistema operativo servidor para aplicaciones a Internet.

La familia de Windows 2000 estaba formada por varias versiones del sistema: una para las estaciones de trabajo (Windows 2000 Professional) y varias para servidores (Windows 2000 server, advanced server, datacenter server).

1.5.14 Características de Windows 2000

Windows 2000 incorporaba importantes innovaciones tecnológicas para entornos Microsoft, tanto en nuevos servicios como en la mejora de los existentes. Algunas de las características que posee son:

- Almacenamiento:
- Soporte para FAT16, FAT32 y NTFS.
- Cifrado de ficheros (EFS).
- Servicio de indexación.
- Sistema de archivos distribuido (DFS).
- Nuevo sistema de backup (ASR).
- Sistema de tolerancia a fallos (RAID) con discos dinámicos (software).
- Comunicaciones:
- Servicios de acceso remoto (RAS, VPN, RADIUS y Enrutamiento).
- Nueva versión de IIS con soporte para HTTP/1.1.
- Active Directory.
- Balanceo de carga (clustering)
- Servicios de instalación desatendida por red (RIS).
- Servicios nativos de Terminal Server.

1.5.15 Administración Unix y Linux

Al igual que otros sistemas operativos, el sistema operativo UNIX es un conjunto de programas de utilidad y un conjunto de instrumentos que permiten al usuario conectar y utilizar esas utilidades para construir sistemas y aplicaciones.

Al conjunto de programas que componen unix y que se encargan de proporcionar los recursos del sistema y de coordinar todos los detalles internos de la computadora se les llama en conjunto sistema operativo o kernel.

UNIX se caracteriza por ser un sistema "multiusuario" porque permite que dos o más personas utilicen la computadora al mismo tiempo.

Los Usuarios se comunican con el Kernel a través de otro programa conocido como el shell. El shell es un "*Intérprete de Línea de Comandos*" que traduce los comandos tecleados por el usuario y los convierte en instrucciones que puede entender el Kernel.

1.5.16 Características del Sistema Operativo UNIX

Los siguientes conceptos son comunes para todos los sistemas UNIX, por lo cual se puede afirmar que éstos componen las características principales de UNIX.

Kernel: Este es el componente principal del sistema operativo. Se encarga de asignar tareas y manejar el almacenamiento de datos. El usuario rara vez opera directamente con el kernel, que es la parte residente en memoria del sistema operativo.

Shell: Esta es la utilidad que procesa las peticiones de los usuarios. Cuando alguien teclea un comando en la terminal, el shell interpreta el comando y llama el programa deseado. También es un lenguaje de programación de alto nivel que puede utilizarse en la combinación de programas de utilidad para crear aplicaciones completas.
El shell puede soportar múltiples usuarios, múltiples tareas, y múltiples interfaces para sí mismo. Los dos shells más populares son el BourneShell (System V) y el Cshell (BSD Unix), debido a que usuarios diferentes pueden usar diferentes shells al mismo tiempo, entonces el sistema puede aparecer diferente para usuarios diferentes. Existe otro shell conocido como KornShell (así llamado en honor de su diseñador), que es muy popular entre los programadores.

1.5.17 Programas de utilidad (Utilerías)

El Sistema Operativo UNIX incluye una gran variedad de programas de utilidad que pueden ser fácilmente adaptadas para realizar tareas específicas. Estas utilerías son flexibles, adaptables, portables y modulares, y pueden ser usadas junto con filtros y redireccionamientos para hacerlos más poderosos.

1.5.18 Sistema multiusuarios

Dependiendo del equipo disponible, un UNIX puede soportar desde uno hasta más de 100 usuarios, ejecutando cada uno de ellos un conjunto diferente de programas.

1.6 MÓDULO VI ADMNISTRACIÓN DE BASE DE DATOS

1.6.1 Administrador de la base de datos

Es la persona encargada de definir y controlar las bases de datos corporativas, además proporciona asesoría a los desarrolladores, usuarios y ejecutivos que la requieran. Es la persona o equipo de personas profesionales responsables del control y manejo del sistema de base de datos, generalmente tiene(n) experiencia en SGBD, diseño de bases de datos, Sistemas operativos, comunicación de datos, hardware y programación.

Un administrador de base de datos de tiempo completo normalmente tiene aptitudes técnicas para el manejo del sistema en cuestión a demás, son cualidades deseables nociones de administración, manejo de personal e incluso un cierto grado de diplomacia. La característica más importante que debe poseer es un conocimiento profundo de las políticas y normas de la empresa, así como el criterio de la empresa para aplicarlas en un momento dado. La responsabilidad general del DBA es facilitar el desarrollo y el uso de la Base de Datos dentro de las guías de acción definidas por la administración de los datos.

El administrador de bases de datos es responsable primordialmente de:

- Administrar la estructura de la Base de Datos.
- Administrar la actividad de los datos.
- Administrar el Sistema Manejador de Base de Datos.
- Establecer el Diccionario de Datos.
- Asegurar la confiabilidad de la Base de Datos.
- Confirmar la seguridad de la Base de Datos.
- Administrar la estructura de la Base de Datos.

Esta responsabilidad incluye participar en el diseño inicial de la base de datos y su puesta en practica así como controlar, y administrar sus requerimientos, ayudando a evaluar alternativas, incluyendo los SGBD a utilizar y ayudando en el diseño general de la bases de datos. En los casos de grandes aplicaciones de tipo organizacional, el DBA es un gerente que supervisa el trabajo del personal de diseño de la bd.

Una vez diseñada las bases de datos, es puesta en práctica utilizando productos del SGBD, procediéndose entonces a la creación de los datos (captura inicial). El DBA participa en el desarrollo de procedimientos y controles para asegurar la calidad y la alta integridad de la bd.

Los requerimientos de los usuarios van modificándose, estos encuentran nuevas formas o métodos para lograr sus objetivos; la tecnología de la BD se va modificando y los fabricantes del SGBD actualizan sus productos. Todas las modificaciones en las estructuras o procedimientos de bd requieren de una cuidadosa administración.

16.2 Administración de la actividad de datos

El DBA no es usuario del sistema, no administra valores de datos; sino la actividad de datos; protege los datos, no los procesa. Dado que la base de datos es un recurso compartido, el DBA debe proporcionar estándares, guías de acción, procedimientos de control y la documentación necesaria para garantizar que los usuarios trabajen en forma cooperativa y complementaria al procesar datos en la bases de datos.

1.6.3 Administrar el sistema manejador de base de datos

Existe una gran actividad al interior de un SGBD. La concurrencia de múltiples usuarios requiere la estandarización de los procesos de operación; el DBA es responsable de éstas especificaciones y de asegurarse que estas lleguen a quienes concierne. Todo el ámbito de la base de datos se rige por estándares, desde la forma de como se captura la información (tipo de dato, longitud, formato), como es procesada y presentada. El nivel de estandarización alcanza hasta los aspectos más internos de la base de datos; como sé accesa a un archivo, como se determinan los índices primarios y auxiliares, registros, etc.

El DBA debe procurar siempre que los estándares que serán aplicados beneficien también a los usuarios, privilegiando siempre la optimización en la operación del SGBD y el apego de las políticas de la empresa. Entre las funciones del DBA se encuentra la de revisar los estándares periódicamente para determinar su operatividad, ajustarlos, ampliarlos o cancelarlos y hacer que éstos se cumplan.

1.6.4 Establecer el diccionario de datos

Cuando se definen estándares sobre la estructura de la base de datos, se deben de registrarse en una sección del diccionario de datos a la que todos aquellos usuarios relacionados con ese tipo de proceso pueden acceder. Este metadato debe precisar información que nos indique con claridad el tipo de datos que serán utilizados, sus ámbitos de influencia y sus limitantes de seguridad.

1.6.5 Asegurar la confiabilidad de la base de datos

Se trata de realizar un sistema de bases de datos lo suficientemente robusto para que sea capaz de recuperarse frente a errores o usos inadecuados. Se deben utilizar gestores con las herramientas necesarias para la reparación de los posibles errores que las bases de datos pueden sufrir, por ejemplo tras un corte inesperado de luz.

1.6.6 Confirmar la seguridad de la base de datos

Coordinar las nuevas propuestas para realizar ajustes en los derechos de acceso a datos compartidos y aplicaciones específicamente propuestas serían analizados en conjunto con los supervisores o directivos de las áreas involucradas para determinar si procede pudieran aparecer problemas cuando dos o más grupos de usuarios quedan autorizados para notificar los mismos datos. Uno de tales conflictos es el de la actualización perdida; este ocurre cuando el trabajo de un usuario queda sobrescrito sobre por el de un segundo usuario. El DBA queda responsabilizado para identificar la posible ocurrencia de dichos problemas así como de crear normas y procedimientos para su eliminación. Se obtendrán este tipo de garantías cuando el SGBD sea capaz de implementar las restricciones aplicables al acceso concurrente, y este sea utilizado adecuadamente por programadores y usuarios; para borrar lo anterior, se hace indispensable el apego a los estándares el seguimiento de instructivos y manuales y las reglas establecidas para los diversos procesamientos y procedimientos que se llevan acabo.

Entre las alternativas mas utilizadas por el DBA para tratar de resolver o minimizar este problema se encuentran las siguientes:

- Restringir el acceso a los procedimientos para ciertos usuarios.
- Restringir al acceso a los datos para ciertos usuarios procedimientos y/o datos.
- Evitar la coincidencia de horarios para usuarios que comparten procedimientos.

Las técnicas de recuperación son otra función esencial del DBA al administrar la actividad de datos. A pesar de que el SGBD lleva a cabo una parte del proceso de recuperación, los usuarios determinan en forma critica la operatividad de esos sistemas de protección. El DBA debe anticipar fallas y definir procedimientos estándares de operación; los usuarios deben saber que hacer cuando el sistema este caído y que es lo primero que debe realizarse cuando el sistema este puesto en marcha nuevamente. El personal de operación deberá saber como iniciar el proceso de recuperación de la BD que copias de seguridad utilizar; como programar la reejecución del tiempo perdido y de las tareas pendientes; es importante también establecer un calendario para llevar a cabo estas actividades sin afectar a otros sistemas dentro de la organización que hagan uso de los mismos recursos de computo. Destacan por su importancia en el proceso de recuperación y a su vez en la atención que prestan a otros sectores de la organización. Los dispositivos de comunicación remota, los sistemas de interconexión y otros accesorios de uso compartido.

El DBA es el responsable de la publicación y mantenimiento de la documentación en relación con la actividad de los datos, incluyendo los estándares de la BD, los derechos de recuperación y de acceso a la BD, los estándares para la recuperación de caídas y el cumplimiento de las políticas establecidas. Los productos SGBD más populares que se encuentran en el mercado proporcionan servicios de utilerías para ayudar al DBA en la administración de los datos y su actividad. Algunos sistemas registran en forma automática los nombres de los usuarios y de las aplicaciones a las que tienen acceso así como a otros objetos de la BD. Incorpora también utilerías que permitan definir en el diccionario de datos las restricciones para que determinadas aplicaciones o módulos de ellas solo tengan acceso a segmentos específicos de la BD.

1.6.7 Objetivos del administrador de la base de datos

Mantener la Integridad de los Datos. Una base de datos debe protegerse de accidentes tales como los errores en la entrada de los datos o en la programación, del uso mal intencionado de la base de datos y de los fallos del hardware o del software que corrompen los datos. La protección contra accidentes, que ocasiona inexactitudes en los datos, es parte del objetivo de garantizar la integridad de los datos. Estos accidentes incluyen los fallos durante el procesamiento de las transacciones, los errores lógicos que infringen la suposición de que las transacciones preservan las restricciones de consistencia de la base de datos y las anomalías debido al acceso concurrente en la base de datos (acceso concurrente). La integridad, se encarga de asegurar que las operaciones ejecutadas por los usuarios sean correctas y mantengan la consistencia de la base de datos.

Mantener la Seguridad de los Datos. La protección de la base de datos de usos mal intencionados o no autorizados se denomina seguridad de los datos. La seguridad se encarga de limitar a los usuarios a ejecutar únicamente las operaciones permitidas.

Mantener la Disponibilidad de los Datos. La posibilidad de fallos de hardware o de software requiere procedimientos de recuperación de la base de datos. Tiene que proporcionar medios para el restablecimiento de las bases de datos que se hayan corrompido por desperfectos del sistema, a un estado uniforme.

1.6.8 Funciones básicas del administrador de la bases de datos

Creación de Bases de Datos y Tablas

Creando Bases de Datos:

- Localización de las bases de datos.

- Tipo de base de datos (modo de direccionamiento).

Creando tablas:

- Seleccionando tipos de datos.
- Tablas fragmentadas o no fragmentadas.
- Localización de la tabla.
- Determinación del espacio en disco.
- Modo de aseguramiento de candados.

Especificación de las restricciones de Integridad de los datos. Las restricciones de integridad se mantienen en una estructura especial del sistema que consulta el gestor de la base de datos cada vez que se tiene lugar una actualización en el sistema. Estos son algunos métodos para asegurar la integridad de los datos:

Privilegios:

- Base de datos.
- Tabla.
- Columna

Integridad de identidad, semántica, referencial y de negocio.
Vistas

Administrar la concurrencia. La administración de la concurrencia involucra como los datos son consultados y actualizados en un ambiente multiusuario. Existen dos tipos de control de la concurrencia:

- Concurrencia de Lectura: (Instrucción SELECT)
- Administrada a través de los niveles de aislamiento.
- Concurrencia de Actualización: Instrucciones INSERT, DELETE y UPDATE.

Optimización del Acceso a Datos

- Índices.
- Estadísticas de actualización.
- Distribución de datos.

Definir el Esquema Conceptual. Es tarea del administrador de datos decidir con exactitud cual es la información que debe mantenerse en la base de datos, una vez identificado los datos a almacenar en un nivel abstracto, el dba debe crear a continuación el esquema conceptual correspondiente, empleando el DDL conceptual.

Definir el Esquema Interno. El dba debe definir la representación de la información en la base de datos almacenada (diseño físico). Debe crear la definición de estructura de almacenamiento correspondiente (esquema interno) con el DDL interno y definir la correspondencia entre los esquemas interno y conceptual.

Vincularse con los Usuarios. El dba debe encargarse de la comunicación con los usuarios, garantizar la disponibilidad de los datos que requieren y escribir y/o ayudar a los usuarios a escribir los esquemas externos necesarios, empleando el DDL externo aplicable.

Procedimientos de Respaldo y Recuperación. El dba debe definir un plan de recuperación adecuado que incluya descarga o vaciado periódico de la base de datos en un medio de almacenamiento de respaldo, y procedimientos para cargar otra vez la base de datos a partir del vaciado más reciente cuando sea necesario.

Supervisar el Desempeño y Responder a cambios en los Requerimientos. El dba debe organizar el sistema de modo que se obtenga el desempeño que sea "el mejor para la empresa", y realizar los ajustes apropiados cuando cambien los requerimientos.

Concesión de Autorización para el Acceso a los Datos. La concesión de diferentes tipos de autorización, permite al administrador de la base de datos regular que partes de la base de datos van a poder ser accedidas por varios usuarios.

Definición de esquema. Es el esquema original de la base de datos se crea escribiendo un conjunto de definiciones que son traducidas por el compilador de DDL a un conjunto de tablas que son almacenadas permanentemente en el diccionario de datos.

Definición de la estructura de almacenamiento del método de acceso. Estructuras de almacenamiento y de acceso adecuados se crean escribiendo un conjunto de definiciones que son traducidas por e compilador del lenguaje de almacenamiento y definición de datos.

1.6.9 Funciones específicas del SGBD

El sistema manejador de bases de datos es la porción más importante del software de un sistema de base de datos. Un SGBD es una colección de numerosas rutinas de software interrelacionadas, cada una de las cuales es responsable de alguna tarea específica.

A demás de administrar la actividad de datos y la estructura de la base de datos, el DBA debe administrar el SGBD mismo. Deberá compilar y analizar estadísticas relativas al rendimiento del sistema e identificar áreas potenciales del problema. Dado que la BD esta sirviendo a muchos grupos de usuarios, el DBA requiere investigar todas las quejas sobre el tiempo de respuesta del sistema, la precisión de los datos y la facilidad de uso. Si se requieren cambios el DBA deberá planearlos y ponerlos en práctica.

El DBA deberá vigilar periódica y continuamente las actividades de los usuarios en la base de datos. Los productos SGBD incluyen tecnologías que reúnen y publican estadísticas. Estos informes pudieran indicar cuales fueron los usuarios activos, que archivos y que elementos de datos han sido utilizados, e incluso el método de acceso que se ha aplicado. Pueden capturarse y reportarse las tasas de error y los tipos de errores. El DBA analizará estos datos para determinar si se necesita una modificación en el diseño de la BD para manejar su rendimiento o para facilitar las tareas de los usuarios; de ser así, el DBA la llevará a cabo.

El DBA deberá analizar las estadísticas de tiempo de ejecución sobre la actividad de la BD y su rendimiento. Cuando se identifique un problema de rendimiento, ya sea mediante una queja o un informe, el DBA deberá determinar si resulta apropiada una modificación a la estructura de la base de datos o al sistema. Casos como la adición de nuevas claves o su eliminación, nuevas relaciones entre los datos y otras situaciones típicas deberán ser analizadas para determinar el tipo de modificación procedente.

Cuando el fabricante del SGBD en uso anuncie una nueva versión del producto, debe realizarse un análisis de las características que esta incorpora e insopesarlas contra las necesidades de la comunidad de usuarios. Si se decide la adquisición del producto, los usuarios deben ser notificados

y capacitados en su uso. El DBA deberá administrar y controlar la migración tanto de las estructuras, como de los datos y las aplicaciones. El software de soporte y otras características de hardware pueden implicar también modificaciones de las que el DBA es responsable ocasionalmente, estas modificaciones traen como consecuencia cambios en la configuración o en algunos parámetros de operación del SGBD.

1.7 Módulo VII HABILIDADES DIRECTIVAS

1.7.1 Liderazgo para el cambio

¿Para qué las habilidades directivas?

Las personas dedicadas a diversos aspectos relacionados con la computación deben tener claro que su trabajo forma parte de un sistema que integra recursos técnicos, humanos y materiales destinados a lograr objetivos comunes.

Administrar es alcanzar un objetivo mediante el esfuerzo humano coordinado, de ahí la necesidad de desarrollar la habilidad para comunicarse con las personas con las que colaboramos, comprender las necesidades del grupo en lo individual y lo general, descubrir las posibilidades técnicas y humanas para obtener lo mejor de cada persona y el mejor funcionamiento grupal.

Diferencia entre jefe y líder

Jefe:

- Autoridad formal de arriba abajo
- Relación rol a rol
- Centrado en la tarea
- Atiende las necesidades de la organización

Líder:

- Autoridad de abajo arriba
- Relación de persona
- Centrado en los procesos socioemociónales
- Atiende las necesidades de las personas y del grupo

Diferencias entre líder y administrador

Administrador:

- Administra
- Copia
- Mantiene
- Manda
- Acepta la realidad
- Enfoque a sistemas y estructuras
- Control
- Cómo y cuando
- Hace las cosas bien
- Da la horas

Líder:

- Innova
- Crea
- Desarrolla
- Convence
- Investiga la realidad
- Enfoque en la gente

- Confianza
- Que y por que
- Hace las cosas correctas
- Construye relojes

Pasar de jefe a líder significa reconocer tres áreas de resultados igualmente importantes:

Tareas: Donde se trata de lograr resultados específicos, en la cantidad, con calidad y la oportunidad requeridas.

El reto actual: Asegurar una mayor productividad y calidad.

Grupos: Donde se requiere ocuparse de los individuos a su cargo para asegurar su capacitación, motivación y progreso.

El reto actual: Promover una mayor preparación, involucramiento y responsabilidad.

Equipos: Donde debe asegurarse su integración y desarrollo, así como un clima de comunicación y colaboración.

El reto actual: Es conseguir una mayor participación y trabajo en equipo.

El líder debe trabajar con un enfoque de desarrollo:

Inspirar confianza en su gente: Cambiar la actitud tradicional de intervenir para dar instrucciones y corregir, a otra capaz de inspirar una visión común, modelar con el ejemplo, reconocer y recompensar todos los esfuerzos, son las únicas vías para ganar la confianza de su gente y por tanto, reforzar consistentemente el cambio.

Centrarse en las personas: Olvidar la tendencia de considerar a las personas como cosas o simplemente como recursos humanos y responder a las necesidades que cada individuo tiene como ser humano.

Desarrollar un equipo: Derribar las barreras entre funciones y personas, trabajar de manera interdisciplinaria, aumentar la participación y ser capaz de utilizar y fomentar los mejores recursos del grupo, son habilidades indispensables de un líder empeñado en mejorar la calidad y productividad.

Generar un clima motivador: Los procesos de calidad y el buen servicio se basan en la participación de la gente y estimular su iniciativa para que actúen con responsabilidad sin necesidad de vigilancia.

1.7.3 Liderazgo

Entonces, podemos señalar que **Liderazgo** es el proceso de influir, guiar o dirigir a los miembros del grupo hacia el éxito en la consecución de metas y objetivos organizacionales.

1.7.4 El cambio

El cambio es una modificación o movimiento de un plano o estado a otro que implica dejar a un lado determinadas estructuras, procedimientos, comportamientos, etc., para adquirir otras que permitan la adaptación al contexto en el cual se encuentra el sistema, y así lograr una estabilidad que facilite la eficacia y efectividad en la ejecución de acciones.

Existen tres etapas esenciales y secuenciales que facilitan el proceso de cambio de sistemas:

Descongelamiento: Implica desequilibrio, insatisfacción, toma de conciencia de a situación.
Se perciben procedimientos, hábitos, costumbres, actitudes que obstaculizan la adaptación. Genera ansiedad y dudas del propio modo de conducirse.

Se tiene la necesidad de identificar las estructuras sujetas al cambio, satisfacer nuevas necesidades, equilibrio y logro de una situación deseada.

Movimiento: Genera desequilibrio, inestructura, inestabilidad, inseguridad, incertidumbre. Se tiene la necesidad de voltear la mirada al entorno, generar información, buscar alternativas, seleccionar alternativas, abandonar viejas estructuras o esquemas. Se requiere mayor adaptación y adoptar nuevos esquemas y estructuras.

Recongelamiento: Se establece claridad en la situación, se llega a un equilibrio y se tiene mayor adaptabilidad. Esto implica integrar nuevos esquemas, establecer contacto genuino con la opción elegida, considerar el efecto del cambio en el resto de los subsistemas y una cierta permanencia en esta nueva situación.

Ejemplo de un Archivo de Configuración

El cambio puede generar resistencia.

Esta resistencia al cambio es una reacción esperada por parte del sistema, el cual estando en un periodo de equilibrio, percibe la amenaza de la inestabilidad e incertidumbre que acarrean consigo las modificaciones. Por tanto, se puede definir como las fuerzas restrictivas que obstaculizan un cambio.

Los hábitos: Son un obstáculo por el grado de arraigo que los caracteriza y porque resultan una medida de economía, ya que al aplicarlos nos evitamos reflexionar en cada situación, de tal forma que un cambio de hábito implica mayor inversión de energía, o sea, llevar a cabo un esfuerzo adicional.

Miedo a lo desconocido: El mañana no está aquí, por lo tanto resulta un misterio, una fantasía; de tal modo que muchas personas prefieren no enfrentar el riesgo de encontrar sorpresas, sean éstas buenas o malas, por lo que se inclinan a permanecer en el lugar en donde están hoy.

Apego a lo conocido: “Más vale malo por conocido que bueno por conocer”. Una vez vivenciado el éxito que se obtiene con determinada acción, se convierte en hábito y se instala dentro de los modelos típicos de comportamiento.

Tendencia a conservar la estabilidad: Existe una gran tendencia a mantener un ambiente predecible, estructurado

Y seguro, aunque no podemos negar la necesidad de explorar y arriesgar; sin embargo, se puede afirmar que entre más se aferre el individuo a sus modelos de comportamiento se resistirá al cambio.

Apego a lo elaborado por el individuo mismo: Cuando un sujeto es el autor de determinada situación, el cambio se convierte en un desprestigio y poca valoración de su esfuerzo.

¿Cómo disminuir la resistencia al cambio?

- Escuchar las expresiones de resistencia y manifestar empatía.
- Generar información de hechos, necesidades, objetivos y efectos del cambio.
- Ajustar el modo de implantación del cambio a las características de la organización.
- Reducir incertidumbre e inseguridad.
- Buscar apoyos que fomenten la credibilidad.
- No combatir la resistencia, es sólo un síntoma… hay que buscar la raíz.
- No imponer el cambio.
- Hacer un cambio participativo.
- Establecer el diálogo e intercambiar y confrontar percepciones y opiniones.
- Plantear problemas, no soluciones unilaterales.
- Realizar cambios continuamente, aun cuando sean pequeños.
- Crear un compromiso común.
- Plantear el costo – beneficio del cambio.

1.7.5 Presentaciones Efectivas

Comunicación

Se ha señalado que la opinión que tenemos de los demás suele basarse en tres características principales:

- Contenido verbal = 7
- Interés del discurso oral =38
- Lenguaje corporal = 55

1.7.7 Lenguaje corporal

Quieto en un punto determinado, inclinado hacia un lado.
Mensaje oculto: Estoy aburrido y preferiría estar en otra parte.
Solución: Cuando esté quieto, equilibre su peso y nivele las caderas.
Inclinado encima del atril.

Mensaje oculto: Estoy demasiado cansado para mantenerme de pie, me da pereza hacer esto.
Solución: Cuando utilice un atril, mejor colóquese a un lado de éste.

Sentado encima de la mesa con las notas, el retroproyector, etc.

Mensaje oculto: Aquí no tengo que esforzarme, porque soy más importante que los demás.

Solución: No importa lo relajado que se sienta, ¡póngase de pie!.

En definitiva, no existe ninguna postura ideal. Por tanto, haga lo que le parezca bien, dentro de lo razonable.

1.7.8 El Movimiento

- No cerrar las manos delante de usted
- No colocar las manos tras la espalda
- No cerrar los puños a menos que quiera generar un efecto
- No cruzar los brazos
- No jugar con objetos

1.7.9 Reuniones Efectivas

Administrar el tiempo

Asumir el control de los requerimientos que te hacen en tu tiempo disponible.

Asegurar que el uso que haces de tu tiempo – una distribución limitada – se ajuste de la mejor manera a tus metas y necesidades personales.

Establecer tus metas personales. Si no están claras, no tienes un marco de referencia para poder distribuir el tiempo.

1.8 MÓDULO VIII SEGURIDAD EN BASE DE DATOS

Hoy en día se considera a la información de una empresa como uno de los activos más valiosos, por lo que la seguridad de la misma es muy importante.

Un sistema de cómputo es seguro si se puede confiar en que él y su software se comportarán como se espera que lo hagan, y que la información almacenada en él se mantendrá inalterada y accesible durante el tiempo que quiera.

La seguridad implica asegurar que los usuarios están autorizados para llevar acabo lo que tratan de hacer.

La seguridad de una base de datos se refiere principalmente al control de acceso, modificación y definición, tanto de los datos como de la estructura de la base de datos por parte de los diferentes usuarios de la misma.

Algunos sistemas operativos proporcionan algún nivel de seguridad en el control de acceso a usuarios, sin embargo ésta debe radicar principalmente en el SGBD o en la aplicación que maneje la base de datos.

Muchas de las veces las personas piensan que configurar correctamente los servicios de red y corregir vulnerabilidades del SO., es suficiente para que no suceda nada con su información en el SGBD con lo que se descuida el aspecto seguridad en el manejador.

Los SGBD se direccionan por medio de direcciones IP y de puertos en el servidor, por lo que hay veces que sin necesidad de tener una cuenta en el servidor a nivel SO. se puede tener el acceso al mismo, únicamente configurando un cliente o el ODBC con estos datos y una cuenta con privilegios es más con suficiente.

A veces los fabricantes de SGBD por comodidad del usuario usan típicamente el mismo puerto y las mismas cuentas para configurar el servidor, las cuales cuando se instalan tienen el mismo password y no lo tienen.

Así por ejemplo, Sybase utiliza en las últimas versiones el puerto 4100 y en versiones viejitas (v. 10) 2025, cuando se instala el servidor de bases de datos, la cuenta de “sa” (dba) no tiene password.

Lo mismo sucede con MS SQL Server que utiliza el puerto: 1433 y tiene la misma cuenta de “sa” sin password.

En el caso de Oracle utiliza los puertos 1521 y 1526 generalmente, y algunas de sus cuentas y contraseñas son conocidas: system / manager y sys / change_on_install (cuentas / contraseña) las cuales cuentas con permisos suficientes para comprometer al sistema.

1.8.1 Seguridad del sistema operativo

1.8.2 Elementos de seguridad

Vulnerabilidades: Son los puntos débiles del sistema a través de los cuales la seguridad puede ser afectada.

- Físicas: Cualquier persona que tenga acceso físico a las instalaciones puede dañar seriamente el equipo.

- Naturales: El equipo de cómputo es especialmente sensible a cualquier alteración de su medio ambiente, por ejemplos cambios de temperatura, humedad, polvo, etc.

- Hardware y software: Un dispositivo de almacenamiento que falle o una caída del sistema pueden comprometer la integridad de los datos.

- Almacenamiento: Seguridad en cintas y discos de almacenamiento, ya que pueden ser extraviados, robados o dañados.

- De comunicación: Verificar que las transacciones sean validadas o en su caso en caso de una falla de la red sean desechas.

- Humanas: Si el administrador comete un error, realiza una acción indebida o desconoce el sistema puede comprometerse la seguridad y la integridad de los datos.

Amenazas: Es un agente hostil que accidentalmente o mediante alguna técnica especializada puede modificar o liberar la información.

- Naturales: Terremotos, tormentas, fallas en el suministro eléctrico, se deben de considerar planes de contingencia.
- No intencionales: Muchas veces por falta de cultura informática (desconocimiento del usuario del sistema).
- Intencionales:

Externas: Ataques de hackers, intrusiones, alteración del sistema, comprometimiento de información importante.

Internas: El 80% de los ataques vienen de dentro de la organización.

Contramedidas:

- Seguridad Interna: Seguridad en el SO., aplicación de políticas de seguridad para usuarios y personal de la empresa.
- Seguridad en comunicaciones: Utilizar clientes seguros, proxys, firewalls, etc.
- Seguridad física: Instalar detectores de humo o alarmas, sistemas de aire acondicionado, etc.
- Seguridad humana: Evangelizar al personal y a los usuarios.

1.8.3 Características de la seguridad

1.8.4 Confidencialidad

Un sistema manejador de base de datos no debe permitir que la información que esta contenida en él sea accesible a nadie que no tenga autorización adecuada.

1.8.5 Integridad y Autenticidad

El sistema debe de ser capaz de impedir los cambios a los datos ya sea por errores de hardware, software y/o personas no autorizadas, además de contar con los medios para verificar que los datos contenidos no son modificados ilegalmente.

1.8.6 Disponibilidad

Significa que tanto el Software, hardware, y sobre todo la información, se mantendrán disponibles y funcionando en forma eficiente para uso de los usuarios y que el sistema sea capaz de recuperarse rápidamente a problemas.

1.8.7 Esquema de seguridad en el SGBD

Dentro del Sistema Manejador de Base de Datos (SGBD) podemos encontrar un acceso multicapas, como el que se muestra a continuación:

- El usuario final debe tener una cuenta válida dentro de la capa del servidor (SGBD).
- El usuario final debe ser un usuario válido dentro de la capa de la base de datos.
- El usuario final deberá tener permiso dentro de la capa de los datos.

Figura 4.0. Primer capa de seguridad en un SGBD

Dar de alta un login y darle permisos de entrada al SGBD.

- sp_addlogin (Sybase)
- Create user (oracle)
- Grant create session

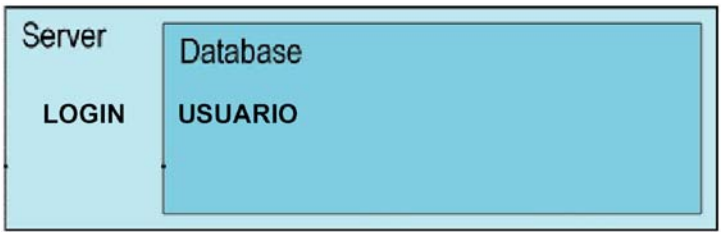

Figura 4.1. Segunda capa de seguridad de un SGBD

Dar de alta un login como usuario de la base de datos en particular.

- sp_adduser (Sybase)
- Grant create session (oracle)

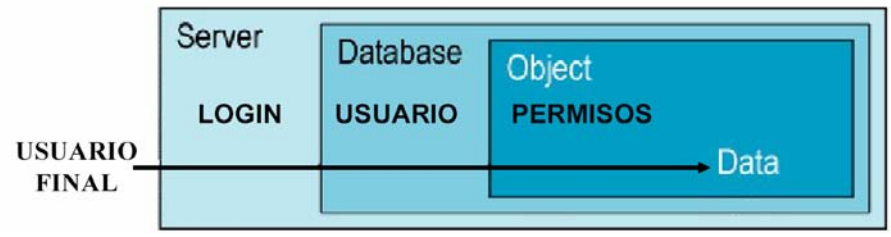

Figura 4.2. Tercer capa de seguridad de un SGBD.

Dar permiso para que use el objeto en particular.

- Grant select, insert, update, delete on [tabla|vista] to [user|grupo].
- Grant select on sueldos to gerente
- Revoke select on sueldos.gratificaciones to jefes_dep

Los grupos de trabajo pueden ser alternativas para controlar los permisos.

Lo que a nivel de usuario se traduciría en lo siguiente:

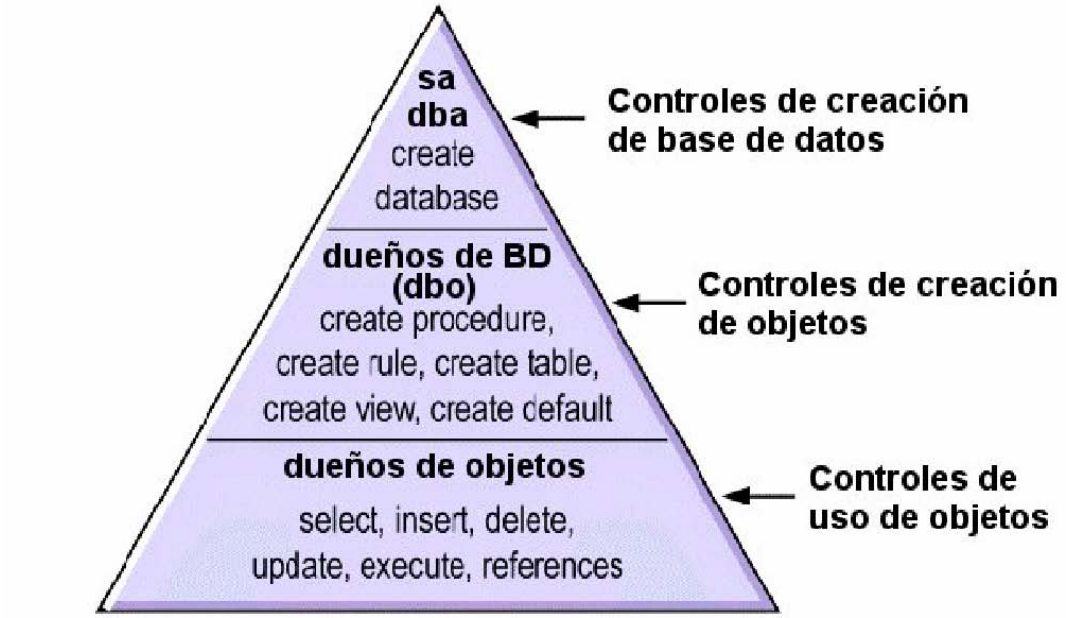

Figura 4.3. Grupos de trabajo para controlar permisos.

A nivel de objetos hay que recordar que:

Cada objeto (tabla, vista, procedimiento almacenado) tiene un dueño y que esta propiedad es intransferible.

Este dueño puede alterar o borrar el objeto, si este es una tabla, podrá crear o borrar índices y triggers sobre la tabla.

Además podrá otorgar o revocar permisos sobre sus objetos a usuarios, grupos y roles (perfiles). A su vez podrá determinar si los demás usuarios serán capaces de otorgarselos a otros usuarios (lo cual no es recomendable).

Además que, se recomienda que el dueño de los objetos de la base, sea un sólo usuario: el dbo, el que realice la generación de objetos.

El usuario que no es el dbo o el dueño del objeto, necesitará permisos para realizar algunas de estas acciones sobre objetos:

- Tablas: select, insert, delete, update.
- Columnas en tablas: select, update.
- Vistas: select, insert, delete, update
- Procedimientos almacenados: execute.

El usuario no requerirá permisos para utilizar los siguientes objetos de la base:

- Defaults.
- Reglas.
- Índices y triggers donde el usuario tenga permisos de uso.

Cualquier usuario podrá usar sin el dbo o el dueño, lo siguiente:

- Tipos de datos definidos por usuario.
- Cursores.

1.9 MÓDULO IX PERFORMANCE & TUNNING

1.9.1 ¿Por qué importa la configuración?

Los valores de los parámetros de configuración determinan el uso de los recursos del sistema.
La configuración del servidor de base de datos afecta directamente su rendimiento.
Los valores de configuración predeterminados, son mínimos y sólo permiten arrancar el servidor de base de datos.

1.9.2 Características de los parámetros de configuración

Los administradores del sistema pueden reconfigurar muchos valores.
Los parámetros de configuración son estáticos o dinámicos.
Los parámetros dinámicos afectan inmediatamente después de ejecutarse sp_configure.
Los parámetros estáticos requieren que el servidor de base de datos reasigne memoria.
Al cambiar los parámetros estáticos se requiere reinicializar el servidor de base de datos.

1.9.3 ¿Por qué utilizar segmentos?

Cuando se añade un nuevo dispositivo a una base de datos con alter database, SQL Server lo sitúa en un espacio predeterminado (los segmentos *default* y *system* de la base de datos). Esto aumenta el espacio total disponible para la base de datos, pero no define qué objetos ocuparán el nuevo espacio. Cualquier tabla o índice podría crecer hasta llenar todo el espacio, dejando tablas cruciales sin espacio de expansión. También es posible que varias tablas e índices muy utilizados se sitúen en un solo dispositivo físico del espacio predeterminado, resultando en un rendimiento pobre de las E/S.

Los segmentos permiten que los administradores del sistema y propietarios de las bases de datos controlen la ubicación de los objetos de base de datos en los dispositivos. Cuando se crea un objeto en un segmento, el objeto puede utilizar todos los dispositivos disponibles en el segmento, pero ningún otro dispositivo. Esta función permite controlar el espacio disponible para los objetos individuales. Más importante aún, puede emplearse para mejorar el rendimiento de SQL Server.

1.9.4 Control del uso del espacio

Si se asignan objetos no cruciales a un segmento, esos objetos no pueden crecer más allá del espacio disponible en los dispositivos del segmento. A la inversa, si se asigna una tabla crucial a un segmento y los dispositivos del segmento no están disponibles para otros segmentos, ningún otro objeto competirá por espacio con esa tabla.
Cuando se llenan los dispositivos de un segmento, se puede ampliar el segmento para incluir otros dispositivos o fragmentos de dispositivo según sea necesario. Los segmentos también permiten usar umbrales para avisar cuándo escasea espacio en determinado segmento de base de datos.

1.9.5 Segmentos y umbrales

Los umbrales controlan la cantidad de espacio libre de un segmento de base de datos y, cuando el espacio se llena, pueden realizar acciones automáticamente. Cada base de datos que almacena el diario de transacciones en un dispositivo aparte de los datos, tiene al menos un umbral: el de última oportunidad. Si se crean segmentos adicionales para los datos, es posible crear procedimientos de umbral nuevos para cada segmento.

1.9.6 Mejora del rendimiento

En un entorno SQL Server grande, de varias bases de datos y/o varias unidades, la asignación cuidadosa de espacio a bases de datos y la ubicación de objetos de base de datos en dispositivos físicos puede mejorar el rendimiento del sistema. Idealmente, cada base de datos tiene un uso exclusivo de los dispositivos, es decir, no comparte un disco físico con otra base de datos. En la mayoría de los casos, el rendimiento puede mejorarse situando los objetos de base de datos más usados en discos físicos dedicados, o "dividiendo" las tablas grandes en varios discos físicos.

1.6.7 Separación de tablas, índices y diarios

En general, situar una tabla en un solo dispositivo físico, sus índices no agrupados en un segundo dispositivo físico y el diario de transacciones en un tercero puede acelerar el rendimiento. El uso de dispositivos físicos separados (controladores de disco) reduce el tiempo necesario para leer o escribir en el disco, ya que suele disminuir el desplazamiento de los cabezales del disco. Si no es posible dedicar dispositivos enteros de esta forma, al menos confine todos los índices no agrupados a un dispositivo físico dedicado.

La opción logon de create database (o sp_logdevice) maneja la ubicación del diario de transacciones en un dispositivo físico aparte. Los segmentos permiten situar tablas e índices en dispositivos físicos específico

1.9.8 División de tablas

La división de una tabla grande y muy usada entre dispositivos situados en distintos controladores de disco puede mejorar el rendimiento general de lectura de una tabla. Cuando una tabla grande existe en varios dispositivos, es más probable que se produzcan lecturas pequeñas y simultáneas en discos distintos

Una tabla puede dividirse entre varios dispositivos mediante tres métodos distintos, cada uno de los cuales requiere el uso de segmentos:

- Utilice particiones si la tabla no tiene un índice agrupado.
- Use el método de carga parcial si la tabla tiene un índice agrupado.
- Separe la cadena de texto de otros datos si la tabla contiene datos de tipo *text* o *image*.

1.9.9 Partición de tablas

La partición de una tabla crea varias cadenas de páginas para la tabla y distribuye esas cadenas entre todos los dispositivos del segmento de la tabla. La partición de una tabla aumenta el

rendimiento de las inserciones, así como el de las lecturas, ya que hay varias cadenas de páginas disponibles para las inserciones.

Antes de que pueda dividirse una tabla, es necesario crearla en un segmento que contenga el número deseado de dispositivos.

No es posible realizar particiones de tablas con índices agrupados. Carga parcial
Si se desea dividir una tabla que tiene un índice agrupado, es posible utilizar sp_placeobject con varios comandos load para cargar diferentes partes de la tabla en segmentos distintos. Este método puede ser difícil de ejecutar y mantener, pero proporciona una forma de dividir tablas y sus índices agrupados entre varios dispositivos físicos.

SQL Server almacena los datos de las columnas *text* e *image* en una cadena aparte de páginas de datos. De forma predeterminada, esta cadena de texto se sitúa en el mismo segmento que los demás datos de la tabla. Dado que leer una columna de texto requiere una operación de lectura para el puntero de texto de la tabla base y otra operación de lectura en la página de texto de la cadena de texto aparte, es posible mejorar el rendimiento situando la cadena y los datos de la tabla base en dispositivos físicos aparte.

1.9.10 Desplazamiento de una tabla a otro dispositivo

También es posible utilizar segmentos para mover una tabla de un dispositivo a otro usando el comando create clustered index. Los índices agrupados, donde el nivel inferior o de hoja del índice contiene los datos en sí, están por definición en el mismo segmento que la tabla. Por lo tanto, es posible desplazar una tabla por completo omitiendo su índice agrupado (si lo hubiera) y creando o volviendo a crear un índice agrupado en el segmento deseado.

1.9.11 Almacenamiento

SQL Server puede tomar algunas decisiones predeterminadas razonables sobre muchos aspectos del manejo del almacenamiento, como dónde situar las bases de datos, tablas e índices y cuánto espacio se asigna a cada uno. No obstante, el administrador del sistema tiene el control final sobre la asignación de recursos de disco a SQL Server y la ubicación física de las bases de datos, tablas e índices en esos recursos.

La responsabilidad de asignar y manejar el almacenamiento suele estar centralizada. No obstante, el administrador del sistema tiene un control total sobre esas materias en muchas instalaciones.

1.9.12 Asignación de dispositivos y ubicación de objetos

Al configurar un nuevo sistema, el administrador del sistema debe considerar aspectos que tienen impacto directo sobre el número y tamaño de los recursos de disco requeridos. Estos aspectos de la asignación de dispositivos hacen referencia a procedimientos y comandos que añaden recursos de disco a SQL Server

Después de asignar los recursos de disco iniciales a SQL Server, el administrador del sistema, el propietario de bases de datos y los propietarios de objetos deben considerar cómo situar las bases de datos y sus objetos en dispositivos específicos de base de datos. Estos aspectos de la

ubicación de objetos determinan dónde residen los objetos de base de datos en su sistema y si los objetos comparten dispositivos o no.

El concepto de asignar dispositivos no debe considerarse aisladamente del concepto de situar objetos. Por ejemplo, si decide que una tabla en particular debe residir en un par de dispositivos dedicados, primero debe asignarlos a SQL Server. Las secciones restantes de este capítulo proporcionan una descripción general que abarca desde la asignación de dispositivos a la ubicación de objetos, y hace referencia a otros capítulos cuando así corresponde

1.9.13 El caché de datos en SQL Server

El caché de datos contiene los datos, índices y páginas de diarios que SQL Server está usando actualmente, así como los que ha usado recientemente. Cuando se instala por primera vez, SQL Server tiene un solo caché de datos predeterminado que se utiliza para toda la actividad de diarios, datos e índices. Este caché puede dividirse creando cachés de datos con nombre. Además, dentro de los cachés con nombre y del caché predeterminado, es posible crear bancos para realizar E/S de gran tamaño. Después se puede vincular una base de datos, tabla (incluida la tabla *s yslogs*), índice, o cadena de páginas de texto o de imagen a un caché de datos con nombre.
Los tamaños grandes de E/S permiten que SQL Server realice recuperaciones previas de datos cuando el optimizador de consultas determina que la recuperación previa mejorará el rendimiento. Por ejemplo, un tamaño de E/S de 16K implica que SQL Server puede leer todo un sector, u ocho páginas de 2K, de una sola vez, en lugar de realizar ocho E/S distintas.

El proceso de configurar cachés de datos con nombre divide el caché predeterminado en estructuras separadas. Los cachés de datos con nombre creados por usted sólo pueden ser utilizados por bases de datos u objetos que se vinculen a ellos explícitamente. Los objetos que no se vinculen explícitamente usan el caché de datos predeterminado.

SQL Server proporciona cachés de datos configurables por el usuario para mejorar el rendimiento, en especial en servidores multiprocesador

1.9.14 Maximización de la memoria de SQL Server

Cuanto más memoria haya disponible, más recursos tendrá SQL Server para las memorias intermedias internas y los cachés. Disponer de suficiente memoria para los cachés reduce el número de veces que SQL Server tiene que leer información estática o planes de procedimientos compilados del disco.

No hay disminución del rendimiento por configurar SQL Server para que utilice el máximo de memoria disponible en la computadora. Sin embargo, asegúrese de que evalúa otras necesidades de memoria en el sistema y de que SQL Server utiliza sólo el resto de la memoria disponible. Es posible que SQL Server no arranque si no puede adquirir la memoria para la cual está configurado.
Para determinar la cantidad máxima de memoria disponible para SQL Server en su sistema:
Determine la cantidad total de memoria física instalada en su computadora.
Reste la memoria que precisa el sistema operativo del total de la memoria física.

Si la máquina no está dedicada a SQL Server, reste los requisitos de memoria de otros usos del sistema también. Por ejemplo, reste la memoria que utilizarán las aplicaciones cliente que se ejecutarán en la máquina de SQL Server. Los sistemas de ventanas, como X Windows, requieren

mucha memoria y pueden interferir con el rendimiento de SQL Server cuando se utilizan en la misma máquina.

Reste la memoria que desea asignar para el parámetro de configuración additional network memory.

La memoria que queda después de restar los requisitos del sistema operativo, de otras aplicaciones y de additional network memory , es la memoria total disponible para SQL Server. Configure SQL Server para que utilice esa memoria restante definiendo el parámetro total memory con ese valor.

Considere la posibilidad de cambiar el valor del parámetro de configuración total memory

- Cuando cambie la cantidad de RAM en la máquina
- Cuando cambie el tipo de uso de la máquina
- Si asigna memoria a additional network memory para SQL Server

1.10 MÓDULO X MODELO ORIENTADO A OBJETOS

1.10.1 Metodologías orientadas a objetos
1.10.2 Object Oriented Design, por Grady Booch

La metodología de Booch usa los siguientes tipos de diagramas para describir las decisiones de análisis y diseño, tácticas y estratégicas, que deben ser hechas en la creación de un sistema orientado por objetos.

- Diagrama de Clases. Consisten en un conjunto de clases y relaciones entre ellas. Puede contener clases, clases paramétricas, utilidades y metaclases. Los tipos de relaciones son asociaciones, contenencia, herencia, uso, instanciación y metaclase.
- Especificación de Clases. Es usado para capturar toda la información importante acerca de una clase en formato texto.
- Diagrama de Categorías. Muestra clases agrupadas lógicamente bajo varias categorías
- Diagramas de transición de estados.
- Diagramas de Objetos. Muestra objetos en el sistema y su relación lógica. Pueden ser diagramas de escenario, donde se muestra cómo colaboran los objetos en cierta operación; o diagramas de instancia, que muestra la existencia de los objetos y las relaciones estructurales entre ellos. Diagramas de Tiempo. Aumenta un diagrama de objetos con información acerca de eventos externos y tiempo de llegada de los mensajes.
- Diagramas de módulos. Muestra la localización de objetos y clases en módulos del diseño físico de un sistema. Un diagrama de módulos representa parte o la totalidad de la arquitectura de módulos del sistema.
- Subsistemas. Un subsistema es una agrupación de módulos, útil en modelos de gran escala. Diagramas de procesos. Muestra la localización de los procesos en los distintos procesadores de un ambiente distribuido.

1.10.3 Objectory, por Ivar Jacobson
Conceptos y diagramas

Objectory es un proceso organizado para la construcción industrial de software. Este proceso de diseño está guiado por casos de uso, una técnica que basa su centra el entendimiento de un sistema en la forma en la cual es usado.

- Modelo de Casos de Uso: Se basa en la descripción de elementos o usuarios externos al sistema (actores) y funcionalidad del sistema (casos de uso).
- Modelo de objetos: Representa la estructura estática de objetos. Puede contener objetos entidad, de interfaz y de control, entre otros tipos; y relaciones de herencia, conocido (una referencia estática), consiste de y comunicación.
- Diagrama de interacción. Muestran la secuencia de eventos entre paquetes u objetos necesarios para realizar un caso de uso.
- Diagrama de estado. Muestra los estados internos de un objeto complejo.

Algunos de los conceptos más importantes de esta metodología son:

- Objeto Entidad. Representa información del sistema que debe sobrevivir cierto período de tiempo, por ejemplo, un caso de uso
- Objeto de Interfaz. Modela información y comportamiento que es dependiente de la interfaz actual del sistema

- Objeto de Control. modela funcionalidad que no corresponde a ningún objeto en particular y que se presenta en algunos casos de uso. Estos objetos generalmente operan sobre varios objetos entidad, realizan algún algoritmo y retornan algún resultado a un objeto de interfaz.
- Paquete. Módulo que contiene código, traducible a un módulo en el lenguaje de implementación.
- Unidad. En pruebas, desde una clase hasta un subsistema.

1.10.4 Object Modeling Technique, por James Rumbaugh

Conceptos y Diagramas

OMT hace un cubrimiento de las etapas de análisis, diseño e implementación definidas por la OMG, dejando sin cubrir el modelamiento estratégico.

- Modelo de Objetos. Se define como un diagrama de objetos mas un diccionario de datos. El diagrama de objetos muestra clases y sus relaciones (generalización, agregación, asociación, instanciación). El diccionario de datos es el detalle de las clases en el diagrama de objetos
- Modelo dinámico. Se define como un conjunto de diagramas de estado mas un diagrama de Flujo de eventos Global.
- Modelo funcional. Es un diagrama de flujo con restricciones.

Estas metodologías son fusionadas para formar UML.

UML (Unified Modeling Language) es un lenguaje que permite modelar, construir y documentar los elementos que forman un sistema software orientado a objetos. Se ha convertido en el estándar de facto de la industria, debido a que ha sido impulsado por los autores de los tres métodos más usados de orientación a objetos: Grady Booch, Ivar Jacobson y Jim Rumbaugh. Estos autores fueron contratados por la empresa Rational Software Co. para crear una notación unificada en la que basar la construcción de sus herramientas CASE. En el proceso de creación de UML han participado, no obstante, otras empresas de gran peso en la industria como Microsoft, Hewlett-Packard, Oracle o IBM, así como grupos de analistas y desarrolladores.

Esta notación ha sido ampliamente aceptada debido al prestigio de sus creadores y debido a que incorpora las principales ventajas de cada uno de los métodos particulares en los que se basa (principalmente Booch, OMT y OOSE). UML ha puesto fin a las llamadas “guerras de métodos” que se han mantenido a lo largo de los 90, en las que los principales métodos sacaban nuevas versiones que incorporaban las técnicas de los demás. Con UML se fusiona la notación de estas técnicas para formar una herramienta compartida entre todos los ingenieros software que trabajan en el desarrollo orientado a objetos.

1.10.5 Análisis Orientado a Objetos

Análisis orientado a objetos esta basado en un modelo de cinco capas:

- Capa clase/objeto.
- Capa de estructura.
- Capa de atributos.
- Capa de servicios
- Capa de tema.

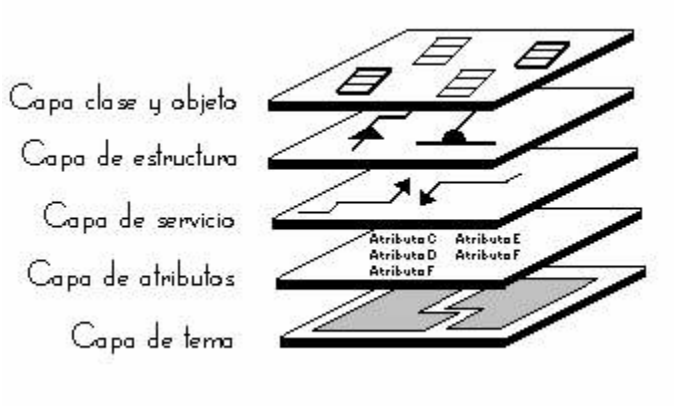

Figura 5.0. Capas del análisis orientado a objetos.

1.10.6 Análisis y clases de objetos

Objeto: es una abstracción de algo en un dominio de un problema que refleja las capacidades de un sistema para llevar información acerca de ello, interactuar con ello a ambas cosas.

Es una representación en computadora de alguna cosa o evento del mundo real. Pueden tener tanto atributos y comportamientos.

Clase. Es una categoría de objetos similares. Los objetos se agrupan en clases. Una clase define el conjunto de atributos y comportamientos compartidos que se encuentran a cada objeto de la clase.

Clase y objeto. Un término que se refiere tanto a clase como a los objetos que ocurren en la clase.

Hay cinco tipos generales de objetos que pueden descubrirse durante el análisis. Los objetos a veces representan cosas tangibles como vehículos, dispositivos y libros. Algunas veces los objetos representan papeles actuados por personas u organizaciones. Los objetos también pueden ser derivados de incidentes o eventos. Otros objetos pueden indicar interacciones tales como una venta o un matrimonio. Las interacciones tienen una cualidad de transacción o contrato. Los objetos también pueden detallar especificaciones. Las especificaciones tienen estándares o una cualidad de definición y, por lo general, implican que otros objetos representaran ocurrencias de cosas tangibles.

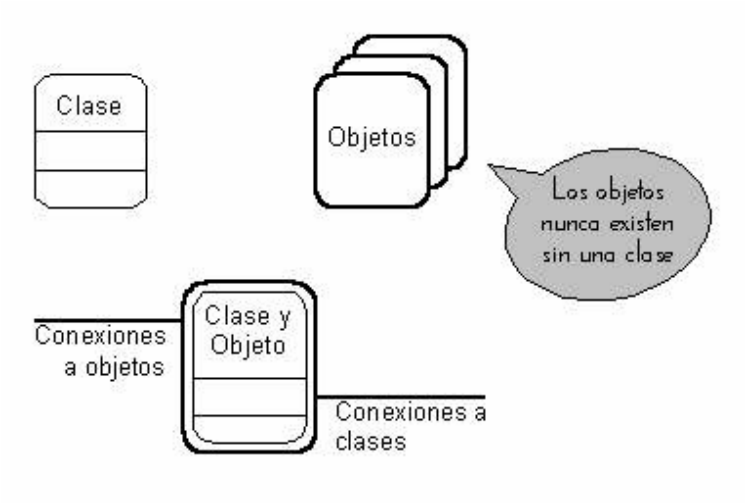

Figura 5.1. Representación del análisis orientado a objetos.

Las clases son representadas por cuadros rectangulares redondeados (bubtángulos) divididos en tres partes. El nombre de la clase se muestra en la división superior del cuadro. Las otras dos divisiones se usan para las capas de atributo y servicio. Cuando una clase aparece sin objetos, puede ser solamente una clase base, debido a que la única razón para tal clase "sin objetos" es que sea un medio para agrupar atributos y servicios que serán heredados por varias otras clases.

- Los objetos que tienen ocurrencia de una clase son representados por un cuadro sombreado rodeado por la clase. Debido a que los objetos tienen ocurrencias de una clase.
- Criterios que podemos usar para que nos ayuden a determinar si se justifica una nueva clase de objetos:
- Hay una necesidad de recordar el objeto. Esto es, el objeto puede ser descrito en un sentido definido y sus atributos son relevantes para el problema.
- Hay una necesidad de determinados comportamientos del objeto. Esto es, aunque un objeto no tenga atributos, hay servicios que debe proporcionar o estados de objeto que deben ser llamados.
- Usualmente un objeto tendrá varios atributos. Los objetos que tienen solamente uno o dos atributos sugieren diseños sobreanalizados.
- Usualmente una clase tendrá mas de una instancia de objeto, a menos de que sea una clase base.
- Usualmente los atributos tendrán siempre un valor significativo para cada objeto de la clase. Los objetos que producen valor NULO para un atributo, o para los que no es aplicable un atributo, por lo general implican una estructura generalización-especificación.
- Usualmente los servicios siempre se comportarán en la misma forma para todos los objetos de la clase. Los servicios que varían dramáticamente para algunos objetos de una clase o que regresan sin realizar acción para algunos objetos también sugieren una estructura generalización-especificación.
- Los objetos deben implementar requerimientos que son derivados del problema y no de la tecnología de solución. La parte de análisis del proyecto orientado a objetos no debe llegar a ser dependiente de una tecnología de implementación particular, tal como un sistema de computadora especifico o un lenguaje de programación especifico. Los objetos que atienden tales detalles técnicos no deben aparecer sino hasta muy tarde en la etapa de diseño. Los objetos dependientes de la tecnología sugieren que el proceso de análisis tiene fallas.
- Los objetos no deben duplicar atributos o servicios que pueden ser derivados de otros objetos del sistema. Por ejemplo, un objeto que guarda la edad de un empleado es superfluo cuando existe un objeto de empleado separado que conserva el atributo fecha de

nacimiento. El objeto edad puede ser eliminado por un servicio edad que es componente del objeto empleado.

1.10.7 Principales características de los manejadores de base de datos orientado a objetos

Permiten el almacenamiento de estructuras datos complejas que no pueden ser almacenadas fácilmente en bases de datos convencionales.

No sólo almacenan tablas, columnas y renglones.

Soportan toda la persistencia necesaria cuando se trabaja con lenguajes OO. Fueron diseñadas para ser integradas de forma transparente con la programación OO.

Sistema administrador de bases de datos que soporta el modelado y creación de datos como objetos.

1.10.8 Persistencia

- Gestión del almacenamiento secundario
- Concurrencia
- Recuperación
- Ad hoc query
- Objetos complejos -Abstracción
- Identidad de los objetos
- Encapsulación
- Tipos o clases
- Herencia - Jerarquía

Persistencia: Conservación de los datos; permanencia.
Concurrencia: Interacción de muchos usuarios.
Recuperación: En caso de fallas de HW o SW, retroceder a un estado coherente de los datos.
Gestión del almacenamiento secundario: Mecanismos no visibles al usuario para mantener la independencia lógica y física del sistema.

1.11 MÓDULO XI TÓPICOS AVANZADOS DE BASE DE DATOS

1.11.1 Data Mining

La tecnología informática constituye la infraestructura fundamental de las grandes organizaciones y permite, hoy, registrar múltiples detalles de la vida de las empresas. Las bases de datos posibilitan almacenar cada transacción, así como otros muchos elementos que reflejan la interacción de la organización con otras organizaciones, clientes, o internamente, entre sus divisiones y empleados, etcétera.

Es imprescindible convertir los grandes volúmenes de datos existentes en experiencia, conocimiento y sabiduría, formas que atesora la humanidad para que sea útil a la toma de decisiones, especialmente en las grandes organizaciones y proyectos científicos. La búsqueda de información relevante siempre es útil a la administración empresarial: el control de la producción, el análisis de los mercados, el diseño en ingeniería y la exploración científica, porque pueden ofrecer las respuestas más apropiadas a las necesidades de información. Varias preguntas se relacionan frecuentemente con los datos, la información y el conocimiento. Su respuesta, demanda la participación de varios especialistas. ¿Cómo puede entenderse un fenómeno sobre la base de la interpretación de grandes volúmenes de datos? ¿De qué manera puede utilizarse la información para la toma de decisiones?, son algunos ejemplos de interrogantes comunes.

La respuesta a estas preguntas es el objetivo del DM, un conjunto de técnicas agrupadas con el fin de crear mecanismos adecuados de dirección, entre ellas puede citarse la estadística, el reconocimiento de patrones, la clasificación y la predicción.

Para descubrir patrones de relaciones útiles en un conjunto de datos se empezaron a utilizar métodos que fueron denominados de diferente forma. El término Data Mining, en inglés, no era, al principio, del agrado de muchos estadísticos, porque sus investigaciones estaban dirigidas a procesar y reprocesar suficientemente los datos, hasta que confirmasen o refutasen las hipótesis planteadas. Desde este ángulo, la minería de datos aplica una dinámica que se mueve en sentido contrario al método científico tradicional.

Con frecuencia, el investigador formula una hipótesis; luego, diseña un experimento para captar los datos necesarios y realizar los experimentos que confirmen o refuten la hipótesis planteada. Este es un proceso, que realizado de forma rigurosa, debe generar nuevos conocimientos.

En la minería de datos, por el contrario, se captan y procesan los datos con la esperanza de que de ellos surja una hipótesis apropiada. Se desea que los datos nos describan o indiquen el porqué presentan determinada configuración y comportamiento.

El DM es un mecanismo de explotación, consistente en la búsqueda de información valiosa en grandes volúmenes de datos. Está muy ligada a los Data Warehousing que proporcionan la información histórica con la cual los algoritmos de minería de datos tienen la información necesaria para la toma de decisiones.

1.11.2 La minería de datos (DM) y el descubrimiento de conocimientos en bases de datos (KDD).

Existe cierta tendencia a identificar como sinónimos a la minería de datos y el descubrimiento de conocimientos en bases de datos, que de forma abreviada se refiere con las siglas KDD (del inglés *Knowledge Discovery in Data Bases*), la convergencia del aprendizaje automático, la estadística, el reconocimiento de patrones, la inteligencia artificial, las bases de datos, la visualización de datos, los sistemas para el apoyo a la toma de decisiones, la recuperación de información y otros muchos campos.

El KDD es el proceso completo de extracción de conocimientos, no trivial, previamente desconocidos y potencialmente útil a partir de un conjunto de datos, mientras que «la minería de datos es una compilación de técnicas reunidas para crear mecanismos adecuados para la toma de decisiones. Entre estas técnicas se pueden citar la estadística, el reconocimiento de patrones, la clasificación y la predicción, la excavación de información relevante de la administración empresarial, el control de la producción, el análisis de los mercados, el diseño en ingeniería y la exploración científica." En otras palabras, el concepto minería de datos se asocia al proceso de construcción de reglas a partir de colecciones de datos con una finalidad previamente determinada y para su uso en la toma de decisiones con respecto a dicha finalidad. El concepto de KDD no comprende necesariamente esta segunda parte. Esta diferencia, muchas veces inadvertida, puede ser la causa de que ambos conceptos se utilicen indistintamente en gran parte de la literatura.

1.11.3 Etapas principales del proceso de Data Mining

1.- Determinación de los objetivos: delimitar los objetivos que el cliente desea bajo la orientación del especialista en DM.

2.- Preprocesamiento de los datos: se refiere a la selección, la limpieza, el enriquecimiento, la reducción y la transformación de las bases de datos.

Esta etapa consume generalmente alrededor del setenta por ciento del tiempo total de un proyecto de DM.

3.- Determinación del modelo: se comienza realizando un análisis estadístico de los datos, y después se lleva a cabo una visualización gráfica de los mismos para tener una primera aproximación. Según los objetivos planteados y la tarea que debe llevarse a cabo, pueden utilizarse algoritmos desarrollados en diferentes áreas de la Inteligencia Artificial.

4.- Análisis de los resultados: verifica si los resultados obtenidos son coherentes y los coteja con los obtenidos por el análisis estadístico y de visualización gráfica. El cliente determina si son novedosos y si le aportan un nuevo conocimiento que le permita considerar sus decisiones.

1.11.4 Fundamentos del Data Mining

Las técnicas de DM son el resultado de un largo proceso de investigación y desarrollo de productos. Esta evolución comenzó cuando los datos de negocios fueron almacenados por primera vez en computadoras, y continuó con mejoras en el acceso a los datos, y más recientemente con tecnologías generadas para permitir a los usuarios navegar a través de los datos en tiempo real. DM toma este proceso de evolución más allá del acceso y navegación retrospectiva de los datos, hacia la entrega de información prospectiva y proactiva. DM está listo para su aplicación en la comunidad de negocios porque está soportado por tres tecnologías que ya están suficientemente maduras:

- Recolección masiva de datos
- Potentes computadoras con multiprocesadores
- Algoritmos de Data Mining

Los componentes esenciales de la tecnología de DM han estado bajo desarrollo por décadas, en áreas de investigación como estadísticas, inteligencia artificial y aprendizaje de máquinas. Hoy, la madurez de estas técnicas, junto con los motores de bases de datos relacionales de alta performance, hicieron que estas tecnologías fueran prácticas para los entornos de Data Warehouse actuales.

El DM envuelve muchos diferentes algoritmos para resolver diferentes tareas.

Todos estos algoritmos intentan estar en un modelo de datos. El algoritmo examina los datos y determina un modelo que es cerrado. Los algoritmos de DM pueden ser caracterizados consistiendo en tres partes:

- Modelo: El propósito del algoritmo es el de colocar el dato en un modelo.
- Preferencia: Algún criterio debe ser usado para colocar un modelo en otro.
- Buscar: Todos los algoritmos requieren alguna técnica para buscar el dato.

1.11.5 Data Warehousing

El DataWarehouse (DW) no es un producto que pueda ser comprado en el mercado, sino más bien un concepto que debe ser construido. DW es una combinación de conceptos y tecnología que cambian significativamente la manera en que es entregada la información a la gente de negocios. El objetivo principal es satisfacer los requerimientos de información internos de la empresa para una mejor gestión, con eficiencia y facilidad de acceso.

El DW puede verse como una bodega donde están almacenados todos los datos necesarios para realizar las funciones de gestión de la empresa, de manera que puedan utilizarse fácilmente según se necesiten. El contenido de los datos, la organización y estructura son dirigidos a satisfacer las necesidades de información de analistas.

El DW intenta responder a la compleja necesidad de obtención de información útil sin el sacrificio del rendimiento de las aplicaciones operacionales, debido a lo cual se ha convertido actualmente en una de las tendencias tecnológicas más significativas en la administración de información.

Los almacenes de datos (o DataWarehouse) generan bases de datos tangibles con una perspectiva histórica, utilizando datos de múltiples fuentes que se fusionan en forma congruente. Estos datos se mantienen actualizados, pero no cambian al ritmo de los sistemas transaccionales. Muchos DW se diseñan para contener un nivel de detalle hasta el nivel de transacción, con la intención de hacer disponible todo tipo de datos y características, para reportar y analizar.

Así un DW resulta ser un recipiente de datos transaccionales para proporcionar consultas operativas, y la información para poder llevar a cabo análisis multidimensional. De esta forma, dentro de una almacén de datos existen dos tecnologías complementarias, una relacional para consultas y una multidimensional para análisis.

Existen muchas definiciones para el DW, la más conocida fue propuesta por Inmon[MicroSt96] (considerado el padre de las Bases de Datos) en 1992: “Un DW es una colección de datos orientados a temas, integrados, no-volátiles y variante en el tiempo, organizados para soportar necesidades empresariales”.

En 1993, Susan Osterfeldt[MicroSt96] publica una definición que sin duda acierta en la clave del DW: “Yo considero al DW como algo que provee dos beneficios empresariales reales: Integración y Acceso de datos. DW elimina una gran cantidad de datos inútiles y no deseados, como también el procesamiento desde el ambiente operacional clásico”. Esta última definición refleja claramente el principal beneficio que el DW aporta a la empresa, eliminar aquellos datos que obstaculizan la labor de análisis de información y entregar la información que se requiere en la forma más apropiada, facilitando así el proceso de gestión.

DW se sustenta en un procesamiento distinto al utilizado por los sistemas operacionales, OLAP (Procesamiento Analítico En Línea), el cual surge como un proceso para ser usado en el análisis de negocios y otras aplicaciones que requieren una visión flexible del negocio.

1.11.6 DataMart

El concepto DataMart es una extensión natural del DW, y está enfocado a un departamento o área específica, como por ejemplo los departamentos de Finanzas o Marketing, permitiendo así un mejor control de la información que se está abarcando.

OLAP, R-OLAP y M-OLAP

Un sistema OLAP se puede entender como la generalización de un generador de informes. Las aplicaciones informáticas clásicas de consulta, orientadas a la toma de decisiones, deben ser programadas. Atendiendo a las necesidades del usuario, se crea una u otra interfaz. Sin embargo, muchos desarrolladores se dieron cuenta de que estas aplicaciones eran susceptibles de ser generalizadas y servir para casi cualquier necesidad, esto es, para casi cualquier base de datos. Los sistemas OLAP evitan la necesidad de desarrollar interfaces de consulta, y ofrecen un entorno único válido para el análisis de cualquier información histórica, orientado a la toma de decisiones. A cambio, es necesario definir dimensiones, jerarquías y variables, organizando de esta forma los datos.

Para los desarrolladores de aplicaciones acostumbrados a trabajar con bases de datos relacionales, el diseño de una base de datos multidimensional puede ser complejo o al menos, extraño. Pero en general, nuestra experiencia nos dice que el diseño de dimensiones y variables es mucho más sencillo e intuitivo que un diseño relacional. Esto es debido a que las dimensiones y variables son reflejo directo de los informes en papel utilizados por la organización.

Una vez que se ha decidido emplear un entorno de consulta OLAP, se ha de elegir entre R-OLAP y M-OLAP. R-OLAP es la arquitectura de base de datos multidimensional en la que los datos se encuentran almacenados en una base de datos relacional, la cual tiene forma de estrella (también llamada copo de nieve o araña). En R-OLAP, en principio la base de datos sólo almacena información relativa a los datos en detalle, evitando acumulados (evitando redundancia).

En un sistema M-OLAP, en cambio, los datos se encuentran almacenados en archivos con estructura multidimensional, los cuales reservan espacio para todas las combinaciones de todos los posibles valores de todas las dimensiones de cada una de las variables, incluyendo los valores de dimensión que representan acumulados. Es decir, un sistema M-OLAP contiene precalculados (almacenados) los resultados de todas las posibles consultas a la base de datos.

M-OLAP consigue consultas muy rápidas a costa de mayores necesidades de almacenamiento, y retardos en las modificaciones (que no deberían producirse salvo excepcionalmente), y largos procesos *batch* de carga y cálculo de acumulados. En R-OLAP, al contener sólo las combinaciones de valores de dimensión que representan detalle, es decir, al no haber redundancia, el archivo de base de datos es pequeño. Los procesos *batch* de carga son rápidos (ya que no se requiere agregación), y sin embargo, las consultas pueden ser muy lentas, por lo que se aplica la solución de tener al menos algunas consultas precalculadas.

En M-OLAP, el gran tamaño de las variables multidimensionales o el retardo en los procesos *batch* puede ser un inconveniente.

1.11.7 Características de un Data Warehouse

Entre las principales se tiene:

- Orientado al tema
- Integrado
- De tiempo variante
- No volátil

1.11.8 Orientado a temas

Una primera característica del DW es que la información se clasifica en base a los aspectos que son de interés para la empresa. Siendo así, los datos tomados están en contraste con los clásicos procesos orientados a las aplicaciones.

1.11.9 Integración

El aspecto más importante del ambiente DW es que la información encontrada al interior está siempre integrada.

La integración de datos se muestra de muchas maneras: en convenciones de nombres consistentes, en la medida uniforme de variables, en la codificación de estructuras consistentes, en atributos físicos de los datos consistentes, fuentes múltiples y otros.

1.11.10 De tiempo variante

Toda la información del DW es requerida en algún momento. Esta característica básica de los datos en un depósito, es muy diferente de la información encontrada en el ambiente operacional. En éstos, la información se requiere al momento de acceder. En otras palabras, en el ambiente operacional, cuando usted accede a una unidad de información, usted espera que los valores requeridos se obtengan a partir del momento de acceso.

Como la información en el DW es solicitada en cualquier momento (es decir, no "ahora mismo"), los datos encontrados en el depósito se llaman de "tiempo variante".

Los datos históricos son de poco uso en el procesamiento operacional. La información del depósito por el contraste, debe incluir los datos históricos para usarse en la identificación y evaluación de tendencias.

El tiempo variante se muestra de varias maneras:

1. La más simple es que la información representa los datos sobre un horizonte largo de tiempo - desde cinco a diez años. El horizonte de tiempo representado para el ambiente operacional es mucho más corto - desde valores actuales hasta sesenta a noventa días.

Las aplicaciones que tienen un buen rendimiento y están disponibles para el procesamiento de transacciones, deben llevar una cantidad mínima de datos si tienen cualquier grado de flexibilidad. Por ello, las aplicaciones operacionales tienen un corto horizonte de tiempo, debido al diseño de aplicaciones rígidas.

2. La segunda manera en la que se muestra el tiempo variante en el DW está en la estructura clave. Cada estructura clave en el DW contiene, implícita o explícitamente, un elemento de tiempo como día, semana, mes, etc.

El elemento de tiempo está casi siempre al pie de la clave concatenada, encontrada en el DW. En ocasiones, el elemento de tiempo existirá implícitamente, como el caso en que un archivo completo se duplica al final del mes, o al cuarto.

3. La tercera manera en que aparece el tiempo variante es cuando la información del DW, una vez registrada correctamente, no puede ser actualizada. La información del DW es, para todos los propósitos prácticos, una serie larga de "snapshots" (vistas instantáneas).

1.11.11 Base de Datos Inteligentes

Las bases de datos inteligentes representan una tecnología para la administración de la información que fue envuelta como resultado de la integración de las bases de datos tradicionales con otros campos como lo son:

- Programación orientada a objetos.
- Sistemas expertos.
- Hypermedia.
- Recepción de información en línea.

La arquitectura de las bases de datos inteligentes constan de tres niveles, los cuales son:

- High-level tools
- High-level user interface
- Intelligent database engine

1.11.12 High-level tools

Estas herramientas proveen al usuario muchas facilidades como son búsquedas inteligentes, calidad de los datos y control de integridad, además de descubrimiento autómata. Estas herramientas representan una biblioteca externa de herramientas poderosas que algunos usuarios quizás encuentren útiles y otros no. Muchas de éstas pueden ser clasificadas como técnicas de administración para la información.

1.11.13 High-level user interface

En este nivel es en el cual los usuarios interactúan directamente. Este nivel crea el modelo de la tarea y ambiente de base de datos con el que el usuario interactuará.
La interfaz del usuario es presentada en dos aspectos. Éste es un modelo central que es presentado al usuario. Este modelo central consiste de la representación orientada a objetos de la información con una colección de herramientas integradas para crear nuevos tipos de objetos, buscando y respondiendo preguntas. En adición, estás son una colección de herramientas de alto nivel, el cual aunque no es una parte esencial del modelo central, da un realce a la funcionalidad del sistema de base de datos inteligente para ciertas clases y usuarios.

Normalmente se pueden distinguir dos niveles de la interfaz del usuario, las cuales son:

- El nivel físico.
- El nivel cognitivo.

El nivel físico de la interfaz es típicamente más obvia, consistiendo de entradas y salida de dispositivos, semejantes al ratón, teclado, monitor. En contraste, el nivel cognitivo de la interfaz es más difícil para describir., consistiendo del modelo subyacente usado para presentar la información, la interpretación que el usuario entonces hace, y las intensiones que entonces el usuario formula.

1.11.4 Intelligent Database Engine

Este es el nivel base. El motor de base de datos inteligente incorpora un modelo que permite hacer una deductiva orientación a objetos representación de la información, que puede ser expresada y operada sobre una variedad de caminos. El motor incluye procedimientos de inferencia de encadenamiento hacia delante y hacia atrás. Muchas de estas características del motor integrado dependerá sobre las especificaciones del ambiente de hardware y software en el cual la base de datos inteligente es implementado.

La interfaz del usuario es soportada por una colección de capacidades de la base de datos. Estas capacidades son el mecanismo que permite un sistema de administración de base de datos o aplicaciones de administración de información su funcionamiento. Ejemplo de estas capacidades incluye el procesamiento de query's y la habilidad para llevar fuera del razonamiento deductivo.

La inteligencia de la interfaz del usuario esta en gran parte determinada por la inteligencia de la aplicación subyacente. Las siguientes son algunas características de un sistema de base de datos y este nivel que realza el total de inteligencia del sistema.

- Modelo de datos basado en el conocimiento y orientado a objetos.
- Base de datos integradas y motor de inferencia.
- Búsquedas sensitivas al contexto de estructura sensitiva.
- Soporte de múltiples medios de almacenamiento.
- Administración inteligente de versión, recuperación y resistencia.
- Soporte de transacciones y concurrencia.
- Optimización de query's.

Un modelo de datos basado en el conocimiento y orientado a objetos permite la representación de información en una forma la cual refleja lo más fácilmente posible la percepción de los usuarios del mundo real.

Las bases de datos integradas y los motores de inferencia siguen después del uso del modelo de datos basado en el conocimiento. Éstos permiten la recuperación deductiva para ser llevado en un ambiente fuera donde la búsqueda e inferencia de la información son integradas.

La búsqueda estructura sensitiva envuelve la recuperación del conocimiento basado sobre la forma. La búsqueda sensitiva envuelve sabiendo donde buscar la información relevante basado en el contexto.

Los medios múltiples de almacenamiento permiten una variedad de tipos (gráficas, sonido, etc.) para ser eficientemente almacenados y recuperados dentro de la base de datos.

Administración inteligente de versión se asegura que las versiones previas y la actual de las bases de datos de desarrollo, sean recuperadas eficientemente.

La recuperación y resistencia son las ediciones que tratan el grado a el cual la base de datos maneja las demandas operacionales que experimenta, incluyendo simulación de acceso de muchos usuarios y fallas de equipo. Así como muchas convencionales bases de datos, las bases de datos inteligentes soportan transacción atómica concurrente. En adición, el motor subyacente de una base de datos inteligente ejecuta extensiva optimización de query para proveer adecuadas

respuestas de tiempo real para complejos queries envolviendo bases de conocimiento orientadas a objetos.

1.11.15 Base de Datos Multidimencionales

El modelamiento Dimensional es una técnica para modelar bases de datos simples y entendibles al usuario final. La idea fundamental es que el usuario visualice fácilmente la relación que existe entre las distintas componentes del modelo.

Consideremos un punto en el espacio. El espacio se define a través de sus ejes coordenados (por ejemplo X, Y, Z). Un punto cualquiera de este espacio quedará determinado por la intersección de tres valores particulares de sus ejes.

Si se le asignan valores particulares a estos ejes. Digamos que el eje X representa Productos, el eje Y representa el Mercado y, el eje Z corresponde al
Tiempo. Se podría tener por ejemplo, la siguiente combinación: producto = madera, mercado = Concepción, tiempo = diciembre-1998. La intersección de estos valores nos definirá un solo punto en nuestro espacio. Si el punto que buscamos, lo definimos como la cantidad de madera vendida, entonces se tendrá un valor específico y único para tal combinación.

En el modelo multidimensional cada eje corresponde a una dimensión particular. Entonces la dimensionalidad de nuestra base estará dada por la cantidad de ejes (o dimensiones) que le asociemos.

Cuando una base puede ser visualizada como un cubo de tres o más dimensiones, es más fácil para el usuario organizar la información e imaginarse en ella cortando y rebanando el cubo a través de cada una de sus dimensiones, para buscar la información deseada.

Por ejemplo:

La descripción de una organización típica es: “Nosotros vendemos productos en varios mercados, y medimos nuestro desempeño en el tiempo”: Un diseñador dimensional lo verá como: “Nosotros vendemos productos en varios mercados, y medimos nuestro desempeño en el tiempo. Donde cada palabra subrayada corresponde a una dimensión.

Esto puede visualizarse como un cubo, donde cada punto dentro del cubo es una intersección de coordenadas definidas por los lados de éste (dimensiones).
Ejemplos de medidas son: unidades producidas, unidades vendidas, costo de unidades producidas, ganancias($) de unidades vendidas, etc.

1.11.16 Características del modelo multidimensional

En general, la estructura básica de un DW para el Modelo Multidimensional está definida por dos elementos: esquemas y tablas.

Tablas DW: como cualquier base de datos relacional, un DW se compone de tablas. Hay dos tipos básicos de tablas en el Modelo Multidimensional:

Tablas Fact: contienen los valores de las medidas de negocios, por ejemplo:
Ventas promedio en dólares, número de unidades vendidas, etc.

Tablas Lock_up: contienen el detalle de los valores que se encuentran asociados a la tabla Fact.

Esquemas DW: la colección de tablas en el DW se conoce como Esquema. Los esquemas caen dentro de dos categorías básicas: esquemas estrellas y esquemas snowflake.

1.11.17 Esquema de estrella

En general, el modelo multidimensional también se conoce con el nombre de esquema estrella, pues su estructura base es similar: una tabla central y un conjunto de tablas que la atienden radialmente.

El esquema estrella deriva su nombre del hecho que su diagrama forma una estrella, con puntos radiales desde el centro. El centro de la estrella consiste de una o más tablas fact, y las puntas de la estrella son las tablas lock_up.

Este modelo entonces, resulta ser asimétrico, pues hay una tabla dominante en el centro con varias conexiones a las otras tablas. Las tablas Lock-up tienen sólo la conexión a la tabla fact y ninguna más.

1.11.18 Esquema snowflake

La diferencia del esquema snowflake comparado con el esquema estrella, está en la estructura de las tablas lock_up: las tablas lock_up en el esquema snowflake están normalizadas. Cada tabla lock_up contiene sólo el nivel que es clave primaria en la tabla y la foreign key de su parentesco del nivel más cercano del diagrama.

1.11.19 Tabla fact o de hechos

Es la tabla central en un esquema dimensional. Es en ella donde se almacenan las mediciones numéricas del negocio. Estas medidas se hacen sobre el grano, o unidad básica de la tabla.

El grano o la granularidad de la tabla queda determinada por el nivel de detalle que se almacenará en la tabla. Por ejemplo, para el caso de producto, mercado y tiempo, el grano puede ser la cantidad de madera vendida ‘mensualmente’. El grano revierte las unidades atómicas en el esquema dimensional.
Cada medida es tomada de la intersección de las dimensiones que la definen.
Idealmente está compuesta por valores numéricos, continuamente evaluados y aditivos. La razón de estas características es que así se facilita que los miles de registros que involucran una consulta sean comprimidos en unas pocas líneas en un set de respuesta.

La clave de la tabla fact recibe el nombre de clave compuesta o concatenada debido a que se forma de la composición (o concatenación) de las llaves primarias de las tablas dimensionales a las que está unida.

Así entonces, se distinguen dos tipos de columnas en una tabla fact: columnas fact y columnas key.

Donde la columna fact es la que almacena alguna medida de negocio y una columna key forma parte de la clave compuesta de la tabla.

1.11.20 Tablas Lock-up o dimensionales

Estas tablas son las que se conectan a la tabla fact, son las que alimentan a la tabla fact. Una tabla lock_up almacena un conjunto de valores que están relacionados a una dimensión particular. Tablas lock_up no contienen hechos, en su lugar los valores en las tablas lock_up son los elementos que determinan la estructura de las dimensiones. Así entonces, en ellas existe el detalle de los valores de la dimensión respectiva.

Una tabla lock_up está compuesta de una primary key que identifica unívocamente una fila en la tabla junto con un conjunto de atributos, y dependiendo del diseño del modelo multidimensional puede existir una foreign key que determina su relación con otra tabla lock_up.

Para decidir si un campo de datos es un atributo o un hecho se analiza la variación de la medida a través del tiempo. Si varía continuamente implicaría tomarlo como un hecho, caso contrario será un atributo.

Los atributos dimensionales son un rol determinante en un DDW. Ellos son la fuente de todas las necesidades que debieran cubrirse. Esto significa que la base de datos será tan buena como lo sean los atributos dimensionales, mientras más descriptivos, manejables y de buena calidad, mejor será el DDW.

CAPÍTULO II
SQL SERVER 2005 BUSINESS INTELLIGENT

2.1 SQL SERVER INTEGRATION SERVICES

Microsoft SQL Server 2005 Integration Services (SSIS) es una plataforma que permite generar soluciones de integración de datos de alto rendimiento, entre las que se incluyen paquetes de extracción, transformación y carga (ETL) para el almacenamiento de datos.

Integration Services incluye herramientas gráficas y asistentes para generar y depurar paquetes, tareas para realizar funciones de flujo de trabajo, como las operaciones de FTP, tareas para ejecutar instrucciones SQL o para enviar mensajes de correo electrónico, orígenes y destinos de datos para extraer y cargar datos, transformaciones para limpiar, agregar, mezclar y copiar datos, un servicio de administración, e interfaces de programación de aplicaciones (API) para programar el modelo de objetos de Integration Services.

Integration Services reemplaza Servicios de transformación de datos (DTS), que se incluyó por primera vez como componente de SQL Server 7.0.

2.1.1 Usos típicos de Integration Services

SSIS proporciona un amplio conjunto de tareas, contenedores, transformaciones y adaptadores de datos integrados que permiten desarrollar aplicaciones de negocios. Sin escribir una sola línea de código, puede crear soluciones de SSIS para resolver problemas de negocios complejos mediante ETL y Business Intelligence, administrar bases de datos de SQL Server y copiar objetos de SQL Server entre instancias de SQL Server.

En los escenarios siguientes se describen usos típicos de los paquetes de SSIS.

Los datos suelen almacenarse en muchos sistemas de almacenamiento de datos distintos, por lo que extraer datos de todos los orígenes y mezclarlos en un solo conjunto de datos coherente constituye un desafío. Esta situación puede producirse por diversas razones.

Por ejemplo:

Muchas organizaciones archivan información que está almacenada en sistemas de almacenamiento de datos antiguos. Estos datos pueden no ser importantes para las operaciones diarias, pero pueden resultar útiles para el análisis de tendencias, que requiere datos recopilados a lo largo de un período prolongado de tiempo.

Las sucursales de una organización pueden usar distintas tecnologías de almacenamiento de datos para almacenar los datos operativos. Es posible que el paquete tenga que extraer datos de hojas de cálculo y de bases de datos relacionales para poder mezclar los datos.

Los datos pueden estar almacenados en bases de datos que usan distintos esquemas para los mismos datos. Es posible que el paquete tenga que cambiar el tipo de datos de una columna o combinar datos de varias columnas en una sola columna para poder mezclar los datos.

SSIS puede conectarse a una gran variedad de orígenes de datos, incluso con varios orígenes en un solo paquete. Un paquete puede conectarse a bases de datos relacionales mediante proveedores .NET y DB OLE, y a muchas bases de datos antiguas mediante controladores ODBC.

También puede conectarse con archivos planos, archivos de Excel y proyectos de Analysis Services.

SSIS incluye componentes de origen que extraen datos de archivos planos, hojas de cálculo Excel, documentos XML y tablas y vistas de bases de datos relacionales desde el origen de datos al que se conecta el paquete.

A continuación, los datos se suelen transformar mediante las transformaciones incluidas en Integration Services. Cuando los datos se han transformado a formatos compatibles, pueden mezclarse físicamente en un conjunto de datos.

Después de mezclar correctamente los datos y aplicarles transformaciones, se suelen cargar en uno o varios destinos. SSIS incluye un destino para cargar datos en archivos planos, archivos sin procesar y bases de datos relacionales. Los datos también se pueden cargar en un conjunto de registros en memoria a los que tienen acceso otros elementos del paquete.

Los datos de los almacenamientos de datos y los puestos de datos suelen actualizarse frecuentemente y normalmente las cargas de datos son muy grandes.

SSIS incluye una tarea que realiza una carga masiva de datos directamente desde un archivo plano a tablas y vistas de SQL Server, y un componente de destino que realiza una carga masiva de datos en una base de datos de SQL Server como último paso de un proceso de transformación de datos.

Se puede configurar un paquete de SSIS como reiniciable. Esto significa que podrá volver a ejecutar el paquete desde un punto de comprobación predeterminado (una tarea o un contenedor del paquete). La capacidad de reiniciar un paquete permite ahorrar mucho tiempo, especialmente si el paquete procesa datos de un gran número de orígenes.

Puede utilizar paquetes de SSIS para cargar las tablas de dimensiones y hechos en la base de datos. Si los datos de origen de una tabla de dimensiones están almacenados en varios orígenes de datos, el paquete puede mezclar los datos en un conjunto de datos y cargar la tabla de dimensiones en un solo proceso, en lugar de utilizar un proceso independiente para cada origen de datos.

La actualización de datos de almacenamientos de datos y puestos de datos puede ser compleja, ya que ambos tipos de almacenes de datos suelen incluir dimensiones de variación lenta que pueden ser difíciles de administrar mediante un proceso de transformación de datos. El Asistente para dimensiones de variación lenta automatiza la compatibilidad para las dimensiones de variación lenta, creando dinámicamente las instrucciones SQL que insertan y actualizan registros, actualizan registros relacionados y agregan nuevas columnas a las tablas.

Además, las tareas y transformaciones de los paquetes de Integration Services pueden procesar cubos y dimensiones de Analysis Services. Cuando el paquete actualiza las tablas de la base de datos con las que se ha generado un cubo, puede utilizar tareas y transformaciones de Integration Services para procesar automáticamente el cubo y las dimensiones. El procesamiento de los cubos y las dimensiones ayuda automáticamente a mantener actualizados los datos de los usuarios de ambos entornos: los usuarios que tienen acceso a la información de cubos y dimensiones, y los usuarios que tienen acceso a datos de una base de datos relacional.

SSIS también puede calcular funciones antes de que se carguen los datos en el destino. Si los almacenamientos de datos y los puestos de datos almacenan información agregada, el paquete de SSIS puede calcular funciones como SUM, AVERAGE y COUNT. Una transformación también puede cambiar datos relacionales y transformarlos a un formato menos normalizado pero más compatible con la estructura de las tablas del almacenamiento de datos.

2.1.2 Limpiar y normalizar datos

Independientemente de si los datos se van a cargar en una base de datos de procesamiento de transacciones en línea (OLTP) o de procesamiento analítico en línea (OLAP), una hoja de cálculo de Excel o un archivo, hay que limpiarlos y normalizarlos antes de cargarlos. Puede ser necesario actualizar los datos por las siguientes razones:

Los datos proceden de varias sucursales de una organización y en cada una de las sucursales se usan convenciones y estándares distintos. Para poder usar los datos, es posible que sea necesario cambiar su formato. Por ejemplo, es posible que tenga que combinar el nombre y el apellido en una columna.

Los datos pueden ser alquilados o comprados. Para poder usar los datos es posible que sea necesario normalizar y limpiar los datos de forma que satisfagan los estándares de negocios. Por ejemplo, una organización desea comprobar que todos los registros usan el mismo conjunto de abreviaturas de estado o el mismo conjunto de nombres de productos.

Los datos son específicos de la configuración regional. Por ejemplo, en los datos puede haber diversos formatos de fecha/hora o numéricos. Si se mezclan datos de configuraciones regionales distintas, deben convertirse a una configuración regional antes de cargarse para evitar que los datos resulten dañados.

SSIS incluye transformaciones integradas que se pueden agregar a paquetes para limpiar y normalizar datos, cambiar las mayúsculas y minúsculas de los datos, convertir datos a un tipo o formato distinto, o crear nuevos valores de columna basados en expresiones. Por ejemplo, el paquete podría concatenar las columnas de nombre y apellido en una sola columna y después convertir los caracteres a mayúsculas.

Un paquete de Integration Services también puede limpiar datos reemplazando los valores de las columnas por valores de una tabla de referencia mediante una búsqueda exacta o aproximada, a fin de encontrar los valores en una tabla de referencia. Normalmente, un paquete realiza la búsqueda exacta primero y, en caso de que no devuelva resultados, realiza la búsqueda aproximada. Por ejemplo, el paquete primero intenta buscar un nombre de producto en la tabla de referencia utilizando el valor de la clave principal del producto. Si esta búsqueda no devuelve el nombre del producto, el paquete intenta la búsqueda de nuevo, pero esta vez realiza una búsqueda aproximada del nombre del producto.

Otra transformación limpia los datos agrupando los valores similares de un conjunto de datos. Esto es útil para identificar registros que pueden ser duplicados y, por tanto, no se deben insertar en la base de datos sin realizar más evaluaciones. Por ejemplo, comparar las direcciones de los registros de clientes puede ayudar a identificar varios clientes duplicados.

2.1.3 Generar Business Intelligence en un proceso de transformación de datos

Un proceso de transformación de datos requiere lógica integrada para responder dinámicamente a los datos que procesa y a los que tiene acceso.

Es posible que sea necesario resumir, convertir y distribuir los datos en función de valores de datos. Incluso es posible que el proceso tenga que rechazar datos en función de una evaluación de valores de columna.

Para satisfacer este requisito, la lógica del paquete de SSIS puede tener que realizar los siguientes tipos de tareas:

- Mezclar datos de varios orígenes de datos.
- Evaluar datos y aplicar conversiones de datos.
- Dividir un conjunto de datos en múltiples conjuntos de datos en función de valores de datos.
- Aplicar agregaciones diferentes a distintos subconjuntos de un conjunto de datos.
- Cargar subconjuntos de los datos en destinos distintos o en varios destinos.

SSIS proporciona contenedores, tareas y transformaciones para generar Business Intelligence en paquetes.

Los contenedores admiten la repetición de flujos de trabajo recorriendo archivos u objetos, y evaluando expresiones. Un paquete puede evaluar datos y repetir flujos de trabajo en función de los resultados. Por ejemplo, si la fecha pertenece al mes actual, el paquete realiza un conjunto de tareas; en caso contrario, realiza un conjunto de tareas alternativas.

Las tareas que usan parámetros de entrada también pueden generar Business Intelligence en paquetes.

Por ejemplo, el valor de un parámetro de entrada puede filtrar los datos recuperados por una tarea.

Las transformaciones pueden evaluar expresiones y después, en función de los resultados, enviar filas de un conjunto de datos a diferentes destinos. Una vez divididos los datos, el paquete puede aplicar distintas transformaciones a cada subconjunto del conjunto de datos. Por ejemplo, una expresión puede evaluar una columna de fecha, agregar los datos de ventas del período correspondiente y después almacenar únicamente la información de resumen.

También es posible enviar un conjunto de datos a varios destinos y aplicar a continuación distintos conjuntos de transformaciones a los mismos datos. Por ejemplo, un conjunto de transformaciones puede resumir los datos mientras otro conjunto de transformaciones expande los datos buscando valores en tablas de referencia y agregando datos de otros orígenes.

2.1.4 Automatizar las funciones administrativas y la carga de datos

Normalmente, los administradores desean automatizar las funciones administrativas como la copia de seguridad y la restauración de bases de datos, la copia de bases de datos de SQL Server y los objetos que contienen, la copia de objetos de SQL Server y la carga de datos. Los paquetes de Integration Services pueden realizar estas funciones.

Integration Services incluye tareas diseñadas específicamente para copiar objetos de bases de datos de SQL Server como tablas, vistas y procedimientos almacenados, para copiar objetos de SQL Server como bases de datos, inicios de sesión y estadísticas, y para agregar, modificar y eliminar objetos y datos de SQL Server mediante instrucciones Transact-SQL.

La administración de un entorno de base de datos OLTP u OLAP suele incluir la carga de datos. Integration Services incluye varias tareas que facilitan la carga masiva de datos. Puede utilizar una tarea para cargar datos de archivos de texto directamente en tablas y vistas de SQL Server, o puede usar un componente de destino para cargar datos en tablas y vistas de SQL Server después de aplicar transformaciones a los datos de la columna.

Un paquete de Integration Services puede ejecutar otros paquetes. Una solución de transformación de datos con muchas funciones administrativas puede separarse en varios paquetes de forma que resulte más sencillo administrar y reutilizar los paquetes.

Si necesita realizar las mismas funciones administrativas en distintos servidores, puede utilizar paquetes. Un paquete puede usar un bucle para recorrer los servidores y realizar las mismas funciones en varios equipos. Integration Services proporciona un enumerador que recorre los objetos de administración de SQL Server (SMO), como ayuda para la administración de SQL Server. Por ejemplo, un paquete puede usar el enumerador de SMO para realizar las mismas funciones administrativas en cada trabajo de la colección **Jobs** de una instalación de SQL Server.

Los paquetes de SSIS también pueden programarse con trabajos del Agente SQL Server.

2.1.5 Arquitectura de Integration Services

SSIS se compone de cuatro partes clave: el servicio Integration Services, el modelo de objetos de Integration Services, el tiempo de ejecución y los ejecutables de tiempo de ejecución de Integration Services, y la tarea Flujo de datos que encapsula el motor de flujo de datos y los componentes de flujo de datos.

En el diagrama siguiente se muestra la relación entre las partes.

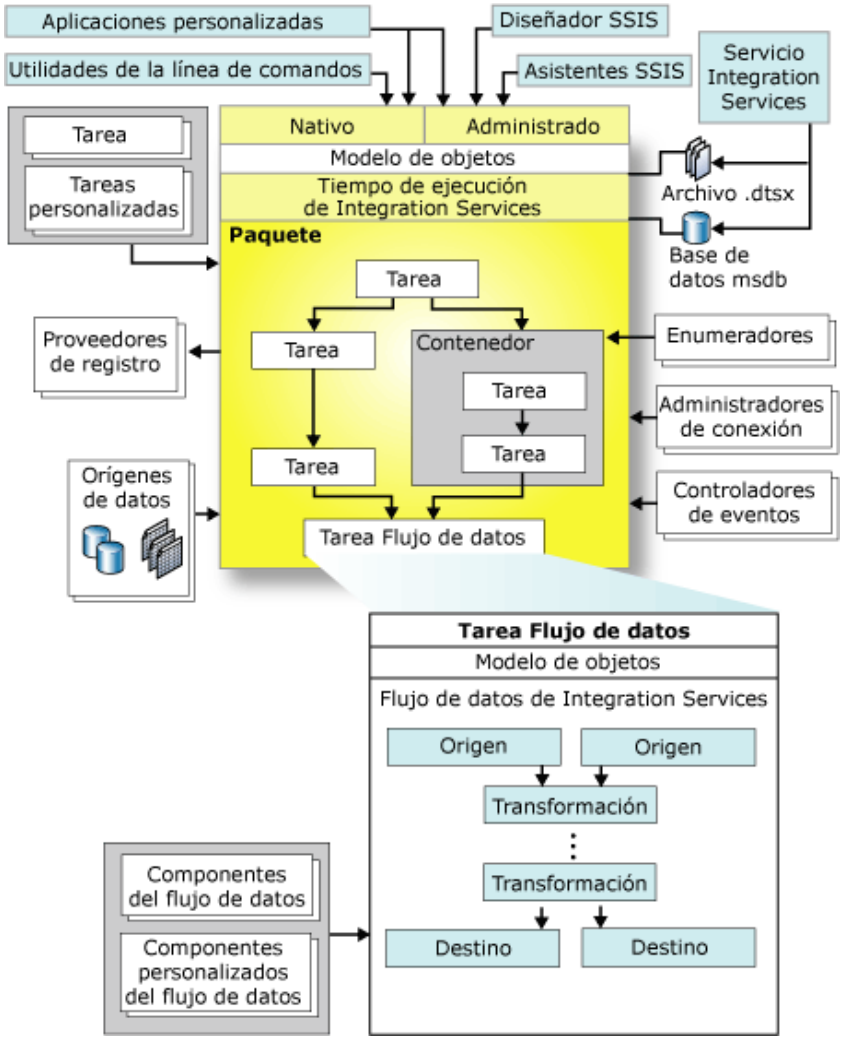

Figura 6.0. Arquitectura de Integration Services.

Los programadores que obtienen acceso al modelo de objetos de Integration Services desde clientes personalizados o escriben tareas o transformaciones personalizadas pueden escribir código mediante cualquier lenguaje compatible con Common Language Runtime (CLR).

2.1.6 Servicio Integration Services

El servicio Integration Services, disponible en SQL Server Management Studio, supervisa la ejecución de paquetes de Integration Services y administra el almacenamiento de paquetes.

2.1.7 Modelo de objetos de Integration Services

El modelo de objetos de Integration Services incluye interfaces de programación de aplicaciones (API) administradas para obtener acceso a las herramientas, utilidades de línea de comandos y aplicaciones personalizadas de Integration Services.

2.1.8 Tiempo de ejecución de Integration Services

El tiempo de ejecución de Integration Services guarda el diseño de paquetes, ejecuta paquetes y admite registros, puntos de interrupción, configuración, conexiones y transacciones. Los ejecutables de tiempo de ejecución de Integration Services son el paquete, los contenedores, las tareas y los controladores de eventos que incluye Integration Services, así como las tareas personalizadas.

2.1.9 Flujo de datos de Integration Services

La tarea Flujo de datos encapsula el motor de flujo de datos. El motor de flujo de datos proporciona los búferes en memoria que mueven datos desde el origen hasta el destino y llama los orígenes que extraen datos de archivos y bases de datos relacionales. El motor de flujo de datos también administra las transformaciones que modifican datos, así como los destinos que cargan datos o ponen datos a disposición de otros procesos. Los componentes de flujo de datos de Integration Services son los orígenes, transformaciones y destinos. También puede incluir componentes personalizados en un flujo de datos.

2.2 SQL SERVER ANALYSIS SERVICES

Microsoft SQL Server 2005 Analysis Services (SSAS) ofrece funciones de procesamiento analítico en línea (OLAP) y minería de datos para aplicaciones de Business Intelligence. Analysis Services admite OLAP y permite diseñar, crear y administrar estructuras multidimensionales que contienen datos agregados desde otros orígenes de datos, como bases de datos relacionales. En el caso de las aplicaciones de minería de datos, Analysis Services permite diseñar, crear y visualizar modelos de minería de datos que se construyen a partir de otros orígenes de datos mediante el uso de una gran variedad de algoritmos de minería de datos estándar del sector.

2.2.1 Arquitectura de Analysis Services

SSAS utiliza componentes de servidor y de cliente para proporcionar la funcionalidad de procesamiento analítico en línea (OLAP) y de minería de datos para aplicaciones de Business Intelligence:

El componente de servidor de Analysis Services se implementa como servicio de Microsoft Windows. SSAS admite varias instancias en el mismo equipo, con cada instancia de Analysis Services implementada como instancia independiente del servicio de Windows.

Los clientes se comunican con Analysis Services mediante el estándar público XML for Analysis (XMLA), protocolo basado en SOAP para emitir comandos y recibir respuestas, que se expone como servicio Web. Además, se proporcionan modelos de objetos de cliente en XMLA, a los que se puede obtener acceso mediante un proveedor administrado, como ADOMD.NET, o un proveedor OLE DB nativo.

Pueden emitirse comandos de consulta mediante los siguientes lenguajes: SQL; MDX (Expresiones multidimensionales), un lenguaje de consulta estándar para el análisis; o Extensiones de minería de datos (DMX), un lenguaje de consulta estándar orientado a la minería de datos. También se puede utilizar el lenguaje ASSL (Analysis Services Scripting Language) para administrar objetos de base de datos de Analysis Services.

Analysis Services también admite un motor de cubo local que permite a las aplicaciones en clientes desconectados examinar datos multidimensionales almacenados localmente

2.2.2 Arquitectura de servidor (Analysis Services)

El componente de servidor de SSAS es la aplicación msmdsrv.exe, que por lo general se ejecuta como un servicio de Windows. Esta aplicación está formada por componentes de seguridad, un componente que escucha XML for Analysis (XMLA), un componente de procesador de consultas y otros componentes internos que realizan las siguientes funciones:

- Analizar instrucciones recibidas de clientes
- Administrar metadatos
- Controlar transacciones
- Procesar cálculos
- Almacenar datos de celdas y dimensiones
- Crear agregaciones
- Programar consultas
- Almacenar objetos en la caché
- Administrar recursos del servidor

2.2.3 Arquitectura de cliente (Analysis Services)

SSAS admite una arquitectura de cliente ligero. El motor de cálculo de Analysis Services depende totalmente del servidor, por lo que todas las consultas se resolverán en él. En consecuencia, para cada consulta sólo se necesita realizar un viaje de ida y vuelta entre el cliente y el servidor, lo que produce un rendimiento escalable a medida que las consultas aumenten en complejidad.

El protocolo nativo para Analysis Services es XML for Analysis (XML/A). Analysis Services proporciona varias interfaces de acceso a datos para aplicaciones cliente pero todos estos componentes se comunican con una instancia de Analysis Services a través de XML for Analysis.

Se proporcionan varios proveedores distintos en Analysis Services para admitir diferentes lenguajes de programación. Un proveedor se comunica con un servidor de Analysis Services enviando y recibiendo XML for Analysis en paquetes SOAP sobre TCP/IP o sobre HTTP a través de Servicios de Internet Information Server (IIS). La conexión HTTP utiliza un objeto COM, denominado bombeo de datos y cuya instancia ha sido creada por IIS, que actúa como conducto para los datos de Analysis Services. El bombeo de datos no examina de ningún modo los datos subyacentes contenidos en la secuencia HTTP, ni ninguna de las estructuras de datos subyacentes está disponible para el código en la propia biblioteca de datos.

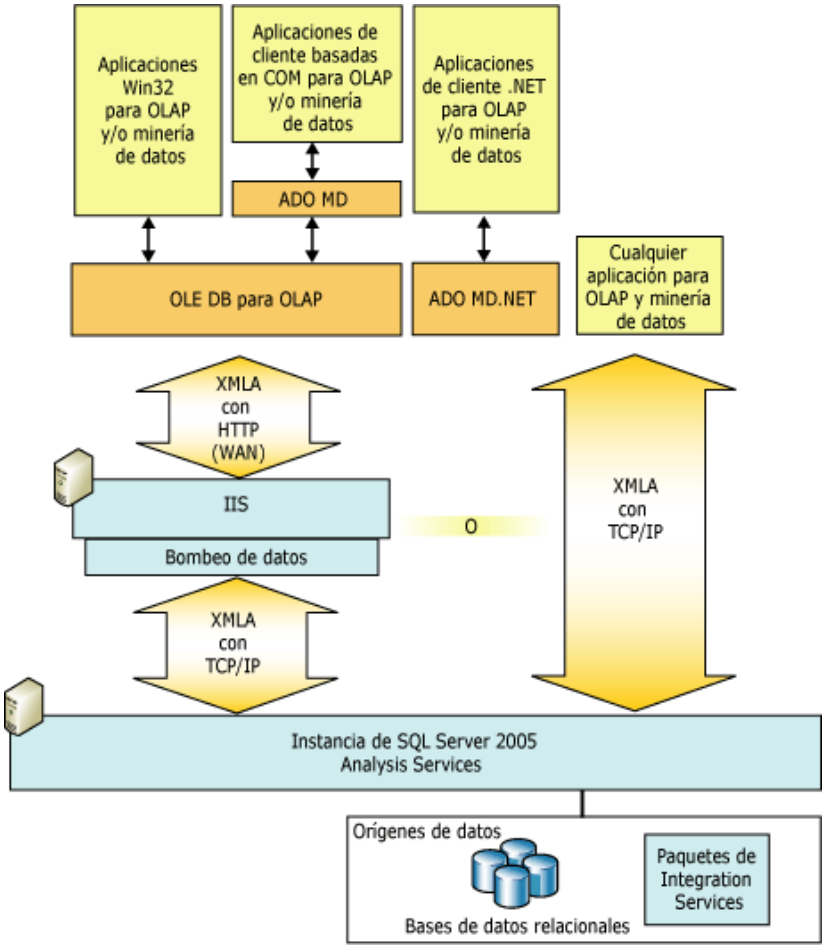

Figura 7.0. Arquitectura de cliente (Analysis Services).

Las aplicaciones cliente de Win32 pueden conectarse con un servidor de Analysis Services mediante interfaces OLE DB para OLAP o mediante el modelo de objetos Microsoft ActiveX Data Objects (ADO) para lenguajes de automatización COM (Modelo de objetos componentes) como, por ejemplo, Microsoft Visual Basic. Las aplicaciones codificadas con lenguajes .NET se pueden conectar con un servidor de Analysis Services mediante ADO MD.NET.

Analysis Services tiene una arquitectura Web con un nivel medio completamente escalable para implementación en organizaciones pequeñas y medianas. Analysis Services ofrece una amplia compatibilidad de nivel medio para servicios Web. Las aplicaciones ASP son compatibles con OLE DB para OLAP y ADO MD, y las aplicaciones ASP.NET son compatibles con ADOMD.NET. El nivel medio, que viene ilustrado en la siguiente figura, es escalable para muchos usuarios simultáneos.

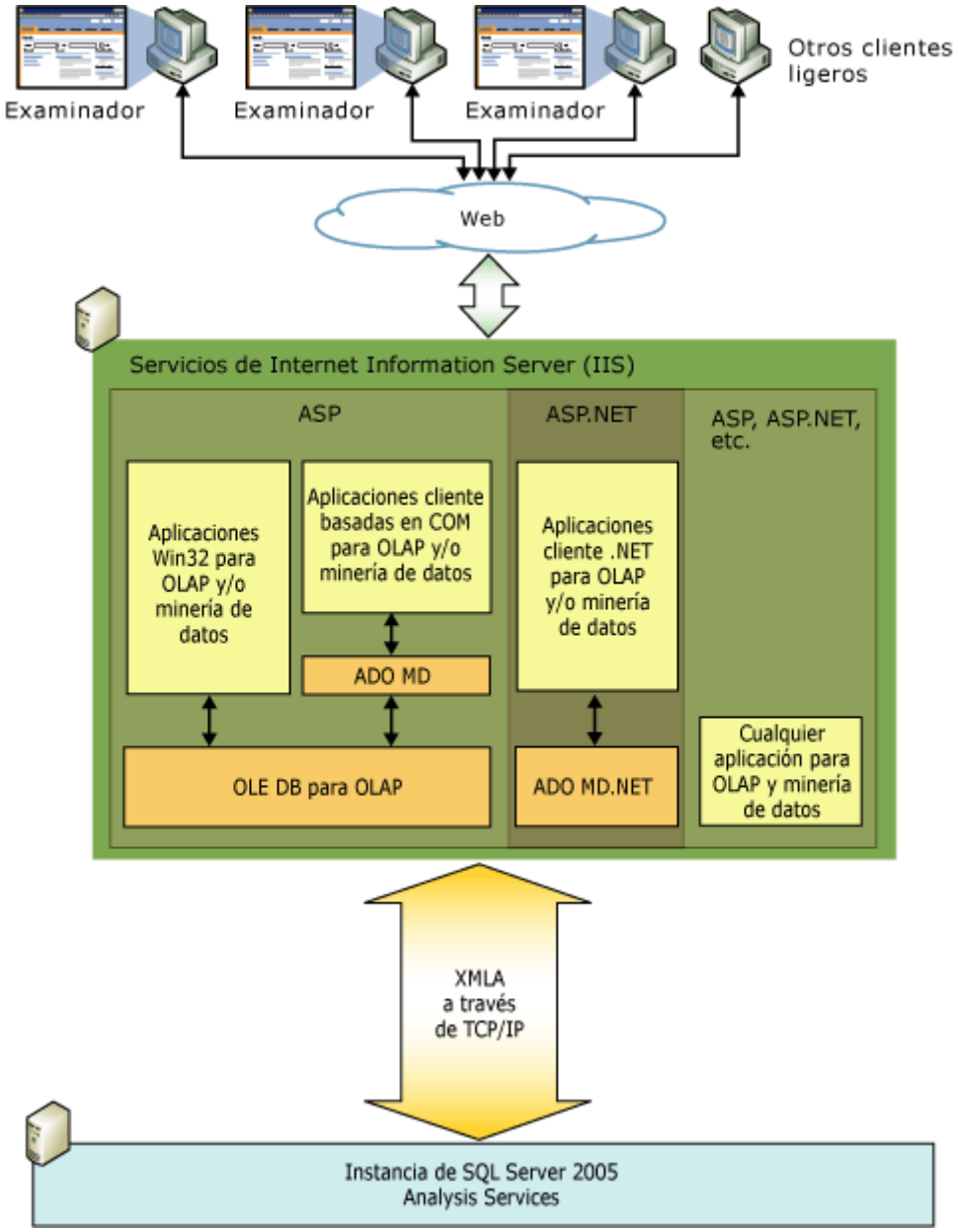

Figura 7.1. Arquitectura Web de Analysis Services

Las aplicaciones cliente y de nivel medio pueden comunicarse directamente con Analysis Services sin necesidad de ningún proveedor. Las aplicaciones cliente y de nivel medio pueden enviar XML for Analysis en paquetes SOAP sobre TCP/IP, HTTP o HTTPS. El cliente puede estar codificado con cualquier lenguaje compatible con SOAP. La comunicación se administra mucho más fácilmente en este caso a través de los Servicios de Internet Information Server (IIS) mediante HTTP, aunque también puede codificarse una conexión directa con el servidor mediante TCP/IP. Se trata de la solución de cliente más ligero para Analysis Services.

2.2.4 Conceptos de Analysis Services

SSAS proporciona funciones de procesamiento analítico en línea (OLAP) y minería de datos para soluciones de Business Intelligence. Antes de diseñar una solución de Business Intelligence mediante Analysis Services, debería familiarizarse con los conceptos de OLAP y minería de datos necesarios para crear una solución correctamente.

Analysis Services combina los mejores aspectos del análisis tradicional basado en OLAP y la elaboración de informes basada en relaciones al permitir a los programadores definir un único modelo de datos, denominado Unified Dimensional Model (UDM), a partir de uno o más orígenes de datos físicos. Todas las consultas de usuario final desde aplicaciones OLAP, de elaboración de informes y de BI personalizadas obtienen acceso a los orígenes de datos subyacentes a través del modelo UDM, que proporciona una única vista empresarial de estos datos relacionales.

Analysis Services proporciona un amplio conjunto de algoritmos de minería de datos para permitir a los usuarios empresariales recopilar los datos mediante la búsqueda de patrones y tendencias específicos. Estos algoritmos de minería de datos se pueden utilizar para analizar los datos a través de un modelo UDM o directamente a partir de un almacén de datos físico.

2.2.5 Objetos de Analysis Services

Una instancia de SSAS contiene ensamblados y objetos de base de datos para su uso con procesamiento analítico en línea (OLAP) y minería de datos.

Las bases de datos contienen objetos OLAP y de minería de datos como orígenes de datos, vistas de origen de datos, cubos, medidas, grupos de medida, dimensiones, atributos, jerarquías, estructuras de minería de datos, modelos de minería de datos y funciones.

Los ensamblados contienen funciones definidas por el usuario que amplían la funcionalidad de las funciones intrínsecas suministradas por los lenguajes Expresiones multidimensionales (MDX) y Extensiones de minería de datos (DMX).

2.3 SQL SERVER REPORTING SERVICES

Microsoft SQL Server 2005 Reporting Services (SSRS) ofrece funcionalidad empresarial de informes habilitados para Web con el fin de poder crear informes que extraigan contenido a partir de una variedad de orígenes de datos, publicar informes con distintos formatos y administrar centralmente la seguridad y las suscripciones.

2.3.1 Presentación de Reporting Services

SSRS es una plataforma de elaboración de informes basada en servidor que puede utilizar para crear y administrar informes tabulares, matriciales, de gráficos y de formato libre con datos extraídos de orígenes de datos relacionales y multidimensionales. Los informes que cree se pueden visualizar y administrar mediante una conexión basada en Web.

Reporting Services contiene los componentes principales siguientes:

- Un conjunto completo de herramientas que se pueden utilizar para crear, administrar y ver informes.
- Un componente Servidor de informes que aloja y procesa informes en diversos formatos. Los formatos de salida incluyen HTML, PDF, TIFF, Excel, CSV, etc.
- Una API que permite a los programadores integrar o extender procesamientos de datos e informes en aplicaciones personalizadas o crear herramientas personalizadas para generar y administrar informes.

Los informes que se generan pueden basarse en datos relacionales o multidimensionales de SQL Server, Analysis Services, Oracle o cualquier proveedor de datos de Microsoft .NET, como ODBC u OLE DB. Puede crear informes tabulares, matriciales o de formato libre. También puede crear informes adaptados a una circunstancia determinada, que utilicen modelos y orígenes de datos predefinidos.

Visual y funcionalmente, los informes que se generan en Reporting Services van más allá de los informes tradicionales, puesto que incluyen características interactivas y basadas en Web. Algunos ejemplos de estas características son los informes de varios niveles de detalle, que permiten la navegación por distintas capas de datos, los informes con parámetros, que admiten el filtro de contenido en tiempo de ejecución, o los informes de formato libre, que admiten contenido con diseños verticales, anidados o adyacentes, vínculos a contenido o recursos basados en Web y acceso seguro y centralizado a informes mediante conexiones Web locales o remotas.

Aunque Reporting Services se integra con otras tecnologías de Microsoft sin necesidad de adaptación, los programadores y otros proveedores pueden generar componentes compatibles con otros formatos de salida de informes, formatos de entrega, modelos de autenticación y tipos de orígenes de datos. La arquitectura de desarrollo y tiempo de ejecución se ha creado con un diseño modular para admitir extensiones de terceros y posibilidades de integración.

2.3.2 Información general de componentes de Reporting Services

SSRS es un conjunto de componentes de procesamiento, herramientas e interfaces de programación que permiten el desarrollo y la utilización de informes completos en un entorno administrado. El conjunto de herramientas incluye herramientas de desarrollo, de configuración y de administración así como herramientas de visualización de informes. Las interfaces de programación incluyen el protocolo simple de acceso a objetos (SOAP), los extremos de direcciones URL e Instrumental de administración de Windows (WMI), para permitir una fácil integración con aplicaciones y portales nuevos o existentes.

El procesamiento se distribuye en múltiples componentes. Para recuperar datos, procesar el diseño de los informes, representar los formatos de presentación y entregar en destinos específicos se utilizan procesadores centralizados y especializados. El procesamiento de una presentación tiene lugar después de recuperar los datos y es independiente del procesamiento de los datos, lo que permite a diversos usuarios consultar el mismo informe simultáneamente en formatos diseñados para distintos servicios o cambiar inmediatamente el formato de visualización del informe, de HTML a PDF, a Microsoft Excel o a XML, con un solo clic. La arquitectura modular se ha diseñado para permitir ampliaciones. Los programadores pueden incluir funciones de informes en aplicaciones personalizadas o ampliar la funcionalidad para hacerla compatible con características personalizadas.

El diagrama siguiente muestra los componentes y las herramientas de Reporting Services. El diagrama también muestra cómo se adaptan las herramientas personalizadas al diseño global. Presenta el flujo de solicitudes y datos entre componentes del servidor y los componentes que envían y recuperan contenido de un almacén de datos.

2.3.3 Diagrama de la arquitectura de Reporting Services

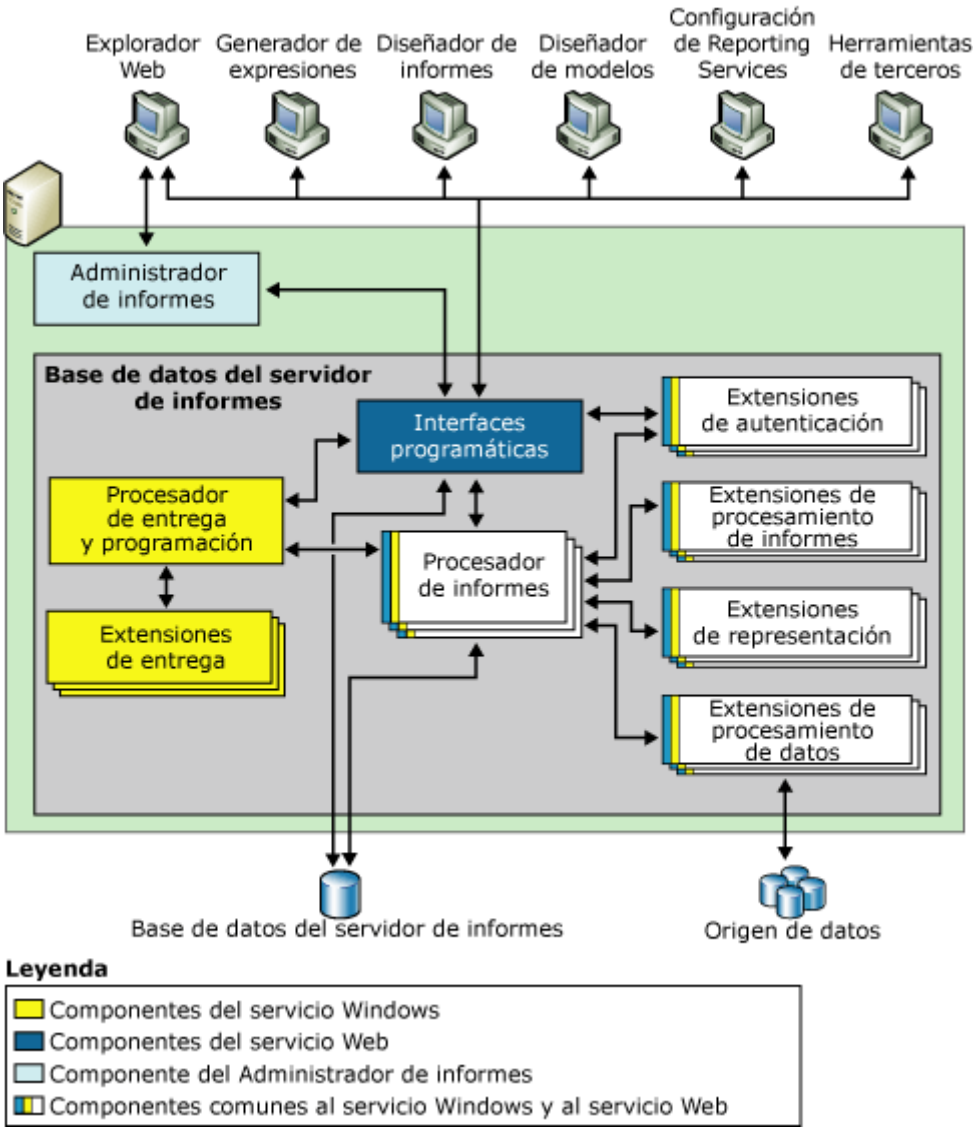

Figura 8.0. Arquitectura de Reporting Services.

2.3.4 Implementar Reporting Services
2.3.5 Planear una implementación de Reporting Services

SSRS ofrece dos modelos de implementación:

Una implementación estándar consiste en una instancia de servidor de informes que utiliza un motor de base de datos de SQL Server local o remoto para alojar la base de datos del servidor de informes. Se puede utilizar SQL Server 2000 o SQL Server 2005 para alojar la base de datos del servidor de informes.

Una implementación escalada consiste en varios servidores de informes que comparten una sola base de datos del servidor de informes. La base de datos se puede instalar en una instancia remota de SQL Server o localmente en uno de los servidores de informes. La instancia de SQL Server que aloja la base de datos del servidor de informes puede ser parte de un clúster de conmutación por error.

Las siguientes ediciones son compatibles con la implementación escalada: Enterprise Edition, Developer Edition y Evaluation Edition.

Para simplificar el proceso de implementación, puede utilizar listas de comprobación que describan la secuencia de tareas que se deben realizar para llevar a cabo una implementación estándar.

2.3.6 Implementación estándar

El siguiente diagrama muestra el modelo de implementación estándar, con la base de datos de servidor de informes ubicada en un servidor remoto. También puede instalarla localmente para que todos los componentes de servidor estén en el mismo equipo.

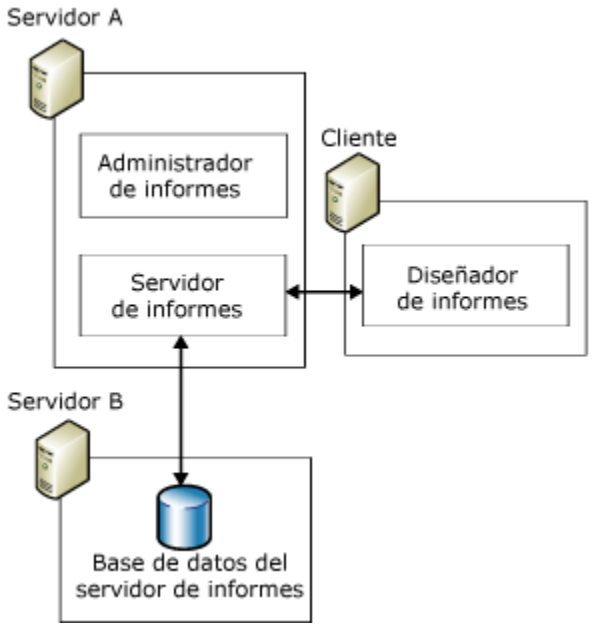

Figura 8.1. Representación estándar de Reporting Services.

Las principales consideraciones al elegir dónde alojar la base de datos del servidor de informes son:

- Recursos de procesamiento
- Disponibilidad de espacio en disco

El servidor de informes y el motor de base de datos compiten por recursos de procesamiento como el tiempo de CPU, la memoria y el acceso al disco. Algunas operaciones de servidor de informes consumen gran cantidad de recursos. Por ejemplo, un servidor de informes intenta utilizar toda la memoria disponible para operaciones de representación de informes. La competición por los recursos de procesamiento se puede reducir si el servidor de informes se ejecuta en un equipo independiente.

Los requisitos de espacio en disco del servidor de informes son la segunda razón por la que se debe utilizar un motor de base de datos de SQL Server remoto para almacenar datos del servidor de informes. Aunque el volumen de una base de datos de servidor de informes puede ser reducido al principio, los requisitos de espacio en disco pueden aumentar notablemente en tiempo de ejecución en función de cómo ejecute los informes y del número de usuarios que tengan acceso al servidor de informes.

2.3.7 Implementación escalada

Puede implementar Reporting Services de forma escalada para crear una instalación de servidor de informes altamente disponible y escalable. Configurar una implementación escalada también puede ser de utilidad si se desea mejorar el rendimiento de las operaciones programadas y la entrega de suscripciones. Una implementación escalada de servidor de informes consta de varios servidores de informes que comparten una sola base de datos de servidor de informes. Cada servidor de informes de la implementación se denomina *nodo*. Los nodos participan en la implementación escalada si el servidor de informes se configura para utilizar la misma base de datos que otro servidor de informes.

Es posible equilibrar la carga de los nodos de servidor para admitir un gran volumen de informes. Asimismo, la base de datos del servidor de informes se puede crear en un clúster de conmutación por error si es necesario cumplir requisitos de alta disponibilidad.

Entre las configuraciones de clúster que no se admiten se encuentra la implementación de una instalación de servidor de informes completa, es decir, un servidor de informes y su base de datos, en cada nodo de un clúster de varios nodos. Concretamente, no se puede implementar Reporting Services en un clúster de dos nodos formado por un nodo activo y un nodo pasivo que se utiliza cuando se produce un error en el nodo activo.

2.3.8 Implementación en un clúster con equilibrio de carga de red (NLB)

Se pueden ejecutar nodos de servidor de informes en un clúster NLB. Para implementar el clúster NLB, se puede utilizar una solución de software o hardware. Para ejecutar los servidores de informes en un clúster NLB, se debe utilizar software y herramientas compatibles con esa funcionalidad. Reporting Services no proporciona ningún método para administrar clústeres de servidores ni servidores virtuales, ni tampoco existe ninguna forma de definir un nombre de servidor virtual que proporcione un único punto de entrada para todos los nodos de una implementación escalada de servidor de informes.

NLB sólo es necesario si se desea mejorar el rendimiento del servidor de informes para los informes a petición y los informes interactivos, como es el caso de los informes de obtención de detalles y de matriz. Los informes programados y el procesamiento de las suscripciones son más rápidos en una implementación escalada, pero no requieren necesariamente un clúster NLB para que el rendimiento sea superior.

2.3.9 Implementación en un clúster de conmutación por error de SQL Server

SQL Server 2005 admite el uso de clústeres de conmutación por error para que se puedan utilizar múltiples discos para una o varias instancias de SQL Server. El uso de clústeres de conmutación por error sólo se admite para la base de datos del servidor de informes; no se puede ejecutar el servicio Web del servidor de informes o el servicio Servidor de informes de Windows como parte de un clúster de conmutación por error.

Para alojar una base de datos de servidor de informes en un clúster de conmutación por error de SQL Server, el clúster debe estar previamente instalado y configurado. Después, se puede seleccionar el clúster de conmutación por error como el nombre del servidor al crear la base de datos del servidor de informes.

Aunque el servicio Web del servidor de informes y el servicio Servidor de informes de Windows no pueden participar en un clúster de conmutación por error, es posible instalar Reporting Services en un equipo que tenga instalado un clúster de conmutación por error de SQL Server. El servidor de informes se ejecuta de manera independiente del clúster de conmutación por error. Si se instala un servidor de informes en un equipo que forma parte de una instancia de conmutación por error de SQL Server, no es obligatorio utilizar el clúster de conmutación por error para la base de datos del servidor de informes, sino que, para alojarla, se puede utilizar otra instancia de SQL Server.

2.3.10 Diagrama de implementación escalada

El siguiente diagrama muestra varios servidores de informes y bases de datos de servidor de informes implementados en clústeres de servidores distintos.

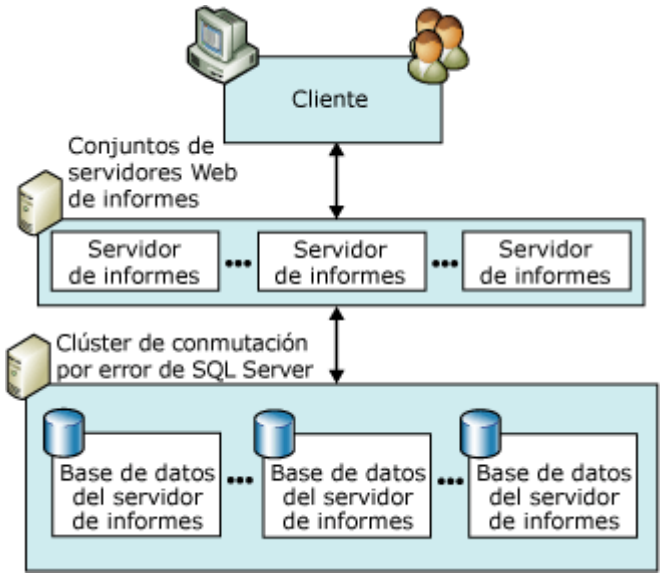

Figura 8.2. Diagrama de implementación escalada.

2.3.11 Métodos para crear un informe

Reporting Services le permite crear informes mediante el Generador de informes o el Diseñador de informes.

2.3.12 Crear informes con el generador de informes

Para crear informes puntuales, utilice el Generador de informes. El Generador de informes es una aplicación de ClickOnce Windows Forms que los usuarios descargan a sus equipos locales desde el servidor de informes. Permite a los usuarios crear informes arrastrando los campos desde modelos de informes predefinidos hasta una plantilla de diseño de informe prediseñada. Los usuarios pueden formatear, agrupar, clasificar y filtrar los datos. Además, pueden modificar o definir fórmulas. Con el Generador de informes, los usuarios no necesitan entender la estructura subyacente del origen de datos ni ningún lenguaje de computación complejo. Simplemente deben estar familiarizados con los datos de sus orígenes de datos.

2.3.13 Crear informes con el diseñador de informes

Para crear informes más complejos, utilice el Diseñador de informes. El Diseñador de informes admite una amplia variedad de características de creación de informes, incluidas características programables y personalizables. Los informes creados en el Diseñador de informes pueden ser informes libres o muy estructurados, sencillos o muy complejos. Posee un control total sobre el diseño y puede agregar características avanzadas tales como expresiones, ensamblados personalizados que se ejecutan desde el informe e interacción con el informe para ver detalles y realizar vinculaciones con los datos correspondientes. También puede crear informes básicos formados por tablas, imágenes y listas sencillas.

En el Diseñador de informes, se puede crear un informe de tres maneras. Se puede crear un informe en blanco y agregar manualmente las consultas y el diseño. Se puede utilizar el Asistente para informes, que crea automáticamente un informe tabular o matricial basado en la información proporcionada. También se puede importar un informe existente de Microsoft Access.

Los informes se publican en el servidor de informes como archivos de definición de informe (.rdl). Una definición de informe es un documento XML, por lo que puede crear y editar informes con otras herramientas además de con el Diseñador de informes. Es posible modificar una definición de informe utilizando un editor de texto o una herramienta de otro fabricante diseñada para modificar informes escritos en el lenguaje RDL (Report Definition Language). El Diseñador de informes utiliza la API del Protocolo simple de acceso a objetos (SOAP) de Reporting Services para publicar los informes en un servidor de informes. Si utiliza otra herramienta para crear archivos .rdl que no publica informes directamente en un servidor de informes, puede cargarlos con el Administrador de informes.

CONCLUSIONES

Este trabajo de tesis fue dividido en dos capítulos, el primer capitulo describe un resumen de la información obtenida en el diplomado, el segundo capitulo contiene información de la tecnología de Microsoft SQL Server 2005 Business Intelligent y los componentes que conforman.

Al tomar este diplomado adquirí los conocimientos y habilidades necesarias para implementar, optimizar, administrar y mantener un servidor de base de datos, además aprendí el lenguaje SQL para realizar consultas y explotar los datos de las bases de datos.

Aprendí las principales taras del administrador de base de datos para llevarlas a la práctica.

Diseñar y crear bases de datos, manejar grandes volúmenes de información para la oportuna toma de decisiones que requiere cualquier organización.

También poder seleccionar la plataforma de trabajo adecuada a las necesidades de la empresa con las características del manejador de base de datos así como establecer las políticas de seguridad y planes de contingencia.

El proceso de Business Intelligence, ha comenzado a tener un auge importante en la mayoría de las empresas y debido a la gran cantidad de información que tienen es necesario la implementación de estas estrategias para poder realizar tomas decisión, incrementado la efectividad de la empresa

SQL Server 2005, es una plataforma que permite el análisis y la administración de datos empresarial, con herramientas de Business Intelligence, como Integration Services que permite generar soluciones de integración de datos de alto rendimiento, Analisys Services ofrece funciones de procesamiento analítico en línea OLAP y minería de datos, Reporting Services que ofrece funcionalidad para el desarrollo de informes obteniendo datos de una gran variedad de orígenes.

BIBLIOGRAFÍA

[2005] Administración de Base de datos
Universidad Nacional Autónoma de México
Dirección General de Servicios de Cómputo Académico

[Microsoft 2007] Documentación de SQL Server 2005
Microsoft Corporation 2007
http://technet.microsoft.com/es-es/library/ms203721.aspx

[Jacobson 2005] SQL SERVER 2005 ANALYSIS SERVICES
Reed Jacobson, Stacia Misner
Hitachi Consulting
Microsoft Press 2005

[Jacobson 2005] SQL SERVER 2005 REPORTING SERVICES
Reed Jacobson, Stacia Misner
Hitachi Consulting
Microsoft Press 2005

REFERENCIAS

[Ferré 2004] Desarrollo Orientado a Objetos con UML
Xavier Ferré Grau y María Isabel Sánchez Segura, 2004
http://www.clikear.com/manuales/uml/introduccion.asp

[Avenue 1995] Sybase SQL Server (Guía de Administración del Sistema)
Christie Avenue, Emeryville 1995
http://manuals.sybase.com/onlinebooks/group-asarc/svs11001/sagsp/

[Monografías 1997] El Centro de Tesis, Documentos, Publicaciones y Recursos Educativos más amplio de la Red. 1997 Monografias.com S.A.
http://www.monografias.com/trabajos19/administracion-base-datos/administracion-base-datos.shtml