UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA

"ANÁLISIS PARA LA MIGRACIÓN DE UNA BASE DE DATOS Y APLICACIÓN EN UN SISTEMA INSTITUCIONAL"

TESIS

QUE PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTAN

ALICIA CORTES OLIVARES

ERNESTO SALAZAR RODRÍGUEZ

DIRECTOR DE TESIS
M.I. EDUARDO ALARCÓN AVILA

MÉXICO, D.F., JUNIO DE 2006

A mis padres:

Gracias por el apoyo incondicional que siempre me han brindado.

Alicia

A mis padres que siempre me apoyaron durante mis años de estudiante,

Muchas Gracias.

Ernesto

A la Facultad de Ingeniería de la Universidad Nacional Autónoma de México:

Gracias por brindarnos un espacio en su alumnado.

INDICE

OBJETIVO

El objetivo de la presente tesis es exponer el procedimiento definido y las consideraciones específicas que se observaron para migrar una base de datos de Informix SE2000 a Oracle 10gi.

INTRODUCCION

El proceso de migración de información de un motor de base de datos a otro presenta retos que pueden ir desde el desconocimiento de la herramienta destino hasta la imposibilidad de contar con herramientas *case* de apoyo a la tarea por desarrollar, ademas de la falta de personal que conozca las aplicaciones asociadas a la base de datos, y tiempos insuficientes en comparación con la complejidad de la tarea en cuestión.

Cambiar el motor de base de datos requerirá a la organización invertir en capacitación para el personal en el uso de las herramientas asociadas a la nueva base de datos; será necesario también, dedicar tiempo a tareas diferentes de los objetivos sustantivos del negocio.

En la presente tesis se expone el proceso de migración de una base de datos de uso cotidiano en una Institución de salud pública; dicha base de datos soporta los procesos de Administración de Abastecimientos, principalmente insumos farmacéuticos y médicos.

Oracle 10gi es la herramienta definida por el área de innovación tecnológica como parte de la estrategia institucional de migración a nuevas tecnologías. Condición inicial para el proceso de migración es que todas las aplicaciones dependientes de la base de datos original deberán seguir funcionando sin cambio operativo o de presentación al cambiar de motor de datos.

Las aplicaciones asociadas a la base de datos que se desea migrar están desarrolladas en Delphi 5. El área de sistemas de la dependencia ha determinado que al momento de migrar la base de datos no se modificará la herramienta de desarrollo, dado que las aplicaciones del sistema se han venido construyendo durante los últimos cinco años.

En el aspecto técnico podemos citar las diferencias entre los motores de base de datos relacionadas a la organización física de los archivos que la integran, los modelos de conexión, el manejo de transacciones, el uso de tablas temporales, el código de los programas almacenados, funciones, vistas, y por las utilerías de descarga y carga de información de la base de datos original hacia la base de datos destino.

En el aspecto administrativo, los tiempos límite, ausencia o inexistencia de documentación de sistemas, configuraciones, etc., incrementa el nivel de dificultad del proceso.

Es importante considerar que estas circunstancias pueden ser superadas por un grupo de trabajo altamente comprometido y orientado a resultados.

Este es el entorno de trabajo en el que se busca alcanzar el objetivo de la presente tesis. A continuación se resume el contenido de los capítulos integrados en este documento.

Capítulo I. Antecedentes teóricos, en este se describen los conceptos básicos involucrados como la Ingeniería de Programación, los modelos de desarrollo de proyectos de software; cabe mencionar que en este trabajo no es un proyecto de desarrollo de software sino la migración de una base de datos de un motor a otro; en este caso de Informix a Oracle.

Se incluyen también los conceptos más usuales de base de datos; en el apartado de Redes LAN se mencionan brevemente los conceptos básicos sobre infraestructura de comunicaciones; las características de la arquitectura cliente/servidor, que corresponde a la configuración del sistema y la base de datos a migrar.

Capítulo II.- Análisis del problema; se describe la funcionalidad del sistema a migrar, sus características actuales y cómo estas podrán ser modificadas durante y después de la migración, se define el problema, su origen y se mencionan las condiciones de desarrollo del trabajo.

Capítulo III.- Se ve más a detalle tanto la herramienta de migración como todos los supuestos a considerar relacionados al manejo de los datos a migrar, ya que hay diferencias entre como son definidos los datos en INFORMIX y como serán interpretados por Oracle.

Capítulo IV.- En caso de uso se tocará el tema de como migrar el esquema de base de datos, o por decirlo así, el esqueleto de la base de datos para luego migrar la información y finalmente las aplicaciones del sistema, en cada uno de estos puntos se verán los llamados puntos de revisión en donde se puede tomar la desición de modificar tipos de datos y de nueva cuenta realizar el proceso de migración, ya sea del esquema de la base de datos, los datos o la aplicación.

Capítulo IV.- Finalmente, en las conclusiones se da un resumen de lo que fueron y deben de ser las lecciones aprendidas despues del desarrollo de este proyecto, en este caso la migración de una base de datos.

CAPITULO I

ANTECEDENTES TEORICOS

1.1 Ingeniería de programación

Definición de sistema

Un sistema es un conjunto de componentes interrelacionados para desarrollar un trabajo específico.

En el caso de un sistema de cómputo, los elementos principales son: *hardware* y *software*; los componentes físicos de la computadora integran el hardware; el software está integrado por los programas de cómputo almacenados en la misma; sin embargo, el equipo de cómputo no tiene sentido a menos que exista un usuario que aproveche las capacidades del sistema.

Podemos mencionar que la fiabilidad es la característica más importante del sistema y se aplica por igual al hardware, al software y al operador.

Por otro lado, un sistema seguro es aquel que no permite accesos no autorizados a sus datos; sin embargo, es claramente imposible predecir todos los modos de acceso y prohibirlos de forma explícita.

Modelos del proceso del software

Un modelo del proceso del software es una representación abstracta de un proceso del software.

En esta sección se introducen varios modelos de proceso generales.

1. *El modelo de cascada*. Toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, representados como fases separadas del proceso, como especificación de requerimientos, diseño de software, implantación, pruebas, etcétera.

2. *Desarrollo evolutivo*. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un primer sistema se desarrolla rápidamente a partir de especificaciones abstractas. Evoluciona con la ayuda del cliente para producir un sistema que satisfaga sus necesidades.

3. *Desarrollo formal de sistemas*: Este enfoque se basa en la producción de una especificación matemática formal del sistema y en la transformación de esta especificación, utilizando métodos matemáticos para construir un programa. La verificación de los componentes del sistema se lleva a cabo generando argumentos matemáticos acordes a su especificación.

4. *Desarrollo basado en la reutilización*. Este enfoque se basa en la existencia de un número significante de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero.

Para el desarrollo de este proyecto nos basaremos en las características del modelo de desarrollo en cascada, aún cuando no se trata de un proyecto de desarrollo de software nuevo.

En general seguiremos el ciclo de vida de un desarrollo de software correspondiente al modelo de cascada (Diagrama 1.1.1).

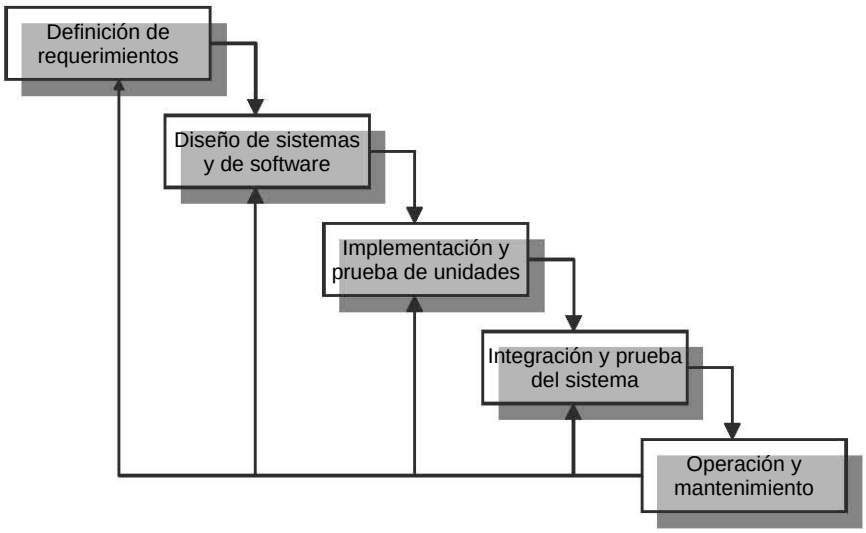


DIAGRAMA 1.1.1

1.2 Base de Datos

Una base de datos es una colección de información o datos organizados.

Por ejemplo, el directorio telefónico es una base de datos; sus datos son los nombres, direcciones, y números telefónicos de los residentes y empresas de una localidad. Otros ejemplos de base de datos son:

- ✓ Listas de correos
- ✓ Listas de inventarios
- ✓ Listas de clientes

Estas colecciones de datos se llaman base de datos porque están integradas por información que es útil a una organización en particular o por información utilizada para un propósito específico.

El directorio telefónico es un buen ejemplo de base de datos, veamos la Tabla 1.2.1.

Apellido	Nombre	Dirección	Teléfono
Pérez	Claudia	Africa 45	55-12-12-12
Pérez	Ernesto	Madero 17	55-16-16-16
Pérez	Laura	Palmas 15	55-11-11-11
Pérez	Luis	Hidalgo 97	55-15-15-15
Pérez	Raúl	Africa 34	55-14-14-14

Tabla 1.2.1

El directorio telefónico está organizado alfabéticamente por apellido.
Si se requiere llamar a Luis Pérez será necesario recorrer la lista por Apellido hasta llegar a la letra P.
Si se conoce el apellido de la persona a la que se desea llamar este esquema funciona perfectamente.

Pero, ¿qué sucede cuando se requiere encontrar los números de teléfono de quienes viven en la calle Africa?. ¿Qué pasa cuando se desconoce el apellido de la persona y sólo se tiene la dirección y el nombre? El directorio telefónico está organizado por apellido, no por dirección ni por nombre de manera que la búsqueda puede tomar más tiempo.

Justamente para resolver este tipo de necesidades es que existen en el mercado productos de software llamados Sistemas Administradores de Base de Datos DBMS por sus siglas en inglés (*Database Management System*).

Sistema administrador de base de datos

Un sistema de administración de base de datos es un producto de software que proporciona a los usuarios la posibilidad de controlar los accesos de lectura/escritura, generación de reportes específicos y análisis de utilización de la información.

También conocido como motor, aquí es donde va a residir de forma lógica y física la información de la base de datos, las reglas de negocio y las relaciones que asocian y jerarquizan a los datos.

Existe en el mercado una amplia gama de productos que son manejadores de base de datos como Oracle, Informix, DB2, SQL Server, etc. La elección de cuál usar depende de la cantidad y complejidad de los datos y sus relaciones, así como de la infraestructura existente y los recursos económicos de cada empresa o persona para adquirirlos y mantenerlos.

El Diagrama 1.2.1 muestra la representación simplificada de un sistema de bases de datos.

Representación simplificada de un sistema de bases de datos

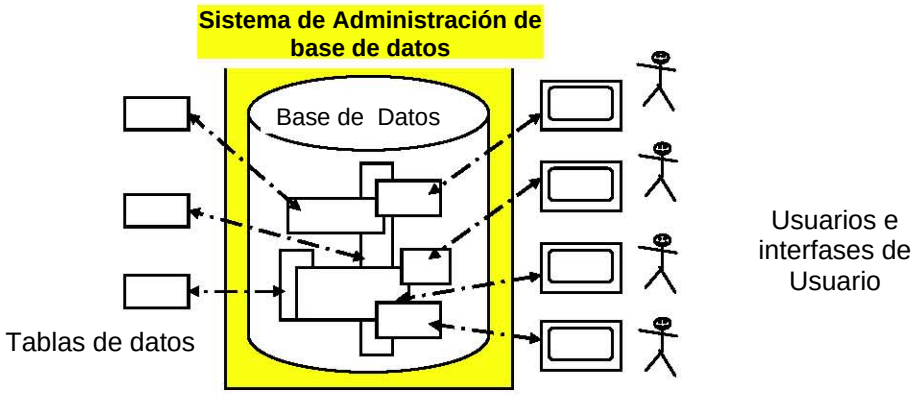


DIAGRAMA 1.2.1

Por la forma en que se organiza la información, las bases de datos pueden ser de tres diferentes tipos a) Relacionales, b) Jerárquicas o de c) Red.

El software DBMS específico para administrar una base de datos relacional se conoce genéricamente como Sistema Administrador de Base de Datos Relacionales RDBMS por sus siglas en inglés (Relational Database Management System).

Por ser el tipo de base de datos de uso más extendido y el empleado en el presente trabajo, nos dedicaremos especialmente a describir las características de una base de datos relacional y de un RDBMS.

Características de un RDBMS

En la Tabla 1.2.2 se detallan algunas de las características generales que cubre un RDBMS.

Administración del acceso a datos	Administración de la comunicación con el sistema operativo y organización, almacenamiento y recuperación de información.
Recuperación de caída	Protección ante fallas de hardware y errores provocados por los usuarios.
Recuperación de caída: Múltiples usuarios	Control de acceso concurrente para múltiples usuarios y restablecimiento de la información al instante anterior a la caída.
Múltiples aplicaciones	Control de accesos concurrentes de lectura y escritura a los datos, permitiendo la comunicación entre las diferentes aplicaciones.
Seguridad	Protección de accesos y control de operaciones de lectura y escritura basado en grupos y roles de usuarios y esquemas particulares de permisos para cada Usuario.
Integridad	Validación previa a la actualización de información, siguiendo las reglas de valor y relación entre los datos en una tabla y entre ellas.
Extensibilidad	Mecanismos que permiten extender la base de datos sin interrumpir su ejecución tales como creación de nuevas estructuras o código almacenado, respaldos incrementales, exportar información, carga de datos.
Distribución de datos	Distribuir los datos en diferentes lugares, organizaciones y plataformas de hardware.

1.2.1 Modelo relacional

En el modelo relacional una base de datos se organiza a partir de tablas; de manera que:

Una base de datos es una Colección de Tablas

Una base de datos contiene al menos dos tablas y puede contener tantas tablas como se requiera; si la base de datos contiene solo una tabla estaríamos hablando de un archivo aislado.

Cada tabla en una base de datos contiene un tipo específico de información, diferente del contenido en otras tablas.

Por ejemplo, es posible mantener separada la información de los productos que la empresa vende, las órdenes de compra y los clientes de la organización de manera que será necesario crear tablas separadas para cada uno de estos grupos de datos.

En cualquier momento es posible agregar o eliminar tablas

Una tabla está integrada por Renglones y Columnas

Una tabla es una estructura o herramienta conceptual cuya arquitectura está definida por renglones y columnas como se muestra en el Diagrama 1.2.1.1.

Cada *renglón* o *registro* contiene todos los datos de uno de los objetos que la tabla describe. Por ejemplo para una base de datos que almacena la información de una tienda.

- Cada registro en la tabla *cliente* contiene los datos de un cliente.
- Cada registro en la tabla *ordenes* contiene los datos de cada venta realizada por la compañía.
- Cada registro en la tabla *ítems* contiene los datos de cada producto requerido en una *orden.*

Y así podemos continuar para cada tabla de la base de datos del almacén.

Cada tabla se compone de una o más columnas

Una tabla incluye una o más *columnas*, una columna contiene un tipo particular de información, en el caso de la tabla de *clientes* las columnas pueden ser el apellido, la dirección, el teléfono, etc.; para la tabla de productos las columnas pueden ser la clave, la descripción, el precio unitario, la presentación, etc.

El nombre de cada columna se asigna cuando se crea la tabla en la base de datos.

En algunos DBMS los renglones se conocen como filas, tuplas o registros y las columnas como campos o atributos.

Por ejemplo, las columnas de la tabla Cliente son las siguientes (Tabla 1.2.1.1)

Columna	Información que contiene
num_cte	Número de Cliente
apellido	Apellido del Cliente
nombre	Nombre(s) del Cliente
telefono	Teléfono del Cliente

Tabla 1.2.1.1 Columnas de la tabla Cliente

En el Diagrama 1.2.1.1 se muestran registros o renglones y columnas de la tabla Cliente.

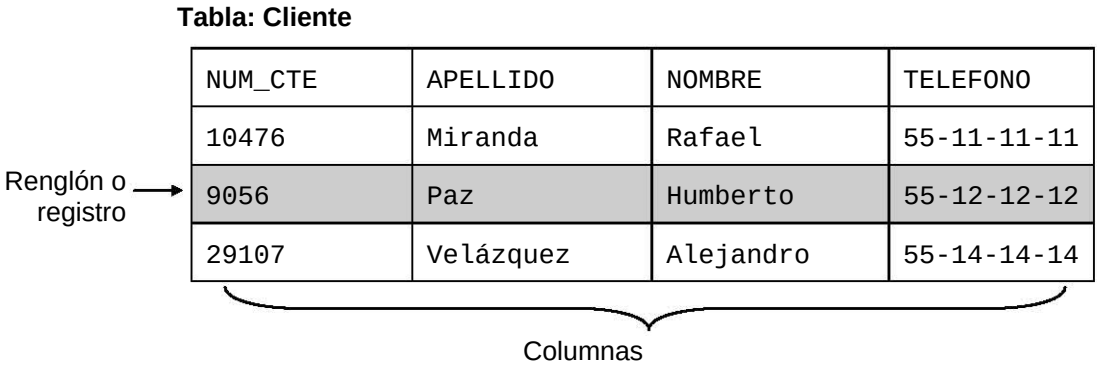


DIAGRAMA 1.2.1.1

Características de una base de datos relacional

Regresando al concepto de base de datos relacionales, en la definición de las tablas y la información que se almacena en las mismas deben observarse lineamientos generales que permiten que el acceso, recuperación y actualización de la información requiera el menor tiempo posible para un sistema de este tipo.

Dichos lineamientos son:

1. *Versatilidad para la representación de relaciones*
   El sistema de administración de datos debe proporcionar la posibilidad de recuperar información en diferentes vistas de agrupación o consulta.

2. *Redundancia mínima*
   La redundancia, es decir almacenar un dato que no es parte de la liga entre dos tablas en más de una tabla, es costosa al ocupar mayor espacio de almacenamiento que el necesario y se requieren múltiples operaciones de actualización cuando el dato cambia, por lo tanto la información no deberá duplicarse innecesariamente dentro de la estructura.

3. *Integridad*
   Además de proteger los datos contra posibles problemas del equipo físico, el sistema debe incluir procedimientos de validación que aseguren que los valores de los datos se ajusten a reglas de asociación definidas de antemano.

Características de una tabla en una base de datos relacional

La información que se almacena en una tabla debe cumplir con las siguientes restricciones básicas.

- Cada columna debe de ser única y no pueden existir columnas duplicadas.
- Cada columna debe de estar identificada por un nombre específico.
- Los valores de una columna deben pertenecer al *dominio* (conjunto de valores válidos) que representa, y es posible que un mismo dominio se utilice para definir los valores de varias columnas.

Una tabla que cumpla las condiciones anteriores tiene asociadas las siguientes propiedades:

- Las filas pueden estar en cualquier orden.

- A una fila se le hace referencia mediante todos los valores que la forman.
- Las columnas pueden estar en cualquier orden, pero se recomienda que las columnas llave empiecen por el lado izquierdo o sean las primeras en ser listadas.
- Se hace referencia a una columna mediante el nombre que la identifica.

SQL

SQL es el acrónimo de Structured Query Language. Es un lenguaje de consulta de base de datos basado en el idioma inglés desarrollado por IBM y estandarizado por ANSI.
Los lenguajes SQL de Oracle e Informix están basados en una implementación extendida del ANSI-standard SQL.

SQL no es muy diferente de un RDBMS a otro. Es relativamente simple convertir un esquema de un RDBMS a otro RDBMS, especialmente los tipos de datos simples. Desde luego se requiere un esfuerzo mayor para las tareas de administración como el respaldo, integridad, manejo de recursos, y la migración de aplicaciones; el esfuerzo se incrementa en función del uso de características o funcionalidades propietarias de los diferentes RDBMS, por ejemplo en los lenguajes de programación de procedimientos almacenados, tipos de datos especiales, etc.

DDL, DML

Un subconjunto de instrucciones de SQL agrupadas en el Data Definition Language (DDL), está integrado por instrucciones declarativas o descriptivas de los objetos de la base de datos tal y como los percibe el usuario. Las sentencias principales crean, modifican y destruyen elementos de la base de datos.

Un segundo subconjunto de instrucciones de SQL llamadas Data Manipulation Language (DML) son empleadas para actualizar, modificar o consultar información almacenada en las estructuras de la base de datos. Las operaciones básicas en esta categoría son: Insertar, Eliminar y Actualizar registros en la base de datos.

La sintaxis de las sentencias DML y DDL pueden variar entre diferentes motores de bases de datos, sin embargo, estos cambios no son sustanciales pues cumplen con las características generales de SQL estándar.

Ejemplo de sentencias DDL

Las diferencias entre ambos motores de datos se hacen presentes también en la sintaxis de las sentencias DDL para las operaciones básicas; a continuación se presenta un resumen de las mismas.

Sentencia CREATE TABLE en Oracle e Informix

Informix	Oracle
Syntax: CREATE TABLE example_table (datetime_column datetime not null, integer_column int null, text_column text null, varchar_column varchar(10) null)	**Syntax:** CREATE TABLE example_table (datetime_column date not null, integer_column number null, text_column clob null, varchar_column varchar2(10) null)

Sentencia ALTER TABLE en Oracle e Informix

Informix	Oracle

Syntax:
ALTER TABLE example_table
{
ADD (newCol_name newcol_type
[BEFORE oldCold_name]
}
 	
ALTER TABLE example_table
{
MODIFY (oldcol_name newcol_type
}
	Syntax:
	ALTER TABLE table_name modify
	(column1_name column1_datatype);

Ejemplo de sentencias DML

Sentencia DELETE en Oracle e Informix

Informix	Oracle
Syntax: DELETE {table | view|synonim} [WHERE {condition|{ESQL CURRENT OF cursor_name}}]	**Syntax**: DELETE [FROM] [user.]{table | view} [@dblink] [alias] [WHERE where_clause] **Descripcion**: FROM es opcional. ALIAS puede emplearse como nombre de correlación y puede emplearse en la condición. Aplican sobre una tabla a la vez.

Sentencia INSERT en Oracle e Informix

Informix	Oracle
Syntax: INSERT INTO {table | view}[(column [, column]...)]{VALUES (expression [, expression]...) | query...}|execute procedure;	**Syntax**: INSERT INTO [user.]{table | view}[@dblink][(column [, column]...)]{VALUES (expression [, expression]...) | query...};

Sentencia SELECT en Oracle e Informix

Informix	Oracle
Syntax: SELECT [ALL | DISTINCT | UNIQUE] select_list FROM table_name [table_alias] [WHERE {condition|{ESQL CURRENT OF cursor_name}} [WHERE condition] [GROUP BY column_list] [HAVING condition] [ORDER BY column_name [ASC] [DESC]] [INTO TEMP table_name]	Syntax: SELECT [DISTINCT | ALL] { [table_name], [table_alias]…} FROM [user.][table_name] | [view] [WHERE condition] [GROUP BY expr [,expr]… [HAVING condition]] [ORDER BY {column_name} [ASC] [DESC]]

Ejemplo de sentencias DML

Sentencia UPDATE en Oracle e Informix

Informix	Oracle
Syntax: UPDATE table | view}|synonim SET {clause} WHERE {condition} |ESQL |SPL CURRENT OF {cursor id}	**Syntax:** UPDATE [user.]{table | view} [@dblink] SET [[user.] {table. | view.}] { column = expression | NULL | (select_ statement) [, column = expression | NULL | (select_statement)...] | (column [, column]...) = (select_statement)} [WHERE {condition | CURRENT OF cursor}] **Descripción**: La particula INTO es necesaria. Un insert aplica a una tabla a la vez.

Sentencia DELETE en Oracle e Informix

Informix	Oracle
Syntax: DELETE FROM table WHERE {condition} |ESQL |SPL CURRENT OF {cursor id}	**Syntax :** DELETE [FROM] [user.][table | view } @dblink] [WHERE condition]

Modelo Entidad-Relación

Durante el proceso de construcción de una base de datos se pasa por dos etapas fundamentales: el diseño lógico y el diseño físico de la base de datos.

El propósito del diseño lógico de base de datos es entender y describir los datos. El propósito del diseño físico es maximizar el rendimiento.

El diseño lógico es independiente de la tecnología del DBMS; es posible tomar el diseño lógico e implementarlo en una base de datos Sybase o Informix u Oracle; el diseño físico es específico para un producto DBMS.

Una herramienta básica en el diseño lógico de base de datos es el Diagrama entidad/relación DER (ERD's por su nombre en inglés: entity/relationship diagrams), que explican las relaciones de los elementos de datos dentro de una organización. Este modelo fue presentado por Chen en 1976.

El Diagrama 1.2.2 muestra un ejemplo de DER para Clientes y Orden de compra, donde un cliente puede tener muchas ordenes, pero una orden solo está relacionada a un cliente.

DIAGRAMA ENTIDAD RELACION

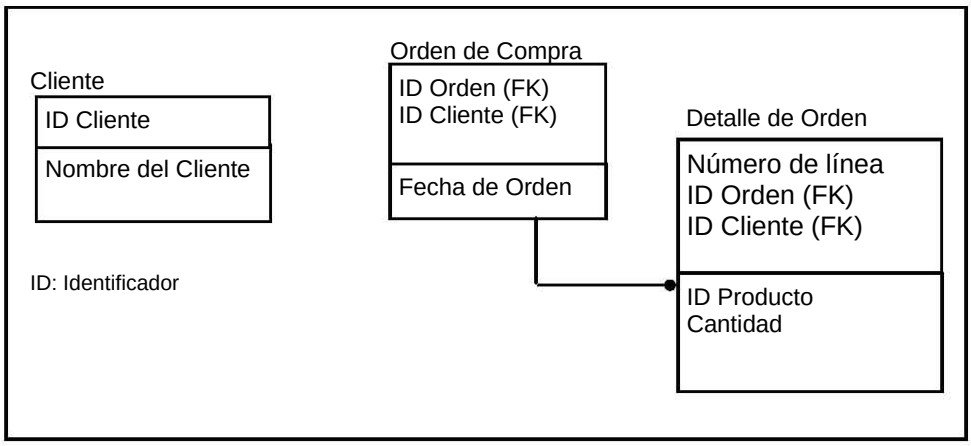


Diagrama 1.2.2

El diseño lógico de una base de datos es el proceso de entender los requerimientos de información de la empresa y la organización de los datos en *entidades*, *atributos* y *relaciones.*

El principal concepto del modelo ER (Entity/Relationship) es la *entidad*, que es un "elemento" en el mundo real con existencia independiente. Una entidad puede ser un objeto físico (una persona, un auto, una casa, un empleado) o un objeto conceptual (una compañía, un puesto de trabajo o un curso universitario).

Cada entidad tiene propiedades específicas, llamadas *atributos*, que la describen. Por ejemplo, una sala de clases tiene un nombre (14A, 120U), una ubicación, un cupo máximo, etc. En nuestro ejemplo, la entidad "alumno" posee los atributos nombre y matrícula. Una entidad particular tiene un valor para cada uno de sus atributos.

Cada uno de los atributos de una entidad posee un *dominio*, el que corresponde al tipo del atributo. Por ejemplo, "matrícula" tiene como dominio al conjunto de los enteros positivos y "nombre" tiene como dominio al conjunto de caracteres.

Para todo conjunto de valores de una entidad, se necesita un atributo o combinación de atributos, que identifique a cada entidad en forma única. Este atributo o combinación de atributos se denomina *llave primaria*. Por ejemplo, el número de matricula es una buena llave para la entidad alumno, no así el nombre, porque pueden existir dos personas con el mismo nombre.

Una *relación* se puede definir como una asociación entre entidades. Por ejemplo, la entidad "libro" puede estar relacionada con la entidad "persona" por medio de la relación "está pedido". La entidad "alumno" puede estar relacionada con la entidad "curso" por la relación "está inscrito". Una relación también puede tener atributos. Por ejemplo, la relación "está inscrito" puede tener los atributos "semestre" y "nota de aprobación".

Un diseño lógico no tiene el mismo propósito que un diseño físico, el propósito principal de un diseño físico es el performance (rendimiento) de la implementación.

Ejemplo de DER

Suponga que estamos modelando los datos de una COMPAÑIA. La base de datos COMPAÑIA debe mantener información sobre los empleados de la compañía, los departamentos y los proyectos.

La descripción de un segmento del mundo (la parte de la compañía a ser representada en la base de datos) es la siguiente:

1. La compañía está organizada en departamentos. Cada departamento tiene un nombre único, un número único, y un empleado particular quien lo administra. Se quiere saber la fecha en que el empleado administrador empezó a hacerse cargo del departamento.
   Un departamento puede tener varios locales.

2. Cada departamento controla un cierto número de proyectos. Cada proyecto tiene un nombre y número únicos y un local.

3. Para cada empleado se desea tener su nombre, ruta, dirección, salario, sexo y año de nacimiento. Un empleado es asignado a un departamento,

pero puede trabajar en varios proyectos, los que no son necesariamente controlados por el mismo departamento.

Se quiere saber: a) el número de horas semanales que un empleado trabaja en cada proyecto y b) el nombre del supervisor directo de cada empleado.

4. Se desea conocer las personas dependientes de cada empleado para propósitos de seguros. De cada dependiente se desea conocer el nombre, sexo, fecha de nacimiento y relación con el empleado.

El Diagrama 1.2.2 muestra el diagrama de Chen de esta base de datos.

DIAGRAMA DE CHEN

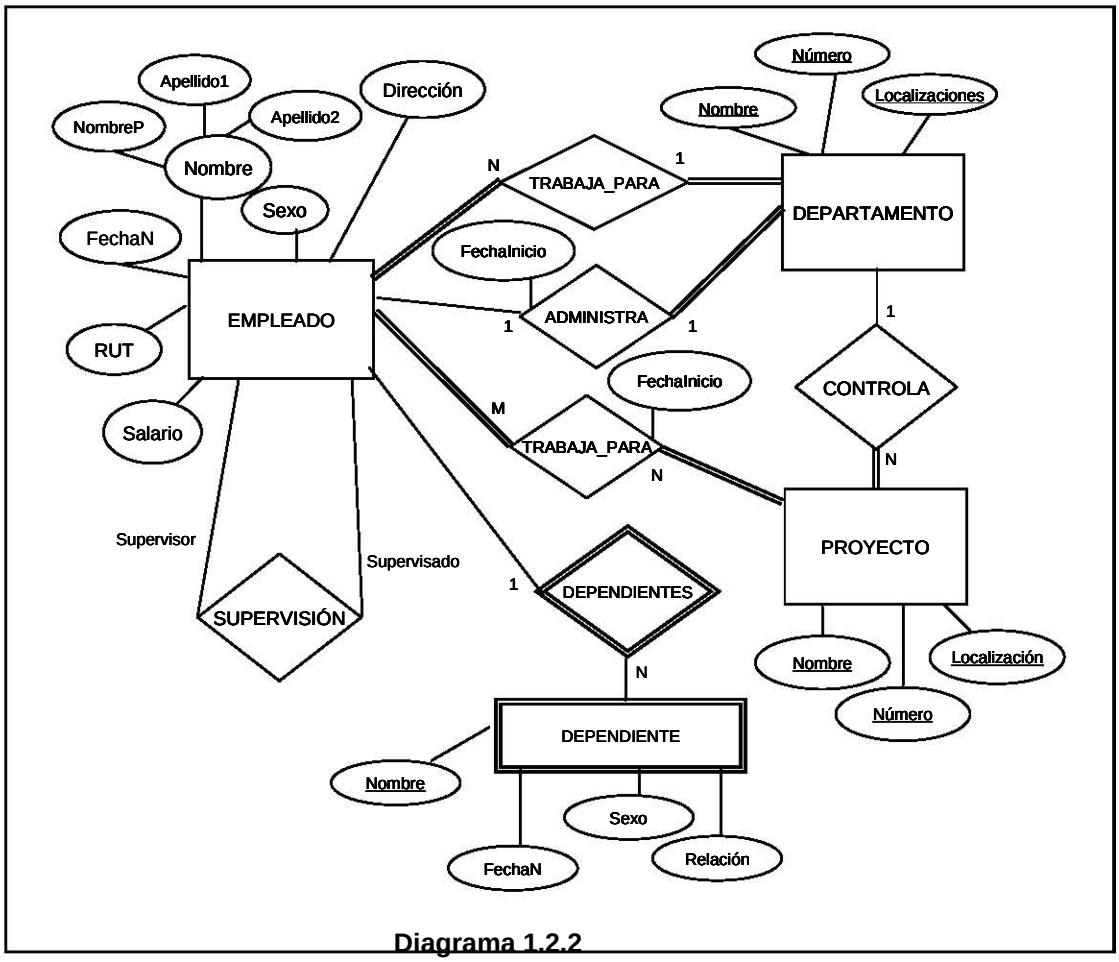


Diagrama 1.2.2

En este diagrama los rectángulos representan conjuntos de entidades, las elipses representan atributos y los rombos representan conjuntos de relaciones.

Podemos también considerar el caso de los usuarios que necesitan registrar las ventas por almacén para una distribuidora de libros. (Diagrama 1.2.3).

DIAGRAMA ENTIDAD RELACION

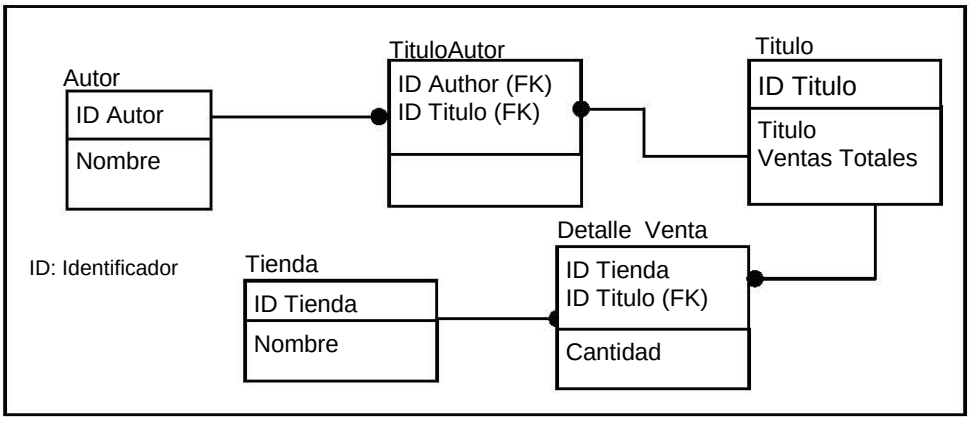


Diagrama 1.2.3

En el *modelo relacional expandido* que es común actualmente, la base de datos incluye además de estructuras de almacenamiento de información, bloques de código compilado o almacenado, clasificados en: procedimientos, funciones, vistas, triggers.

- **Disparadores (triggers):** son bloques de código SQL almacenado, que se ejecuta en forma automática ante eventos específicos que ocurren en la base de datos. Los más comunes son los que se utilizan al insertar, modificar o eliminar registros de una tabla. Es posible asociar triggers a eventos de la base de datos en su conjunto, por ejemplo, al iniciar una instancia de la base de datos. Desde un trigger es posible ejecutar o invocar cualquier función, procedimiento, etc., contenido en la base de datos. En Informix es posible ejecutar una sentencia *commit* pero no así en Oracle.

- **Funciones (functions):** son bloques de código almacenado, permiten el paso de parámetros de entrada, pero siempre devuelven un solo valor. Se puede llamar funciones desde otras unidades de programas y desde herramientas de desarrollo de aplicaciones. Una función puede ser invocada desde una sentencia SQL como si fuera una columna más en una tabla.

- **Índices:** Los índices son estructuras de datos cuyo uso está orientado a aumentar la velocidad de acceso a la información almacenada en las tablas. Una tabla puede tener más de un índice.

- **Procedimiento almacenado (stored procedure):** código que ejecuta determinada acción. Permite el paso de parámetros de entrada, salida y entrada/salida. Se pueden llamar procedimientos desde otras unidades de programas, o directamente desde herramientas de desarrollo de aplicaciones como Deplhi, Forms, Reports, Visual Basic, etc.

- **Secuencias:** Las secuencias son un tipo especial de objeto de Oracle, es un numerador que asegura una serie de números únicos. Se pueden definir como crecientes o decrecientes y el intervalo de incremento es variable. Con cada nuevo número generado, no es posible generar el mismo valor posteriormente.

- **Vistas:** Las vistas son sentencias SQL almacenadas en la base de datos. Las ventajas de la utilización de una vista residen en que, una vez creada, se puede utilizar y manipular casi como una tabla. Pueden hacer referencia a más de una tabla. Permiten simplificar el acceso a la información almacenada en varias tablas o restringir las características de los registros que muestra, permitiendo implementar un nivel específico de seguridad al ocultar la estructura de las tablas relacionadas.

- **Constraint:** es una regla de integridad, se define en la creación de la tabla a la que está asociado y todos los valores asignados a la columna de la tabla deben cumplir con la restricción especificada por el constraint.
  La definición, es decir, la regla que define el *constraint* para una columna se verifica cada vez que se inserta un registro (Insert) o el valor de la columna cambia (Update); si la definición del constraint no se cumple, la operación de Insert o de Update falla.
  Cuando se define un constraint, se especifica si el valor NULL no se permite como entrada para la columna asociada.

RDBMS en el mercado

Informix es un sistema de administración de base de datos que durante las décadas de los 80's y 90's se convirtió en uno de los estándares más importantes de la industria de cómputo a nivel mundial.

Durante esos años ha sido importante la competencia por el mercado con Oracle, otro RDBMS con amplia presencia en organizaciones públicas, privadas, de banca, industria, distribución, etc.

En ambos casos se trata de conjuntos de programas o módulos de programas utilizados en la administración de datos mediante los cuales se reduce el tiempo necesario para organizar, almacenar y recuperar información en una amplia gama de posibilidades que van desde las consultas directas mediante una interfase de SQL, aplicaciones para generación de reportes, herramientas de extracción y carga de altos volúmenes de información, aplicaciones para formatear información en formatos correspondientes a otros sistemas, etc.

Ambos sistemas han evolucionado con el tiempo proporcionando cada vez entornos de desarrollo de aplicaciones más rápidos; las interfases gráficas y el acervo de materiales como programas tutoriales, cursos, etc., en Internet han hecho estos entornos de manejo de información más robustos y con menores costos de capacitación.

Sin embargo, la tendencia en algunas organizaciones que están en proceso de normalización y actualización de su plataforma de cómputo han definido Oracle como la herramienta estándar para el mantenimiento de bases de datos.

En cuanto a herramientas de desarrollo de aplicaciones como Delphi, se trata de una herramienta gráfica de amplia demanda por la rapidez con que un prototipo puede evolucionar a una aplicación definitiva; en general es posible acceder a una amplia gama de sistemas RDBMS y algunos sistemas manejadores de archivos, siempre y cuando se cuente con las versiones de aplicaciones de conexión con la base de datos compatibles con la versión de Delphi con la que se está trabajando.

Es común que para las versiones más recientes de la herramienta de desarrollo no se tenga una funcionalidad completa con versiones muy antiguas del RDBMS y viceversa.
En estos casos, se tendrá acceso a funcionalidades generales, pero probablemente con funcionalidades muy específicas no se obtengan los resultados que se requieren sin realizar alguna modificación en la forma en que se opera la aplicación final o el acceso a la base de datos.

1.3 Red Local (LAN)

Una red de área local LAN (Local Area Network) es un sistema en el que se comparten recursos e información entre diferentes equipos de cómputo y usuarios, con la mayor eficiencia posible y manteniendo a la vez privacidad y seguridad en los intercambios de datos.

Una red local se forma con los siguientes elementos.

- **Recursos** aislados e independientes llamados *nodos*, que usualmente son computadoras pero pueden también ser impresoras, ruteadores o unidades especiales de almacenamiento controladas por un procesador dedicado.

- Uno o varios **medios** de comunicación o enlace para la **transmisión de datos** entre nodos, junto con el equipo necesario para administrar la interconexión, dichos medios pueden ser cableado, comunicación satelital, microondas.

- **Software** especializado para **intercomunicar** y controlar los diversos nodos bajo una configuración particular.

Las Redes de Área Local contribuyen a mejorar la productividad de la organización al facilitar el uso de recursos de manera compartida.

Una Red de Área Local está limitada geográficamente a una área pequeña, una oficina, un piso en un edificio o un edificio completo; la transmisión de datos es considerada de alta velocidad, en un rango de 10 MBPS (Mega Bits por segundo).

1.3.1 Recursos

Las características de los elementos tradicionales de una red son las siguientes.

- **Nodo:** Generalmente son equipos personales, terminales, estaciones de trabajo, impresoras, etc.

- **Servidor (Server):** El Servidor es el corazón de la Red de Área Local (LAN). Esta computadora, generalmente de alta velocidad, aloja el sistema operativo y gestiona el flujo de datos a través de la red. Las estaciones de trabajo individuales y los dispositivos periféricos compartidos están conectados al Servidor.

  El servidor puede ser de dos tipos: dedicado o no dedicado.

Servidor dedicado: Es una computadora cuya única función es responder a los requerimientos de los demás equipos que estén conectados a la red.

Servidor no-dedicado: Es un equipo de cómputo que realiza las funciones de una computadora personal, Terminal o estación de trabajo y además provee los servicios que comúnmente se requieren a un servidor.

Las redes pueden incluir tres tipos generales de Servidor: de archivo, de impresión, y de comunicación. A veces, los servidores de archivos, de impresión y de comunicaciones residen en una computadora y otras veces las tareas están distribuidas entre varias computadoras.

- ***Estación de Trabajo (Workstation*)**: Una estación de trabajo es por lo general una computadora con sistema operativo propio. Está conectada al servidor y a otros nodos a través de un medio de comunicación (Diagrama 1.3.1).

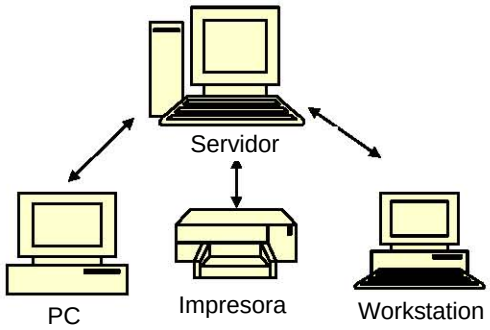


DIAGRAMA 1.3.1

1.3.2 Medios de transmisión

Los esquemas de conexión entre nodos son variables, cuando los equipos están cerca basta con un par de módems y un cable para la interconexión, empleando un sistema conocido como punto a punto que, por las limitaciones de velocidad de los módems únicamente resulta viable cuando la cantidad de datos a transmitir es pequeña.

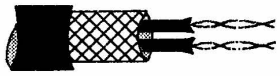
Actualmente se cuenta con fibra óptica, enlaces de microondas, de satélite, etc. como medios de transmisión de información a alta velocidad.

A continuación se describen brevemente algunos de los medios de transmisión más comunes.

El *cable de par trenzado sin blindar* es barato, fácil de instalar y puede utilizarse en redes reducidas. Se recomienda cable de cuatro pares de hilos, la distancia máxima que podría soportar es de 120 metros de longitud, se recomiendan 45 metros.

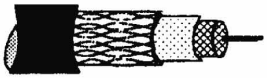
CABLE TIPO-3

El *cable de par trenzado blindado* es similar al cable de par trenzado sin blindaje, excepto que son más gruesos y están protegidos de las interferencias por una capa aislante protectora. El tipo más comúnmente empleado de redes de Área Local es el cable "IBM tipo-1".
El blindaje y el número de vueltas por pulgada convierten al cable de par trenzado blindado en una alternativa fiable. La longitud máxima recomendable tanto para cable tipo-1 como para el cable tipo-2 es de 200 metros, se sugieren 100 metros.



CABLE TIPO-1

El *cable coaxial* consta de dos conductores rodeados por dos capas aislantes. La primera capa aislante encierra un hilo lateral de cobre conductor. Esta primera capa tiene un blindaje trenzado que la cubre.
La distancia máxima recomendada para el uso del cable coaxial es de 300 metros.



CABLE COAXIAL

El *cable de fibra optica* transmite los datos como impulsos de luz a través de cables de vidrio. Actualmente los grandes sistemas de redes están soportados por cables de fibra óptica.
El cable de fibra óptica tiene importantes ventajas sobre los cables de cobre.
Al no ser sensibles a las interferencias electromagnéticas no pierden información por esta causa, este tipo de cable es delgado y flexible.

CABLE DE FIBRA OPTICA

- ***Tarjetas de Red (NIC's-Network Interfaces Cards*)**: Las tarjetas de red se instalan en la computadora o estaciones de trabajo para poder conectarse a la red, existe una amplia gama de tarjetas en el mercado para este fin.

- ***Repetidores***: La función que desarrolla un repetidor es de transmitir y volver a amplificar la señal que es transmitida a lo largo de la red. Es importante recordar que los Repetidores trabajan para repetir la señal, no es posible conectar redes utilizando medios diferentes de transmisión.

- ***Bridge***: Es una combinación de hardware y software; el componente de software determina el segmento de red al que debe dirigirse un paquete de información, dependiendo de la dirección destino del mismo. Los *bridges* locales pueden ser tanto internos como externos. Ambos funcionan de la misma forma, pero las diferencias de sus rendimientos pueden ser considerables. Los bridges externos casi siempre tienen un rendimiento mayor; sin embargo, son más costosos de implantar. Los bridges internos residen dentro del Servidor simplemente en una tarjeta adicional de interfaz de red (Diagrama 1.3.2).

BRIDGES EXTERNOS

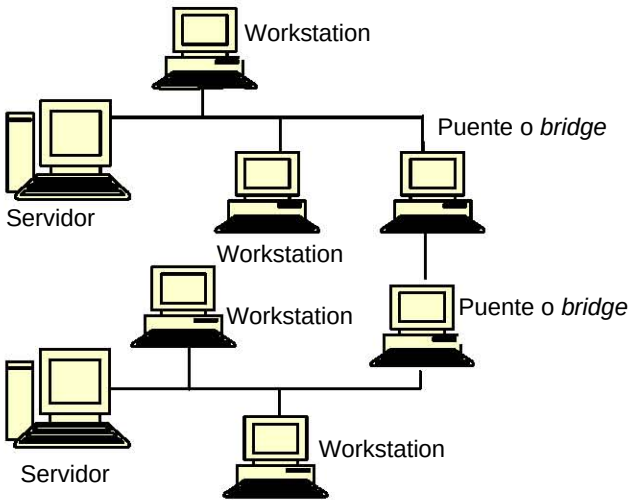


DIAGRAMA 1.3.2

- ***Gateways*:** Los Gateways son dispositivos de comunicación que permiten conectar sistemas no similares. Permiten conectar redes con mainframes o mini computadoras. Como los bridges, los gateways pueden ser locales o remotos, dependiendo si la distancia física impone o no una forma de transmisión intermedia (Diagrama 1.3.3).

CONEXIÓN POR MEDIO DE UN GATEWAY

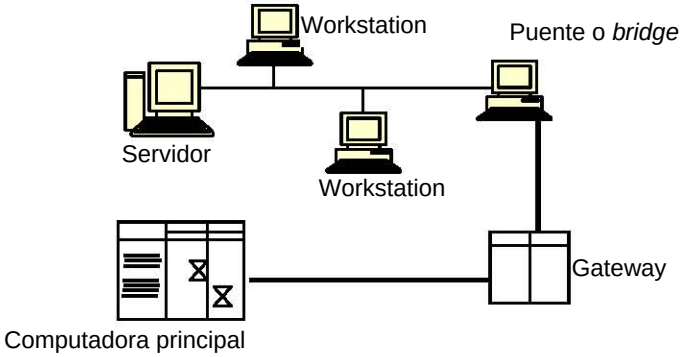


DIAGRAMA 1.3.3

1.2.3 Topología de las Redes:

Topología es el término utilizado para describir la forma en que están conectadas las estaciones de trabajo, terminales, y servidores que componen la red. La topología influye directamente en el tipo de medio con que se conecta la red y con gran parte del costo de la red LAN; existen varios tipos, entre ellos:

a) punto a punto
b) estrella
c) bus lineal
e) anillo

Las redes pueden estar formadas por un solo tipo de topología, pero también pueden conectarse redes híbridas.

a) ***Redes punto a punto:*** Es la topología más simple, consiste en ligar dos computadoras por medio de sus puertos seriales (Diagrama 1.3.4).

TOPOLOGÍA PUNTO A PUNTO

DIAGRAMA 1.3.4

Topología en estrella

Desventajas	Ventajas
• Requiere el uso exclusivo del puerto serial, limitando el uso del módem y el mouse.	• Es un esquema seguro al involucrar solo dos computadoras.
• La comunicación por el puerto serial es lenta.	• Su costo es mínimo.

b) ***Redes en estrella:*** En la topología de Estrella se tiene un procesador central y terminales distantes conectadas a él, se puede decir que todas las terminales están conectadas al Servidor pero no están conectadas entre ellas. Esto provoca que todo tráfico de datos tenga un control centralizado para toda la red (Diagrama 1.3.5).

TOPOLOGÍA EN ESTRELLA

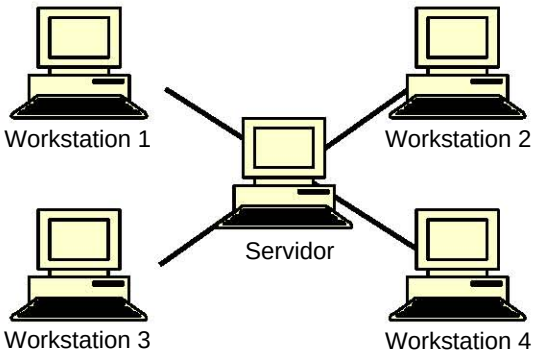


DIAGRAMA 1.3.5

Topología en estrella

Desventajas	Ventajas
• La transmisión de datos es lenta.	• El fallo de una terminal que falla no afecta el funcionamiento del resto de la red.
• La expansión de la red es limitada, depende de la capacidad de la computadora central.	• El diagnóstico de fallas es centralizado y por lo tanto inmediato.
• Cuando el servidor falla, toda la red sufre las consecuencias.	• El control de acceso a los periféricos es directo al estar conectados directamente al procesador central.

c) ***Red de bus lineal*:** Es la topología más común; consiste en un bus lineal sencillo de comunicación con terminadores en cada extremo (por ejemplo, cable coaxial), cada nodo se conecta al bus mediante un terminador (Diagrama 1.3.6.)

TOPOLOGÍA DE BUS LINEAL

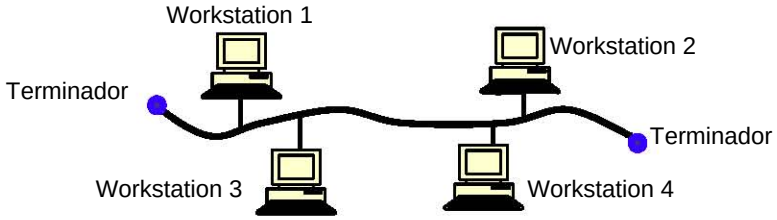


DIAGRAMA 1.3.6

Ventajas	Desventajas
• Es posible agregar nodos de manera sencilla.	• Es difícil aislar las fallas.
• Cuando un nodo falla no afecta al resto de la red.	• Los nodos son generalmente computadoras con sistema operativo propio.

d) ***Red de anillo*****:** En una red de anillo el cableado va de estación en estación (y al Servidor) sin que haya un principio y un final como se muestra en el Diagrama 1.3.7, generalmente las redes de anillo como las de bus pueden extenderse a grandes distancias.

La información fluye en un solo sentido y de manera ordenada. Resulta económico cuando las estaciones se encuentran en un área geográfica limitada, aunque no representa colisiones, la velocidad de transmisión depende del número de nodos que contenga la red. Si alguno de los nodos de la red tiene alguna falla, la red deja de funcionar (Diagrama 1.3.7).

Ventajas	Desventajas
• Cualquier paquete que se transmita podrá ser visto por todos los nodos de la red.	• Los datos fluyen en una sola dirección alrededor del anillo.
• Presenta menos colisiones que los sistemas de bus.	• Un paquete con errores puede circular indefinidamente de la red si no hay un monitoreo adecuado de la red.
	• Si falla un nodo, la comunicación en la red se interrumpe.

TOPOLOGÍA DE ANILLO

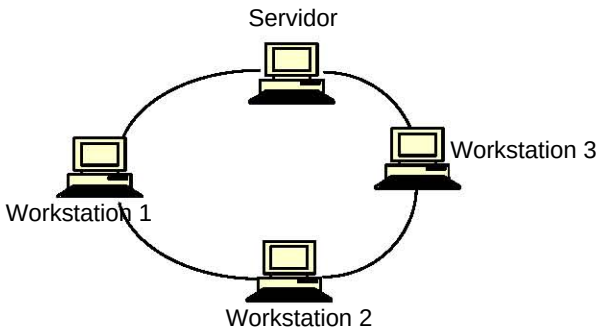


DIAGRAMA 1.3.7

1.2.4 Sistemas de Redes

El proceso de comunicación de datos a través de la red está integrado por seis componentes: a) la computadora origen, b) el protocolo, c) el transmisor, d) el medio físico, e) el receptor y f) la computadora destino.

La computadora origen puede ser una estación de trabajo, un servidor, un gateway o cualquier dispositivo de la red. El protocolo consta de un conjunto de circuitos integrados y software que controla la tarjeta de interfaz de red.
El protocolo es el responsable de la lógica de la comunicación en la red; el transmisor inicia la señal electrónica a través de la topología física. El receptor reconoce y captura la señal traduciéndola por medio del protocolo, como se muestra en el Diagrama 1.3.8.

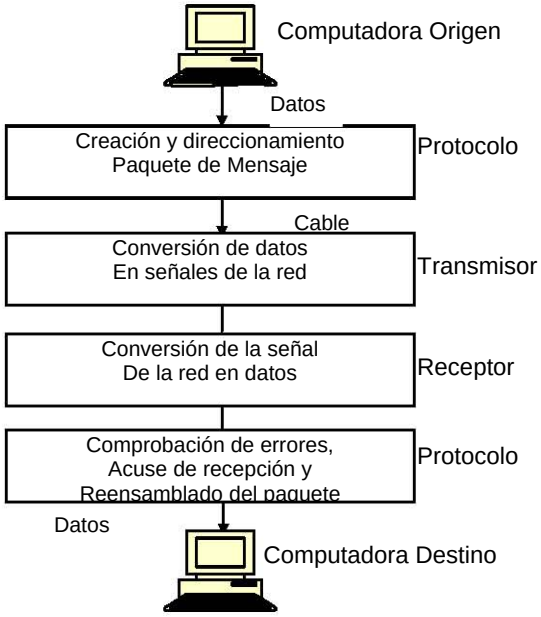


DIAGRAMA 1.3.8

1.2.4 Control para la transferencia de datos

A continuación se hace mención sobre los protocolos, los cuales se encargan de controlar el enlace y la transferencia de datos. Las funciones de protocolos son:

- Control de transferencia de datos
- Chequeo y recuperación de errores.
- Codificación de la información.
- Transparencia de la información
- Utilización de la línea.
- Sincronización
- Transparencia para facilitar las comunicaciones.
- Bootstrapping. Rutina de inicialización de la red.

1.2.5 Modelo OSI

El modelo OSI (Interconexión de Sistemas Abiertos en sus siglas en inglés), intenta plasmar un estándar definido por los principales organismos de normalización, administraciones de telecomunicación y empresas.
La arquitectura del Modelo OSI se muestra en el Diagrama 1.3.9.

Los objetivos del estándar del modelo OSI son:

- Proporcionar las normas para la interconexión de sistemas.
- Definir los puntos de interconexión para el intercambio de información entre los sistemas.

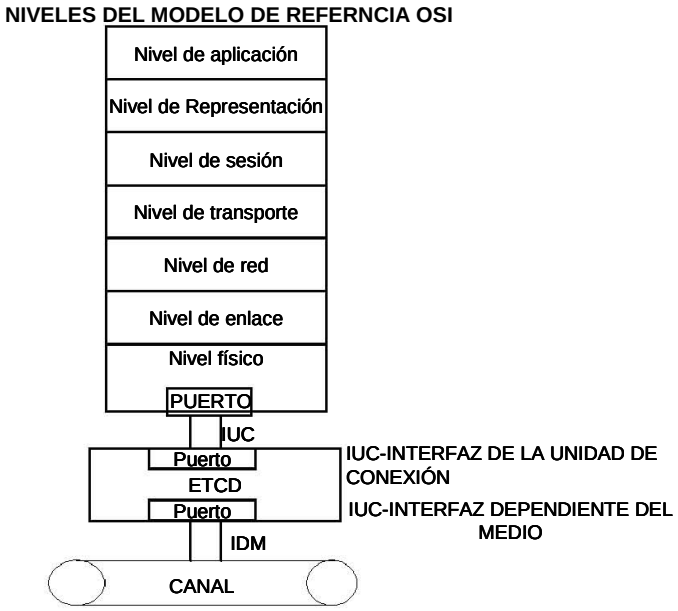


Diagrama 1.3.9

Niveles del Modelo OSI

Nivel Físico Es el nivel más bajo del modelo, se encargara de activar, mantener y desactivar la Estación de Trabajo conectada al medio de transmisión (canal). Maneja la parte física de la conexión.

Nivel de Enlace Su objetivo es la transferencia correcta de los datos por el canal. Proporciona la sincronización a los datos, para delimitar el flujo de bits por el canal, garantiza la identidad de los mismos para que lleguen correctamente al destino. Detecta los errores en la transmisión y recupera, si es posible la información.

Niveles del Modelo OSI . . .

Nivel de Red	Define una interfase entre el usuario y la red de comunicación de paquetes. Establece el encaminamiento ("Ruteo") de los mensajes por la red, o su comunicación con distintas redes.
Nivel de Transporte	Es la interfase entre los niveles superiores y la red de comunicación de datos. Permite elegir al usuario entre los diversos recursos de la red.
Nivel de Sesión	Mecanismo de intercambio de datos entre los usuarios. Posee una serie de servicios que el usuario puede solicitar, por ejemplo: ▪ Solicitud de datos/servicios ▪ Aceptación de mensajes. Cada usuario para acceder a la red debe estar dado de "alta" en la misma por medio de una sesión.
Nivel de Presentación	Asigna una sintaxis a los datos, de acuerdo al tipo, por ejemplo enteros, fraccionarios, caracteres, etc. Recibe los datos de nivel de aplicación y checa la sintaxis, sin importar el significado. Posee tablas sintácticas para efectuar esos chequeos.
Nível de Aplicación	Atiende al usuario final. Este nivel toma en cuenta la semántica de datos. Contiene varios elementos de servicio capaces de gestionar procesos de aplicación tales como gestión de trabajos, sentencias send/receive (enviar/recibir) de distintos lenguajes de programación; y el intercambio de datos comerciales. Así mismo, este nivel maneja los conceptos de Terminal virtual y archivo virtual.

Los protocolos más utilizados en las redes LAN son: Bus o Ethernet (norma 802.3 IEEE) Token Ring (IEEE 802.5) Token Passing (IEEE 802.5).

- Bus o Ethernet (Norma IEEE 802.3). - Este estándar se basa en los primeros tres niveles del modelo OSI. La liga de datos se desarrolla en dos subniveles: Control de logs lógicas (LLC- Logical Link Control) y control medio de acceso (MAC-Media Access Control). Donde el subnivel MAC es concerniente a la detección de colisiones de datos.

El Diagrama 1.3.10 se muestran los módulos que participan en una transmisión y recepción de información.

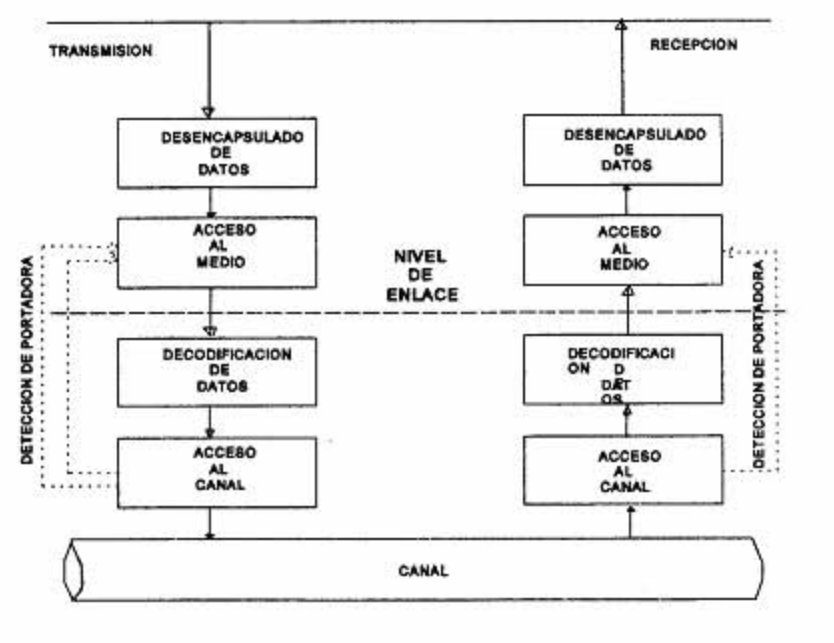


Diagrama 1.3.10

Las funciones de cada módulo son:

- Encapsulado y desencapsulado de datos. Establece el formato del mensaje, proporcionando las direcciones fuente y destino y checa el mensaje contra errores; este proceso es transparente.

- Acceso al Medio
  - Transmite el mensaje al nivel físico y lo recupera de este.
  - Almacena el mensaje en un buffer intermedio.
  - Intenta evitar colisiones, las maneja si existe.

➢ Token Ring (Norma IEEE 802.5). - El standard 802.5 fue desarrollado para abarcar redes de área local con topología de anillo, éstas manejan una señal (token) para el paso de la información de una Estación de Trabajo a otra.

1.4 ARQUITECTURA CLIENTE/SERVIDOR

En la arquitectura *cliente/servidor* la base de datos y la aplicación están divididas en dos partes: un *front-end* o *cliente* y un *back-end* o *servidor (server).* Ver el Diagrama 1.4.1

El *cliente* ejecuta una aplicación que accesa la información de la base de datos e interactúa con el Usuario por medio de un teclado, pantalla o un dispositivo señalizador como un mouse.

El *Server* ejecuta el software de base de datos y maneja las funciones requeridas para concurrencia y control de acceso a datos compartidos en la base de datos.

En la configuración más común se tiene un RDBMS como servidor y computadoras que ejecutan una aplicación gráfica como front-end.

Aún cuando la aplicación cliente y el RDBMS pueden ejecutarse en la misma computadora, es más eficiente y efectivo separar en diferentes computadoras al cliente y al server comunicándolos a través de una red.

La ventaja de este modelo, respecto del modelo mainframe, es su efectividad en costo y escalamiento. En lugar de requerir un costoso equipo central con gran capacidad de procesamiento, la arquitectura cliente/servidor utiliza varias máquinas de menor capacidad para llevar a cabo las mismas tareas.

La arquitectura cliente/servidor es fácil de escalar ya que es sencillo agregar máquinas cliente conforme la aplicación lo requiera; desde luego que el hecho de ir agregando más clientes para escalar la aplicación puede ocasionar que sobrepasemos las capacidades del servidor.

CONFIGURACION CLIENTE SERVIDOR

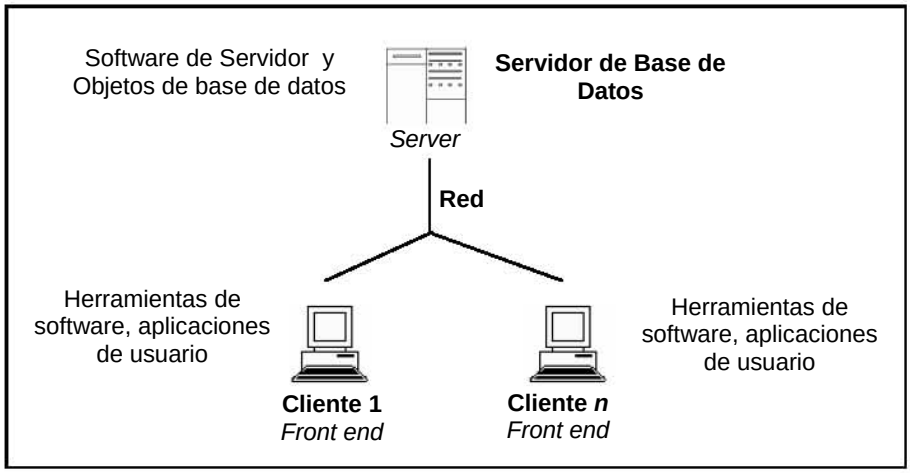


Diagrama 1.4.1

Las características de este modelo son las siguientes:

- Las aplicaciones cliente no realizan procesamiento de información, limitándose a la entrada de solicitudes de los usuarios, requiriendo datos al servidor, y analizando y presentando los datos empleando las capacidades de presentación del cliente (por ejemplo, gráficas y hojas de cálculo).
- Las aplicaciones cliente se diseñan sin dependencia de la localización física de los datos. Si el dato se distribuye o mueve a otros servidores de base de datos, la aplicación continúa funcionando sin modificaciones.
- La carga de procesamiento y almacenamiento de información se divide en dos máquinas.
- El servidor es responsable de optimizar y ejecutar las consultas o solicitudes enviadas por los clientes; además de extraer la información envía los resultados a través de la red al cliente. Una vez que el cliente recibe la información la procesa y la presenta al usuario.
- Brinda soporte para la operación de sistemas multiusuario.
- El costo de los equipos es menor que el de los Mainframe.
- Se pueden escalar fácilmente.
- El Costo Total de Propiedad es elevado y la administración es compleja.

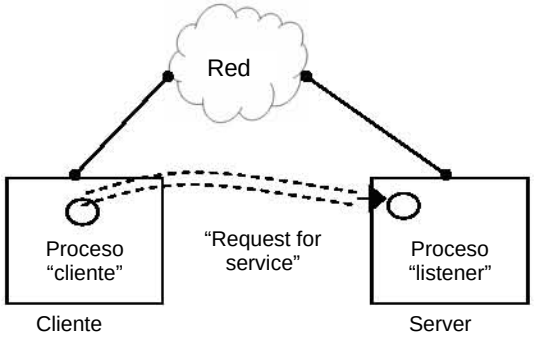
En el modelo Cliente/Servidor, un cliente (o solicitante de servicio) envía un request (solicitud) al servidor (o proveedor del servicio) que se ubica en algún lugar de la red. Un listener (programa de escucha) para ese tipo particular de servicio intercepta y evalúa la solicitud.

Las solicitudes válidas se pasan a un servidor con capacidad abierta (en posibilidad de proporcionar el servicio). El servidor realiza (ejecuta o resuelve) el servicio solicitado; se comunica con el cliente cuando es necesario, y envía los datos o resultados de regreso al cliente.
Todas estas tareas son invisibles para el usuario de las aplicaciones y de la base de datos; son resueltas por el RDBMS y el sistema operativo de los equipos.

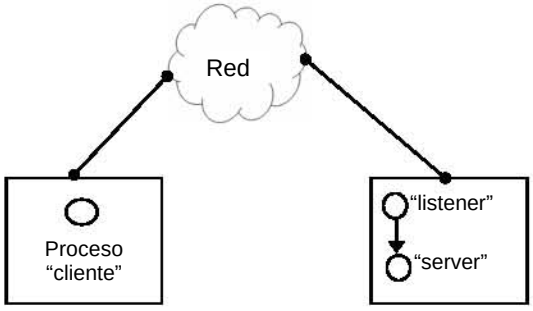
Este proceso se ilustra en el Diagrama 1.4.2.

Configuración cliente/servidor

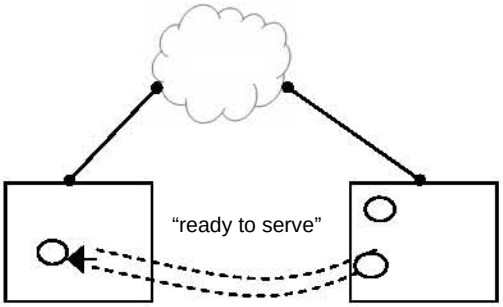
1 El *cliente* envía un "*request for service"* al *servidor*



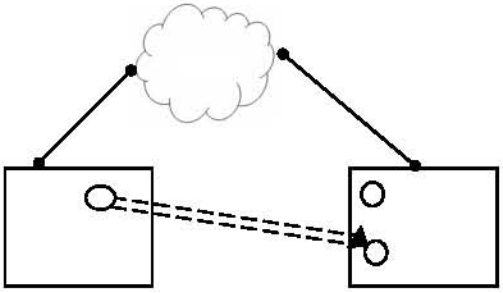
2 El "*listener"* en el servidor valida la solicitud, crea o asigna procesos del *"Server"* y continúa "*escuchando*"



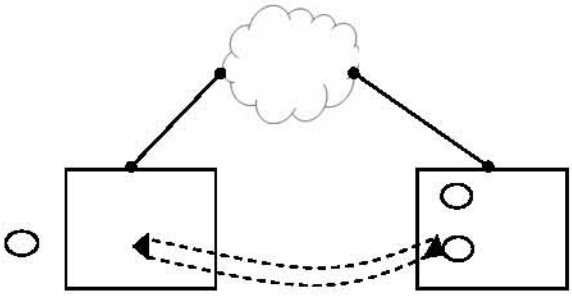
3 El proceso "*Server"* envía al *cliente* el mensaje "ready to serve"



4 El "*cliente"* envía una solicitud al *"server"*



5 El *"cliente"* y el *"Server"* envían y reciben solicitud y respuesta



6 Cuando alguno de los lados termina el proceso *Server* muere o regresa al pool

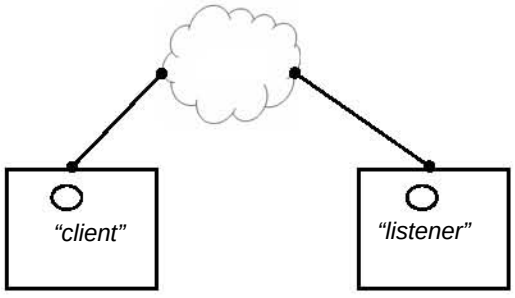


Diagrama 1.4.2

1.5 Metodología

Una metodología es un conjunto de principios y métodos por medio de la cual se conduce una práctica.

Estos principios y prácticas proporcionan un marco y disciplina para completar exitosamente un proyecto aplicando la metodología seleccionada.

Podemos decir que la utilidad de una metodología crece cuando durante la aplicación de la misma ésta autocorrige los problemas que se van encontrando en el desarrollo.

Resúmen

Hasta este momento los conceptos revisados son necesarios como antecedentes para la realización del presente trabajo, que a su ver son:

- ✓ La definición de un sistema y sus características.
- ✓ La definición de una base de datos y sus características como herramienta de administración y mantenimiento de información.
- ✓ La definición de red local y sus características dado que es el medio que nos permite compartir los recursos de cómputo, de almacenamiento de información y de intercambio de datos.
- ✓ La descripción de la arquitectura *cliente/servidor* dado que es la configuración del conjunto de aplicaciones que integran el sistema a migrar.
- ✓ La definición de *metodología*, dado que se sigue una guía de tareas orientada a la consecución del objetivo de la manera más rápida y eficiente.

CAPITULO II

ANALISIS DEL PROBLEMA

2.1 Cobertura del sistema

Las aplicaciones que forman parte del Sistema de Administración del Abasto de una de las más importantes instituciones de salud pública del país son el soporte para la administración, planeación y seguimiento a los procesos de abastecimiento de materiales médicos a las Unidades Médicas o Clínicas que dan servicio a los derechohabientes del Instituto en todo el país.

En el Diagrama 2.1.1 se muestra la forma en que está organizada el área de Sistemas de Abastecimiento de la Institución. La base de datos "*almacenes"* soporta la operación de aplicaciones a nivel de oficinas Delegacionales.

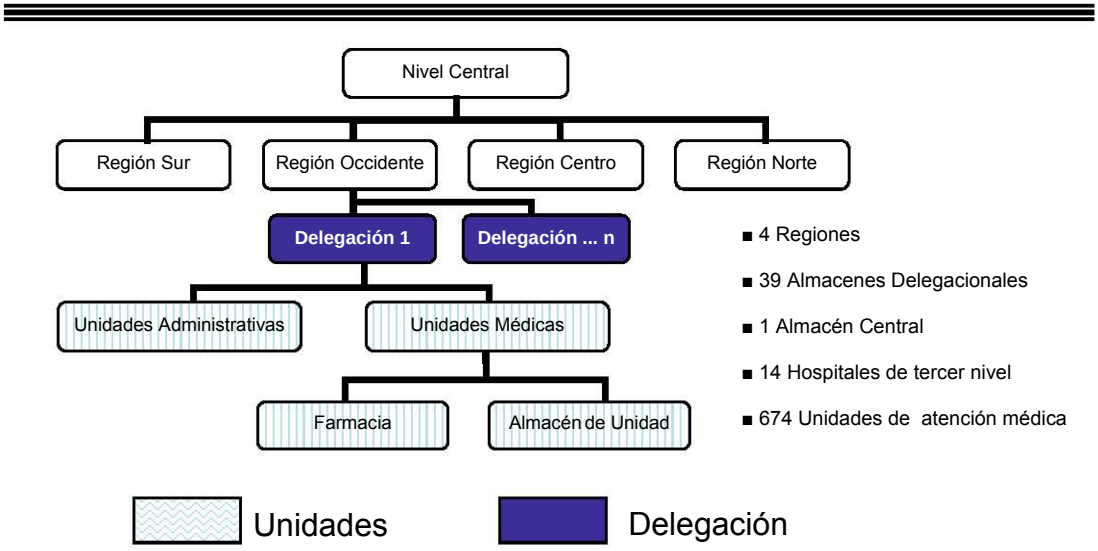


Diagrama 2.1.1

En el Nivel Normativo se definen las características de los productos que forman parte de los catálogos institucionales, por ejemplo, los medicamentos autorizados, los equivalentes, la presentación de los mismos, periodos de caducidad, Proveedores, Unidades Médicas, etc.

Geográficamente existen cuatro regiones, y para cada región se tienen de 1 a *n* Delegaciones; a su vez, para cada Delegación existen m Unidades Médicas y algunas Unidades Administrativas.

Las Unidades Médicas son las Clínicas a las que acuden los Derechohabientes del Instituto, éstas cuentan generalmente con una Farmacia y en algunos casos con un pequeño Almacén de Unidad Médica. La Farmacia y el Almacén de Unidad Médica guardan estrecha relación dado que éste último mantiene los niveles de surtido de materiales médicos requeridos por la Farmacia.

2.2 Características del sistema actual

El Sistema de Administración del Abasto es un sistema cliente/servidor, desarrollado en Delphi 5.0 y la información está almacenada en una base de datos de Informix SE2000.

Existen procesos de intercambio de información entre los servidores delegacionales y los servidores de las farmacias de la Delegación correspondiente. Dichos intercambios de datos son procesos automáticos en los que el Usuario no interviene directamente.

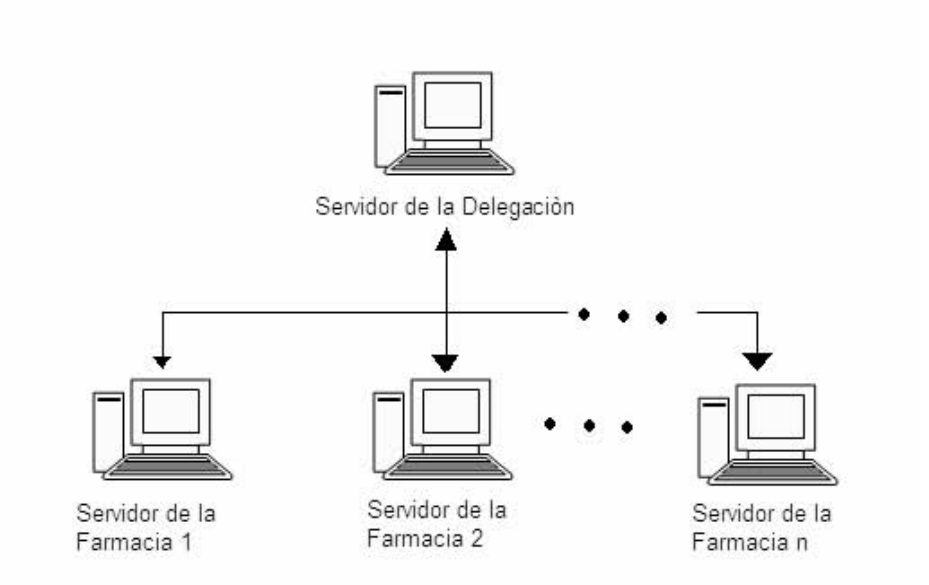


Diagrama 2.2.1

La construcción de las aplicaciones que integran el Sistema de Administración del Abasto inició hace aproximadamente ocho años y a lo largo del tiempo han sufrido modificaciones y han sido recompiladas en Delphi 5.0 y la información se tiene almacenada en Informix SE2000.

En el Diagrama 2.2.1 se muestra la organización del Sistema de Administración del Abasto.

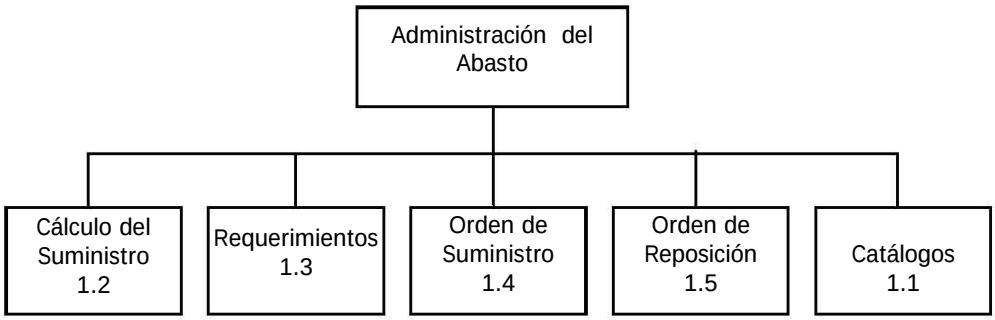


Diagrama 2.2.1

2.3 Definición del problema

El área de Sistemas de Abastecimiento del Instituto requiere migrar la base de datos *almacenes* de Informix SE2000 a Oracle 10gi, la migración incluye el esquema de datos y la información contenida en la misma.

Se ha mencionado anteriormente que la base de datos (*almacenes*) soporta la operación de las aplicaciones del Sistema de Administración del Abasto que están desarrolladas en Delphi 5.

Mientras que parte de la estrategia de cambio a nuevas tecnologías define la plataforma Microsoft .Net como el nuevo entorno de desarrollo en el Instituto el proyecto de reingeniería de sistemas del área de Abasto está planeado para iniciar aproximadamente seis meses después del cambio de las bases de datos institucionales a Oracle 10gi.

Esta circunstancia define que como parte del requerimiento de migración todas las aplicaciones Delphi del Sistema de Administración del Abasto deberán operar sin modificaciones luego del cambio de motor de base de datos.

2.4 Condiciones del desarrollo

Actualmente en el Instituto se lleva a cabo un proceso de actualización a nuevas tecnologías de la infraestructura de comunicaciones, herramientas de desarrollo de aplicaciones y certificación de la calidad de procesos internos así como de automatización de algunos de los servicios que ofrece a la población.

La estrategia de cambio define la sustitución del motor de base de datos institucional que originalmente es Informix SE2000 y que a partir de este proyecto será sustituido por Oracle 10gi.

Las consideraciones técnicas y de negocio tomadas en cuenta por el área de Innovación Tecnológica para tomar tal determinación no forman parte del alcance del presente trabajo.

Condición del desarrollo de esta tarea de migración es no modificar en su presentación o funcionamiento las aplicaciones que integran el Sistema de Administración del Abasto o alguno de sus módulos.

<table>
<tr><th colspan="2">Condiciones generales para el desarrollo</th></tr>
<tr><td>➢ Base de datos</td><td>El proceso de migración se aplicará únicamente a la base de datos en Informix SE2000 llamada almacenes.</td></tr>
<tr><td>➢ Aplicaciones y código fuente</td><td>Se considera migrar únicamente las aplicaciones que integran el Sistema de Administración del Abasto (Tabla 2.4.1).
<table>
<tr><th>Opción de Menú</th><th>Número de aplicaciones</th></tr>
<tr><td>1.1. Catálogos</td><td></td></tr>
<tr><td>1.1.1. Artículos</td><td>1</td></tr>
<tr><td>1.1.2. Unidades</td><td>1</td></tr>
<tr><td>1.1.3. Clasificación-Servicios</td><td>1</td></tr>
<tr><td>1.2. Cálculo de Suministro</td><td></td></tr>
<tr><td>1.2.1. Consumo Promedio Mensual</td><td>1</td></tr>
<tr><td>1.2.2. Frecuencias</td><td>1</td></tr>
<tr><td>1.2.3. Máximos y Mínimos</td><td>1</td></tr>
<tr><td>1.3. Requerimientos</td><td>4</td></tr>
<tr><td>1.4. Orden de Suministro</td><td>4</td></tr>
<tr><td>1.5. Orden de Reposición</td><td>4</td></tr>
</table>
Tabla 2.4.1</td></tr>
</table>

Condiciones generales para el desarrollo **…**	
➢ *Equipamiento*	El equipamiento proporcionado al grupo de trabajo encargado de la migración es el siguiente: • Servidor con sistema operativo Unix. • Instancia de Informix SE2000. • Copia de la base de datos original en la instancia de Informix SE2000. • Instancia de Oracle 10gi. • Una base de datos vacía en la instancia de Oracle 10gi. • La cuenta propietaria de la base de datos en la instancia de Oracle 10gi. Las características técnicas, de configuración, capacidad de procesamiento, etc., aplicables a los equipos utilizados como servidores, así como versiones de sistema operativo y configuración de la arquitectura de red es responsabilidad de las áreas dedicadas a cada una de estas tareas y está por lo tanto fuera del alcance del presente trabajo.
➢ *Modificaciones al código*	• No se modificará la funcionalidad de las aplicaciones en Delphi 5. • Cuando se identifiquen situaciones que exijan modificar la forma en que se procesa una tarea, se discutirá con el Usuario la solución y se implementará la misma únicamente cuando el nuevo proceso no modifique la funcionalidad de las pantallas del Usuario final. • Las modificaciones al código deberán quedar documentadas con el identificador del programador, fecha y características del cambio y el motivo de la sustitución.

Condiciones generales para el desarrollo **…**	
➢ *Administración* *de base de datos*	• El Instituto cuenta con áreas dedicadas a la administración de los servidores, sistema operativo y bases de datos, por lo que serán esas áreas las encargadas de definir y administrar posteriormente los siguientes esquemas: o Administración de las actualizaciones del motor de base de datos. o Administración y mantenimiento de la arquitectura de comunicaciones y de red. o Definición de los procedimientos de respaldo y recuperación de bases de datos. o Definición de la arquitectura física de las bases de datos en Oracle 10gi. Por lo tanto, estos puntos no son motivo de estudio en el presente trabajo.
➢ *Tiempo*	Se cuenta con un periodo de 9 semanas para el desarrollo, pruebas y puesta en marcha.

CAPITULO III

ANALISIS DE EQUIVALENCIAS Y CORRESPONDENCIA

3.1 Herramientas de migración

Actualmente existen herramientas de software con las que es posible disminuir el esfuerzo, los riesgos y el tiempo para el desarrollo del proyecto de migración.

Estas herramientas automatizan parte de las tareas del proceso, por ejemplo, obtener la definición de las estructuras de datos, el código de los programas almacenados, bibliotecas de funciones, etc., de una base de datos.

Si empleamos una herramienta de este tipo para obtener la definición de la base de datos origen, almacenando además en un repositorio intermedio dicha información podemos acortar el tiempo necesario para asociar cada estructura y tipo de datos al tipo de dato y/o estructura correspondiente en el nuevo motor de base de datos.

Podemos emplear una herramienta case para traducir el código almacenado (vistas, stored procedures, funciones, triggers) a la sintaxis correspondiente en la nueva base de datos; dicha traducción será, mecánica y automática, es decir sin requerir un conocimiento profundo de la funcionalidad de la aplicación que se transforma.

Emplear una herramienta case durante el proceso de conversión requiere mantener documentadas las reglas que indican la correspondencia de los elementos de la plataforma fuente con los elementos de la plataforma destino, por ejemplo: la correspondencia entre tipos de datos.

3.1.1 La semi-automatización

Aún empleando herramientas case, el proceso de conversión no es totalmente automático, generalmente es necesaria la participación del Usuario para modificar o precisar la correspondencia entre tipos de datos, por ejemplo; la codificación del atributo SEXO en un registro de empleados almacena valores numéricos (0:Femenino, 1:Masculino) es asociado por la herramienta de migración a un tipo de dato INTEGER.

El Usuario tendrá que intervenir para modificar esta asignación automática, definiendo el tipo de dato SMALLINT al atributo SEXO de manera que la definición del atributo requiera menor espacio de almacenamiento dado que los posibles valores del campo serán únicamente 0 o 1.
Otra posibilidad para este mismo atributo puede ser el tipo de dato CHAR(1), la asignación definitiva será entonces definida o aprobada por el Usuario final considerando el uso y manejo que se da a cada atributo en el sistema

Es posible entonces aprovechar la funcionalidad de una herramienta case de apoyo a la migración en un esquema de operación semi-automatizado de manera que el Usuario puede afinar la correspondencia de datos estándar o automática, en función de las características de uso y manejo de los datos en el sistema.

En algunos casos este tipo de herramientas toma como entrada un conjunto de reglas y un archivo con el texto de entrada, y produce como salida un archivo que corresponde al texto de entrada transformado por las reglas; en otros casos, se realiza una conexión a la base de datos origen de donde la herramienta recupera la definición de los objetos de la base de datos y genera como salida scripts de creación de objetos en la plataforma destino basándose en las correspondencias definidas y configuradas para la transformación.

De este tipo es la herramienta llamada Oracle Migration Workbench (OMW) que es la herramienta que ofrece Oracle para migrar bases de datos de diferentes plataformas a Oracle.

Dado que en el presente trabajo se tiene como meta migrar una base de datos de Informix SE2000 a Oracle 10gi emplearemos OMW como herramienta de soporte. A continuación se describen las características de OMW.

Oracle Migration Workbench (OMW) - Arquitectura

- Oracle Migration Workbench (OMW) es la herramienta básica de la tecnología de migración de Oracle.
- Es una herramienta basada en pantallas de ayuda (*wizards)* que guían al Usuario en todo el proceso de configuración y definición de reglas de transformación.
- Es una herramienta desarrollada 100% en Java
- Presenta una interfase de Usuario gráfica
- OMW soporta varios motores de base de datos: (Informix 7.3, SQL Server 6.5/7.0/2000, Sybase Adaptive Server 11&12, MS Access 2.0, 95, 97, 2000, MySQL 3.22 / 3.23, DB2/400 V4R3 iSeries, (AS/400)).

  Plug-In: El Plug-in's es el componente de software individual para cada tipo de base de datos que es posible migrar con OMW. Para migrar a Oracle una base de datos de Informix se debe instalar el plug-in de Informix. El plug-in proporciona el marco de estructuras y sintaxis para:

  - Extraer información del diccionario de datos de la base de datos original.
  - Crear el Modelo Fuente de la base de datos.
  - Convertir el Modelo Fuente al Modelo Oracle.

- El funcionamiento de OMW está basado en un repositorio de los objetos existentes en el esquema de datos origen y almacena también las reglas de transformación y las características configuradas por el Usuario en la plataforma destino de base de datos (Diagrama 3.1.1).

Oracle Migration Workbench - Arquitectura

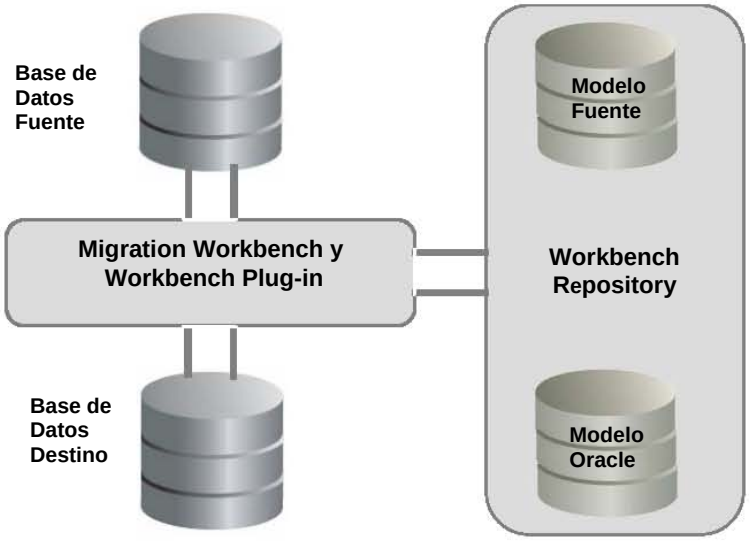


Diagrama 3.1.1

- Los objetos del esquema que migra automáticamente son:

▪ Tablas e Información	▪ Base de datos
▪ Primary Keys	▪ Stored Procedures
▪ Check Constraints	▪ Triggers
▪ Foreign Keys	▪ Grants
▪ Indices	▪ Rules
▪ Vistas	▪ Defaults
▪ Grupos / Usuarios	▪ Tipos de Dato definidos por el Usuario

En algunos casos, no migra en su totalidad el código almacenado en stored procedures y triggers; en dichas ocasiones será necesario migrar manualmente el código; esto es especialmente cuando se hace uso de tablas temporales y estructuras con campos tipo serial.

- OMW agiliza el proceso de migración al generar:
  - El *script* de creación del esquema de base de datos

- La creación de las sentencias de extracción de información de la base de datos original y los guiones de carga masiva de los datos a la plataforma destino.
    - Para extraer los datos de Informix, sentencias *unload,* una para cada tabla.
    - Para cargar la información a la base de datos Oracle, la sentencia correspondiente para Sql Loader y el archivo de control correspondiente.
- Los scripts de creación del código almacenado en la nueva base de datos (especifica los casos en que no completa la migración del código fuente).

➢ Los elementos básicos de OMW son:

- Un repositorio de datos llamado *Repositorio Workbench.*
- Un *wizard de captura* para obtener la definición de la base de datos original en Informix.
- Un *wizard de migración* para convertir el *Modelo Fuente,* recuperado con el wizard de captura, a un *Modelo Oracle*.
- Scripts para copiar los datos de la base de datos fuente a la base de datos destino.

Repositorio Workbench

El *repositorio* Workbench es un conjunto de tablas en una base de datos de Oracle donde se almacena la información del Modelo Fuente y el detalle de las estructuras correspondientes en Oracle.

Conectividad

Para conectarse a la base de datos fuente y a la base de datos destino, OMW emplea conexiones JDBC u ODBC como muestra el Diagrama 3.1.2.

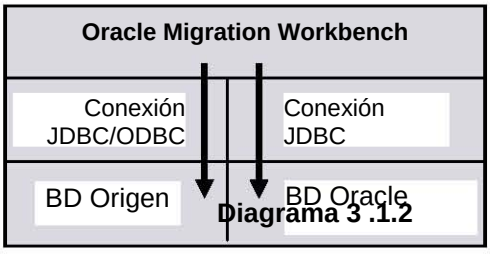


Diagrama 3.1.2

Procedimiento

El procedimiento general de migración requiere el uso del wizard de Captura y el wizard de Migración del OMW, en el Diagrama 3.1.3 se aprecia el procedimiento.

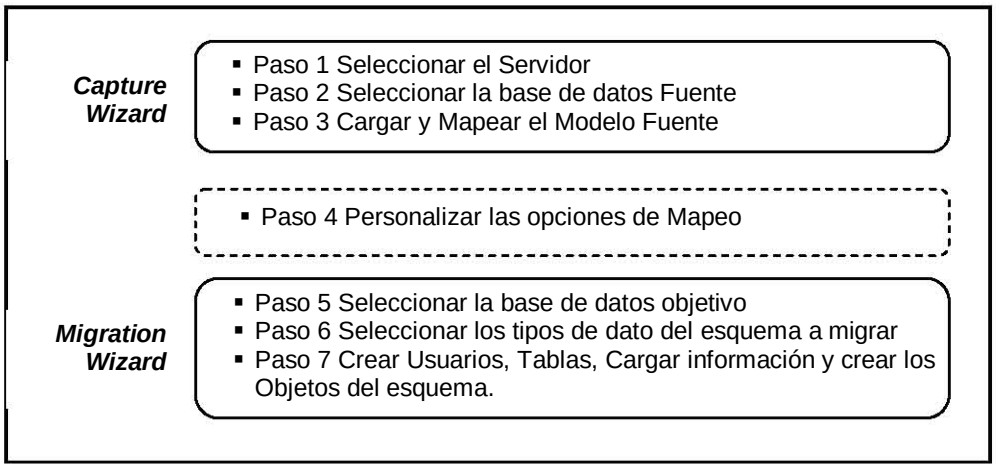


Diagrama 3.1.3

Es posible ejecutar el primer grupo de tareas y posteriormente ejecutar el segundo y/o el tercer grupo. El Repositorio Workbench mantiene el registro de la información en todo momento, de manera que es posible repetir el paso 4 o los pasos 5 a 7 las veces que sea necesario.

Funcionalidad

- Permite consolidar múltiples bases de datos en una base de datos Oracle creando
- un Tablespace por cada base de datos,
- permite retener múltiples usuarios,
- resuelve automáticamente conflictos por nombres de objetos duplicados y
- descarga datos de las tablas originales a archivos planos en caso de altos volúmenes de información.

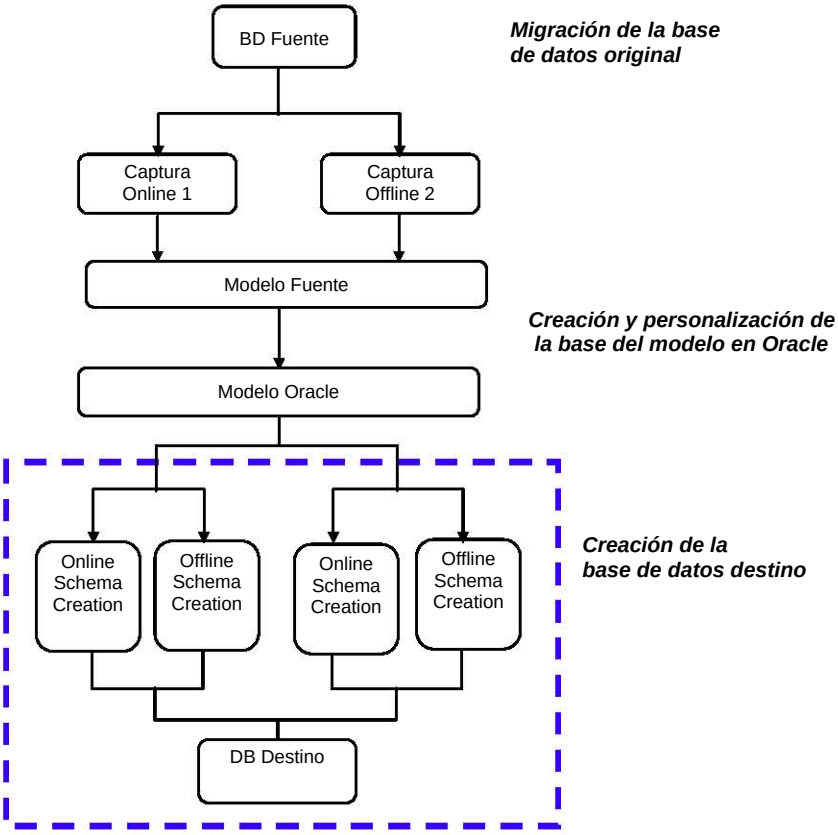
El la página siguiente se presenta el resumen de las bases de datos que es posible migrar empleando OMW instalando el plug-in correspondiente.

Características de diferentes bases de datos soportadas por OMW

Característica	**Microsoft Access** [1]	**Informix Dynamic Server**	**Sybase Adaptive Server y Microsoft SQL Server**	**MySQL**	**IBM DB2/400**	**IBM DB2 UDB**
Tablas	SI	SI	SI	SI	SI	SI
Vistas	SI (Queries)	SI	SI	N/A	SI	NO
Indices	SI	SI	SI	SI	SI	SI
Grupos/Roles	N/A	SI	SI	N/A	NO	SI
Usuarios	NO	SI	SI	SI	SI	SI
Constraints	SI (Reglas de validación)	SI	SI	SI	SI	SI
Privilegios	NO	SI	SI	SI	SI	SI
Tipos de datos definidos por el Usuario	N/A	N/A	SI	SI	SI	SI
Procedimientos almacenados	N/A	SI	SI	N/A	NO	NO
Triggers	N/A	SI	SI	N/A	NO	NO
Embedded SQL	N/A	ESQL/C a Pro*C	N/A	N/A	NO	N/A
Otras capacidades	Relaciones, tablas ligadas, código de aplicación reutilizable.	Tipos de datos *collection,* secuencias.	N/A	N/A	N/A	N/A

En el diagrama Diagrama 3.1.5 se muestra el flujo común de un proceso de migración de base de datos en el que se emplea OMW como herramienta auxiliar en el proceso.

Proceso de migración

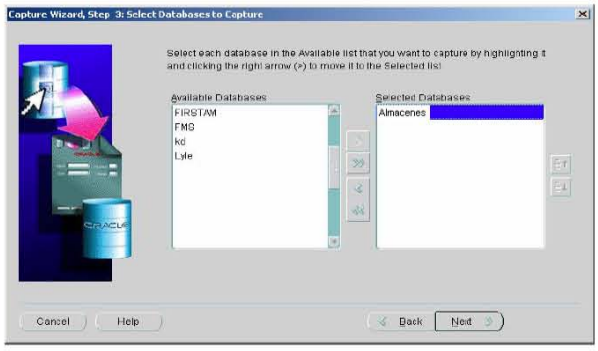


1 Captura Online Captura: con conexión directa a la base de datos.
2 Captura OffLine Captura: sin conexión directa a la base de datos; fuente de información en archivos de texto.

Diagrama 3.1.5

A continuación se muestra la secuencia general de migración empleando OMW.

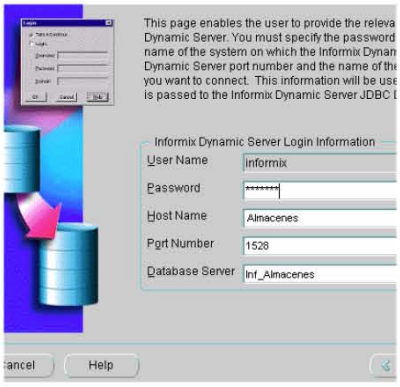
Capture wizard



El primer paso es seleccionar la base de datos original de la lista que presenta esta pantalla.

Diagrama 3.1.6

Conexión a la base de datos fuente



El *Capture Wizard* captura automáticamente la estructura de la base de datos origen una vez que se configura la conexión a la misma.

La pantalla de configuración de la conexión requiere los parámetros de conexión a una base de datos en Informix.
Esta conexión emplea los drivers de JDBC que se incluyen en el material de instalación de Informix.

Diagrama 3.1.6

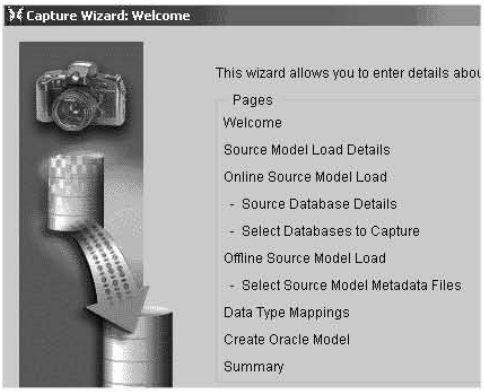
Conexión a la base de datos destino en Oracle

Para configurar la conexión a la base de datos Oracle se requiere:

- ✓ username y password de la cuenta en la que se crearán las estructuras.
- ✓ Host: El nombre del equipo en el que se instaló la base de datos Oracle.
- ✓ Port: El número de puerto TCP/IP del listener de la base de datos.
- ✓ SID: Identificador de sistema de la instancia de la base de datos en Oracle.

Diagrama 3.1.8

Entrada a OMW

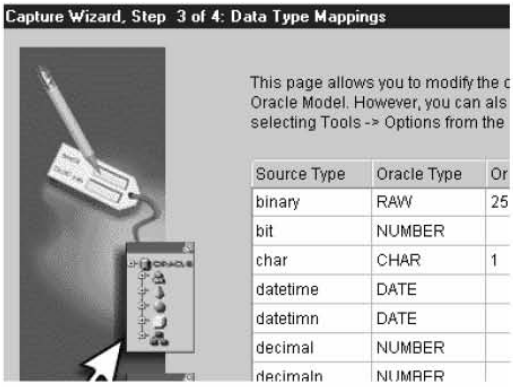


La captura de la base de datos fuente incluye:

- ✓ Carga de la base de datos fuente en el Source Model.
- ✓ Creación del Modelo Oracle.
- ✓ El Capture Wizard facilita la ejecución de estas tareas por medio de un menú controlado por pantallas.

Diagrama 3.1.9

Mapeo de Tipos de Datos



El *Mapeo* se refiere a la asociación de los tipos de datos de Informix con los correspondientes tipos en Oracle, de manera que el proceso sea automático y los cambios permanentes para cada tipo de dato original.

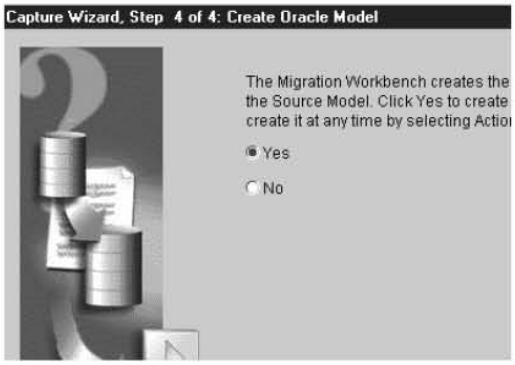
Diagrama 3.1.10

ANALISIS DE DATOS

Una vez capturada la base de datos, los datos están disponibles para el análisis en cualquier momento, de manera que es posible comparar el material del Source Model y el Oracle Model para asegurar que el contenido y estructura de la base de datos Oracle es un reflejo de la base de datos fuente. El análisis se puede realizar para:

- ✓ Generación de reportes de migración
- ✓ Revisión de los mensajes de log
- ✓ Revisión de dependencias entre objetos

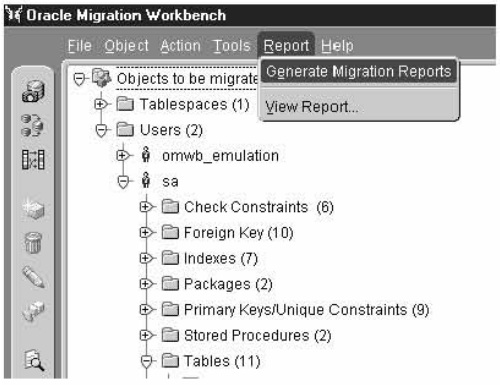
Creación del modelo destino en Oracle



La siguiente pantalla genera el modelo (script) de creación de la base de datos en Oracle.

Diagrama 3.1.11

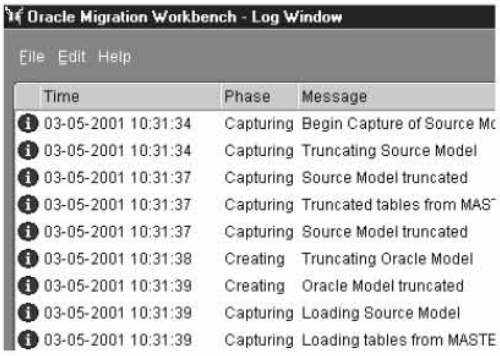
Revisión de los reportes de Migración



Menú de Reportes

Diagrama 3.1.12

Revisión de los mensajes de Log



Consulta del Log de Migración

Diagrama 3.1.13

Vista de estructuras

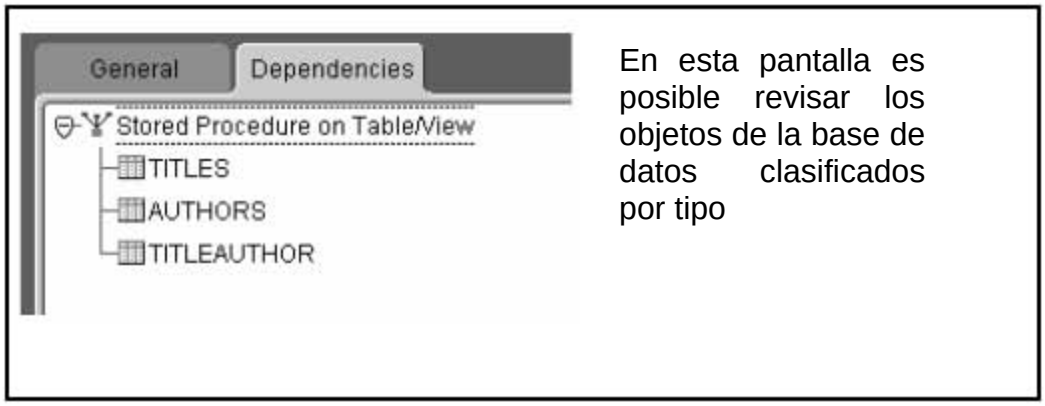


Diagrama 3.1.14

Adecuaciones a los tipos de datos

Los cambios aplicados al Modelo en Oracle determinan la forma en que el contenido en el Modelo Oracle va a ser creado físicamente en la base de datos destino en Oracle. Estos cambios permanecen en el repositorio de Workbench hasta que el Modelo de Oracle sea regenerado.

Source Model Customization

En el diagrama Diagrama 3.1.15 se muestra la pantalla de modificación del mapeo de datos.

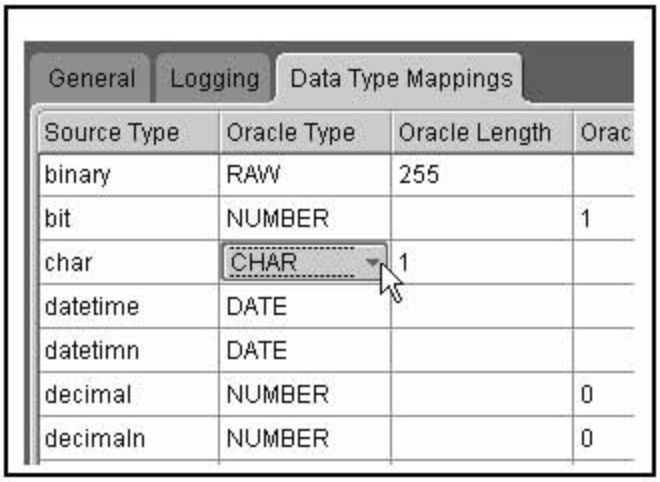


Diagrama 3.1.15

Personalización del Modelo Fuente

Para eliminar elementos del modelo capturado, se pulsa el botón derecho sobre el elemento seleccionado y se pulsa la opción Delete (Diagrama 3.1.16).

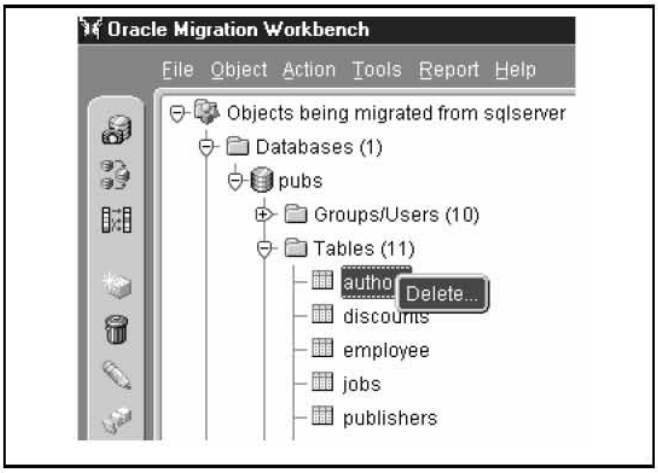


Diagrama 3.1.16

Edición del Texto y Opciones de parking

Es posible editar el texto de procedimientos almacenados, triggers y vistas, esto proporciona un medio para remover las características no soportadas en Oracle, así, cada vez que se generan los scripts de creación de la base de datos destino los cambios se verán reflejados (Diagrama 3.1.17).

Por ejemplo, la opción Auto Select permite al Usuario determinar si el código almacenado original debe ser convertido en Funciones o en Procedimientos en la nueva base de datos. Otra de las características que se pueden habilitar o inhibir es la generación de excepciones (*Exceptions*) en el código almacenado en Oracle.

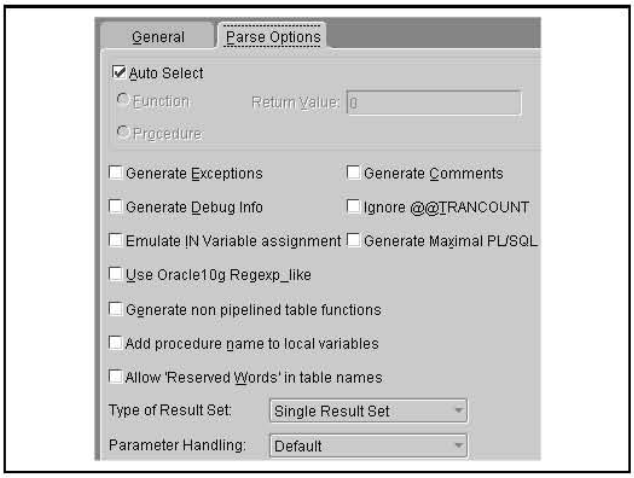


Diagrama 3.1.17

Borrado de tablas en el modelo capturado

Seleccionando el objeto a eliminar, del menú seleccionar Object y a continuación la opción *Delete* (Diagrama 3.1.18).

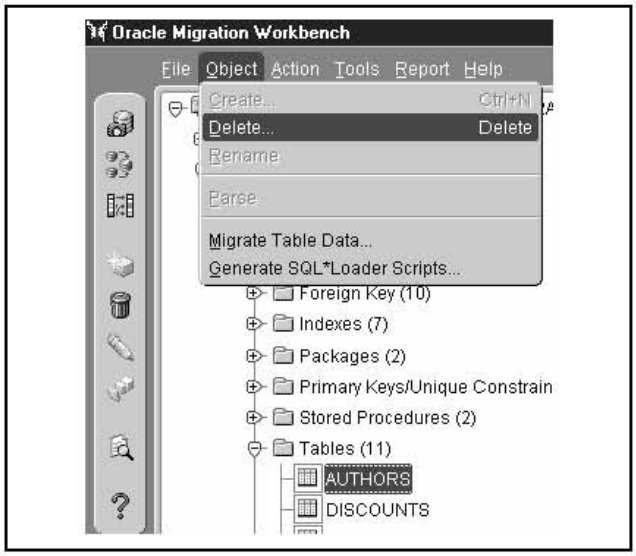


Diagrama 3.1.18

Tablespaces

Si los tablespaces han sido definidos ya en la base de datos de destino en Oracle, es posible emplear las herramientas del administrador que proporciona Oracle, como Discoverer, para obtener los detalles de los *tablespaces* definidos en la base de datos destino (Diagrama 3.1.19).

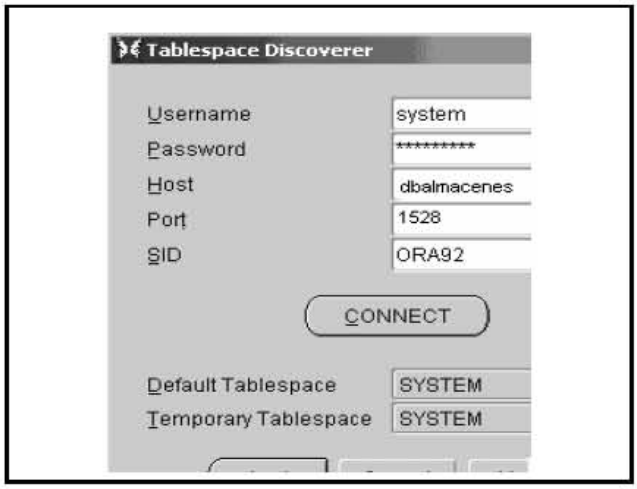


Diagrama 3.1.19

Edición de texto

Ejemplo de la pantalla de edición de texto de código almacenado (stored procedures, triggers, views) en el Diagrama 3.1.20.

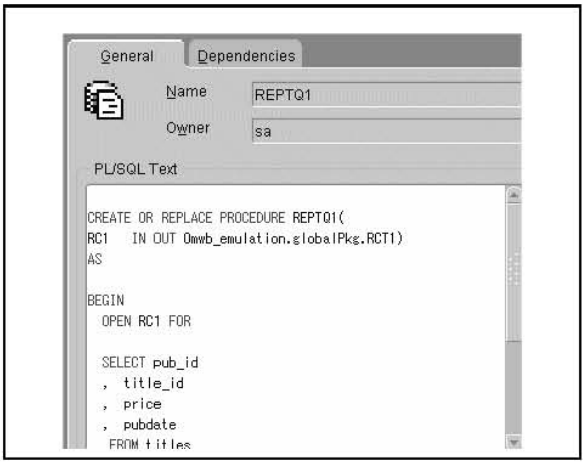


Diagrama 3.1.20

Particularización del modelo de datos en Oracle

La definición de la base de datos en Oracle incluye personalizar las opciones de creación de Usuarios, y los parámetros de asignación de espacio para las tablas y los índices de la base de datos como se muestra a continuación.

Edición de Usuarios

Para modificar las opciones de creación de los Usuarios (passwords y tablespaces default) de manera individual o para un grupo específico se emplea la pantalla siguiente (Diagrama 3.1.21).

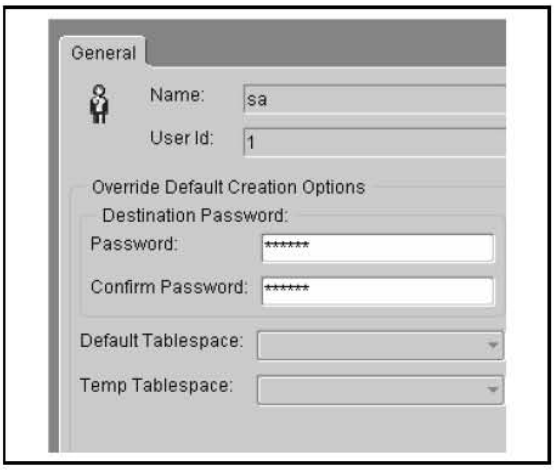


Diagrama 3.1.21

Edición de Tablas e Índices

Las opciones de almacenamiento para tablas e índices en el Modelo Oracle, incluyendo el tamaño y extents de las tablas y el tablespace destino de las mismas se modifica en la pantalla siguiente.

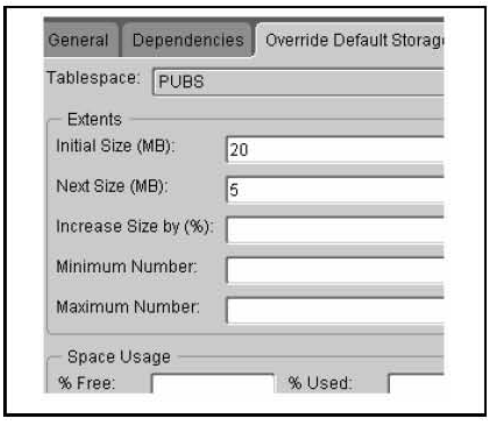


Diagrama 3.1.22

Data Migration

El Migration Wizard migra todas las tablas de la base de datos fuente a la base de datos destino automáticamente. El menú Migrate Table Data envía la información de una tabla específica hacia el Modelo Oracle.

Sin embargo, para migrar altos volúmenes de información de manera eficiente es recomendable emplear la utilería SQL*Loader de Oracle; el primer paso consiste en descargar la información de las tablas originales en archivos planos, un archivo para cada tabla; el segundo paso consiste en insertar directamente la información contenida en los archivos planos hacia la base de datos nueva en Oracle empleando Sql*Loader.

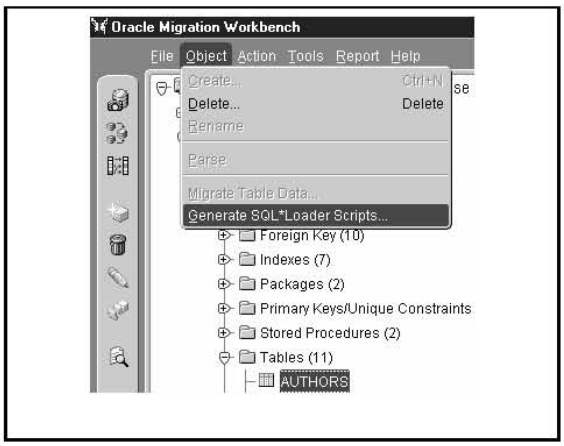


Diagrama 3.1.23

Se requiere un archivo de control con las directivas requeridas por Sql*Loader para cada tabla a cargar con información. OMW genera los scripts de carga para todas las tablas desde la opción de menú que ilustra el Diagrama 3.1.23.

3.2 Análisis de diferencias y equivalencias de estructuras de datos

Es posible encontrar información altamente detallada al respecto en libros, revistas, manuales especializados e Internet, por lo que a continuación, se señalan las características de los objetos de base de datos en Informix, y las anotaciones correspondientes al tipo de dato o implementación equivalente en Oracle.

La tabla 3.2.1 se muestran las similitudes de tipos de objetos entre Oracle 10gi e Informix.

Objetos de esquema en Oracle e Informix

Oracle	Informix
Database	Database
Schema	Schema
Tablespace	Dbspace
User	User
Role	Role
Table	Table
Temporary tables	Temporary tables
Index Cluster	Index
Check constraint	Check constraint
Column default	Column default
Unique key	Unique key
Primary key	Primary key
Foreign key	Foreign key
Index	Index
PL/SQL Procedure	SPL Procedure
PL/SQL Function	SPL Function
Packages	N/A
AFTER triggers	Triggers
BEFORE Triggers	Triggers
Triggers For each row	Triggers for each row
Synonyms	Synonyms
Sequences	SERIAL datatype for a column
Snapshot	N/A
View	View

Tabla 3.2.1

En la Tabla 3.2.2 se muestra un resumen de correspondencias entre tipos de datos Informix-Oracle.

Estas correspondencias están configuradas por default en OMW, de manera que una primera versión de la base de datos migrada a Oracle cumplirá con la definición de estas correspondencias.

Tipos de datos

Informix	Descripción	Oracle	Observaciones
INTEGER INT	Almacena números en el rango -2,147,483,647 a +2,147,483,647	NUMBER(10)	Es posible asociar un check constraint para asegurar el rango de valores de manera que se eviten valores por over flow.
SMALLINT	Almacena valores en el rango -32,767 a +32,767.	NUMBER(5)	Es posible asociar un check constraint para asegurar el rango de valores.
SERIAL	Almacena valores enteros secuenciales asignados automáticamente por el servidor de base de datos cada vez que un nuevo registro es almacenado.	NUMBER(10)	Este tipo de dato se sustituye con una Secuencia y un Trigger para actualizar automáticamente la columna que originalmente era de tipo SERIAL.
DECIMAL DECIMAL(P) Floating point DEC DEC(P)	Genera un número con punto flotante con p dígitos de precisión. Si se omite p, el valor de default es p.	NUMBER	Un número de punto flotante con 38 dígitos de precisión.
DECIMAL(p,s) DEC(p,s)	Un número de punto fijo con precisión p y escala s.	NUMBER(p,s)	Un número de punto fijo con precisión p y escalas.

Tipos de datos . . .

Informix	Descripción	Oracle	Observaciones
SMALLFLOAT REAL	Almacena un número de precisión simple, los números de punto flotante corresponden al tipo flota en C.	FLOAT(63)	
FLOAT(p) DOUBLE PRECISION	Almacena un número de doble precisión con punto flotante correspondiente al tipo de datos *double* en C. p especifica una precisión.	FLOAT(126)	
CHAR(n) CHARACTER(n))	Cadenas de caracteres de n 8-bits, rellenos con blancos. 0< *n* < 32768	Si n <= 2000 el dato en Oracle corresponde a un CHAR(n). Si 2000 < n <= 4000, el dato similar en Oracle es VARCHAR2(n). Si n > 4000, el tipo de dato en Oracle es CLOB o LONG.	Oracle CHAR almacena hasta 2000 bytes de datos. Oracle VARCHAR2 almacena hasta 4000 bytes de datos. Oracle LONG soporta 2G de información, pero existen restricciones sobre columnas LONG, Oracle CLOB puede almacenar hasta 4G.
VARCHAR(n)	Cadenas de caracteres de longitud variable. 0 < *n* < 256.	VARCHAR2(n)	

Tipos de datos . . .

Informix	Descripción	Oracle	Observaciones
DATETIME	Almacena la fecha calendario y el momento del día. La precisión del tipo DATETIME incluye: YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, FRACTION.	DATE	El tipo de datos DATETIME en Informix tiene una mayor precisión, YEAR hasta Fraction of Second
DATE	El valor DATE se almacena internamente como un entero igual al número de días desde Diciembre 31 de 1899.	DATE	Almacena una fecha y hora, el valor por defecto es de 12:00 AM (medianoche).
BYTE	Almacena cualquier tipo de datos binaries, hasta 2GB.	BLOB LONG RAW	Puede haber más de una columna de tipo BYTE por tabla.
TEXT	Almacena cualquier tipo de datos de tipo texto, hasta de 2G. Una tabla puede tener más de una columna de tipo TEXT.	CLOB LONG	El tipo de dato CLOB puede almacenar hasta 4G de caracteres. Puede haber más de una columna de tipo CLOB. LONG tiene un límite de 2G pero existen restricciones para este tipo de datos. BYTE almacena cualquier tipo de datos binarios hasta 2G. Una tabla puede tener más de una columna de tipo BYTE.
INTERVAL		NUMBER(p)	Donde p es la precisión del valor calificador más grande.
MONEY(p,s)		NUMBER(p)	

Tabla 3.2.2

A continuación se describen consideraciones generales a tomar en cuenta en el proceso de migración organizadas como se lista a continuación:

1. Esquema de base de datos
2. Tipos de datos
3. Tipo de datos IMAGE y TEXT (Binary Large Objects)
4. Integridad referencial
5. Indices
6. Cláusula Matches
7. Sentencias no soportadas en Oracle
8. Defaults
9. Tablas temporales

Dado que en la base de datos que da origen a este trabajo no se tienen módulos programados en ESQL/C se excluye el análisis de estos componentes.

El detalle para cada categoría de datos es el siguiente.

1. ***Esquema de datos***

1.1 Versión de Informix SE2000

Existen herramientas de migración específicas para versiones determinadas de Informix y Oracle dado que cada del motor tiene características particulares.

En el caso de OMW existe un plug-in con la configuración de correspondencias específico para cada base de datos destino.

1.2 Tipo de Sistema (OLTP, OLAP, data warehouse etc.)

El número y tamaño de redo logs y el tablespace de Undo dependen del tipo de sistema, esto afecta también la configuración del disco y en consecuencia el número y tamaño de los tablespaces para la base de datos Oracle.

1.3 Tamaño total de la base de datos de desarrollo (y/o producción) en MB.

Se requiere reservar espacio suficiente para la base de datos correspondiente en Oracle y, determinar las características de los data files que alojarán los datos durante la transferencia entre Informix y Oracle. Esta información determinará también la especificación de: INITIAL, NEXT, PCTINCREASE, MINEXTENTS y MAXEXTENTS de los tablespaces en Oracle.

1.4 Nomenclatura de objetos

Es posible que el nombre de alguno de los objetos en Informix corresponda a una palabra reservada en Oracle, en tales casos será necesario considerar agregar un diferenciador como el guión bajo (_) al nombre original, de manera que no haya problemas con las palabras reservadas o los nombres de objetos repetidos en diferentes esquemas de usuario.

Por ejemplo, en Informix es posible definir una tabla con columnas de nombre DESC, LOG, SQL; en Oracle se trata de palabras reservadas, de manera que dichos nombres pueden ser reemplazarse por DESC_, LOG_, SQL_.

Las referencias a dichas columnas en bloques de código almacenado (triggers, stored procedures, vistas) y en el código de las aplicaciones cliente debe ser actualizado.

2. *Objetos del esquema*

2.1 Tablas

El número de tablas impacta el tiempo estimado de migración del esquema. Si se emplea una herramienta case como Oracle Migration Workbench, el proceso será automatizado en su mayor parte.

El tamaño y el uso de las tablas afectarán la ubicación de las mismas en el conjunto de tablespaces de la base de datos. Para ello es conveniente identificar:

- Las tablas más grandes (en número y tamaño).
- Las tablas con mayor número y volumen de transacciones.
- Tablas de búsqueda, catálogos, en número y tamaño

Con la información anterior, el administrador de la base de datos deberá asignar el tablespace para el almacenamiento de cada una de las estructuras, a menos que desde el inicio se especifique distribuir a las estructuras de datos tal y como están organizadas en la versión inicial en Informix.

2.2 Vistas

Puede ser necesario convertir manualmente las vistas complejas por lo tanto deben de ser clasificadas por su complejidad (simple, promedio y compleja).

2.3 Tipos de datos

2.3.1 Caracter

Se identifican las Tablas con columnas tipo caracter de más de 2000 y menos de 4000 caracteres, el tipo de dato sugerido para la migración es CHAR puede mantener hasta 2000 caracteres.
Para un campo de hasta 4000 caracteres una columna tipo Oracle VARCHAR que almacena hasta 4000 caracteres puede ser suficiente.

Para los campos tipo text de Informix se puede emplear una columna tipo LOB (por ejemplo: CLOB-caracter - o BLOB-binary) que puede almacenar hasta 4 gigabytes.
El tipo de dato Oracle LONG no puede ser utilizada en versiones posteriores a Oracle 9i.

2.3.2 DATETIME Data Types

El tipo de datos DATETIME de Informix Dynamic Server llega a 5 posiciones decimales, 1/100000000-ésimo de Segundo. El tipo de dato TIMESTAMP de Oracle 10gi tiene una precision de 1/100000000-ésimo de Segundo y también cuenta con la definición del tipo de dato DATE que almacena la fecha y la hora hasta segundos.

Oracle Migration Workbench está configurado con DATE por default para los procesos de conversión.

2.3.3 Datos IMAGE y TEXT (Binary Large Objects)

Los métodos físicos y lógicos para almacenar los datos tipo BYTE y TEXT en Informix es similar en el almacenamiento de datos tipo BLOB en Oracle. La información tipo BYTE o TEXT se almacena separadamente de los registros de la tabla, asociando cada registro por medio de apuntadores. Es posible almacenar tener varias columnas de tipo IMAGE o TEXT por tabla. Oracle almacena datos tipo IMAGE en campos tipo BLOB y campos TEXT en campos CLOB.

Si la información en la columna de tipo TEXT no excede 4000 bytes, es conveniente almacenar esa información en un atributo de tipo VARCHAR2 en Oracle en lugar de un campo CLOB.

2.4 Integridad referencial

En ocasiones será necesario reescribir manualmente los constraints de validación para asegurar la sintáxis correspondiente en Oracle.

2.5 Tablas con columnas tipo SERIAL

Las columnas de tipo Informix SERIAL generalmente se convertirán en columnas de tipo NUMBER; asociando además una secuencia y un trigger por default a la columna correspondiente para implementar la numeración automática del campo correspondiente.

De esta manera se mantiene la funcionalidad de generación automática de valores para la columna que originalmente era tipo SERIAL.

2.6 Tablas con columnas INTERVAL

El tipo de dato INTERVAL de Informix se transforma a un tipo de dato NUMBER. En Oracle 8 no se tiene un tipo de dato INTERVAL. A partir de Oracle9i se pueden emplear tipos de datos INTERVAL YEAR TO MONTH o INTERVAL DAY TO SECOND para convertir manualmente el tipo de dato.

2.7 Triggers y programas almacenados

Cuando no se emplea una herramienta case para la migración de código almacenado correspondiente a triggers, procedimientos almacenados y vistas, será necesario clasificar los triggers y procedimientos almacenados según su complejidad (simple, promedio, compleja) para evaluar con mayor precisión el esfuerzo que se requerirá para migrar dichos objetos.

2.8 Stored procedures que usan tablas temporales

Si durante la conversión de un stored procedure se identifica el uso de una tabla temporal, es conveniente integrar la definición de la tabla en el script general de la base de datos declarándola como global temporary table, adicionalmente será necesario reemplazar las sentencias Create y Drop asociadas a la tabla temporal por sentencias Delete.

Al compilar un procedimiento almacenado que incluye una sentencia Create o Drop se generará un error de compilación que permitirá identificar rápidamente los casos de código almacenado que incluyen creación de tablas temporales o definitivas.

2.9 Procedimientos almacenados que emplean cursores de barrido de retorno

PL/SQL no soporta retorno de cursores de barrido, sin embargo, es posible simular los cursores de barrido empleando tablas PL/SQL. No es posible determinar una solución única, cada caso deberá ser analizado para determinar la mejor solución, seleccionando entre tablas PL/SQL y consultas directas o tablas temporales.

2.10 Procedimientos almacenados que regresan cursor (por ejemplo, empleando la directive "return with resume")

Esta funcionalidad no está soportada en Oracle, pero es posible emular el result set que regresa Informix como resultado de un procedimiento almacenado, agregando una variable REF CURSOR a la lista de parámetros del procedimiento PL/SQL equivalente.

2.11 Formas en Informix 4GL

Existen herramientas de terceros auxiliares en la automatización del proceso de conversión o de alguno de sus productos. El uso de las mismas depende de la disponibilidad de licencias y/o soporte en el uso de los mismos.

Es posible emplear Oracle Designer para rediseñar las formas 4GL en este caso es el equivalente a rediseñar manualmente las formas originales.

3. *Documentación*

Identificar la documentación asociada al sistema, relacionada con Flujo de procesos, funcionalidad de aplicaciones, documentación de Usuario, etc.

Estos materiales permitirán identificar y resolver problemas potenciales y proporcionan una guía en el proceso de pruebas unitarias de los productos a convertir.

4. *Indices*

Se puede emplear una herramienta case como Oracle Migration Workbench para generar las sentencias DDL de los índices identificados. En caso de no utilizar OMW, es necesario considerar el número de índices en la estimación de tiempo necesario para la conversión.

5. *Cláusula MATCHES*

La cláusula Matches de Informix es una función que nos permite localizar expresiones de tipo cadena de caracteres hasta cierto punto complejas. Aunque no se tiene un equivalente en Oracle, se puede construir una funcionalidad similar con cláusulas LIKE y SUBSTRING de manera manual.

MATCHES	Sentencia	Conversión	Resultado
MATCHES	’[A-Z]’	BETWEEN	’A’ AND ‘Z’
MATCHES	'[abcdefg]'	IN	('a','b','c','d','e','f','g')
MATCHES	'*tennis*'	LIKE	'%tennis%'
MATCHES	'?ennifer*'	LIKE	'_ennifer%'
MATCHES	'[^qwerty]'	NOT	IN ('q','w','e','r','t','y')
MATCHES	'[^a-z]	NOT	BETWEEN 'a' AND ‘z’

6. *Sentencias no soportadas en Oracle*

Todas las sentencias DDL dentro de procedimientos almacenados DBINFO('sqlca.sqlerrd1'), se requiere conversión manual.

DBINFO(DBSPACE, number), se requiere conversión manual.

Todas las sentencias SET excepto SET DEBUG FILE, requiere conversión manual.

7. *Defaults*

Las siguientes restricciones aplican a los valores por defecto para los objetos de sistema.

- **Nombres de objetos:** El nombre de un Usuario puede ser de hasta 8 caracteres de largo; en Oracle el máximo es de 30 caracteres.
- El *usuario* propietario de la base de datos debe declararse como CHAR(30) en Oracle; en Informix la longitud del usuario es de 8 caracteres, pero al exportarlo, la definición cambiará a 30 caracteres completado con blancos.
- **Indexes:** Es necesario migrar primero los indices, luego los unique constraints y al final las primary keys. De lo contrario se crea un índice de sistema para cada unique contraint y primary index.
- **Check Constraints** Los constraints de ckeck no son transformados automáticamente a Oracle, de manera que será necesario revisar las sentencias de creación de los mismos.
- **Check Constraint Owners** Si un Usuario crea un ckeck constraint sobre una tabla que pertenece a otro Usuario, el check constraint será propiedad del dueño de la tabla.

3.3 Análisis para la migración de aplicaciones

Las indicaciones de este apartado están orientadas a describir los ajustes necesarios al código de las aplicaciones en Delphi 5.

Estas indicaciones son resultado de las pruebas realizadas ejecutando las aplicaciones con una conexión a la base de datos en Oracle.

El procedimiento seguido para estas pruebas es el que se muestra en el Diagrama 3.3.1.

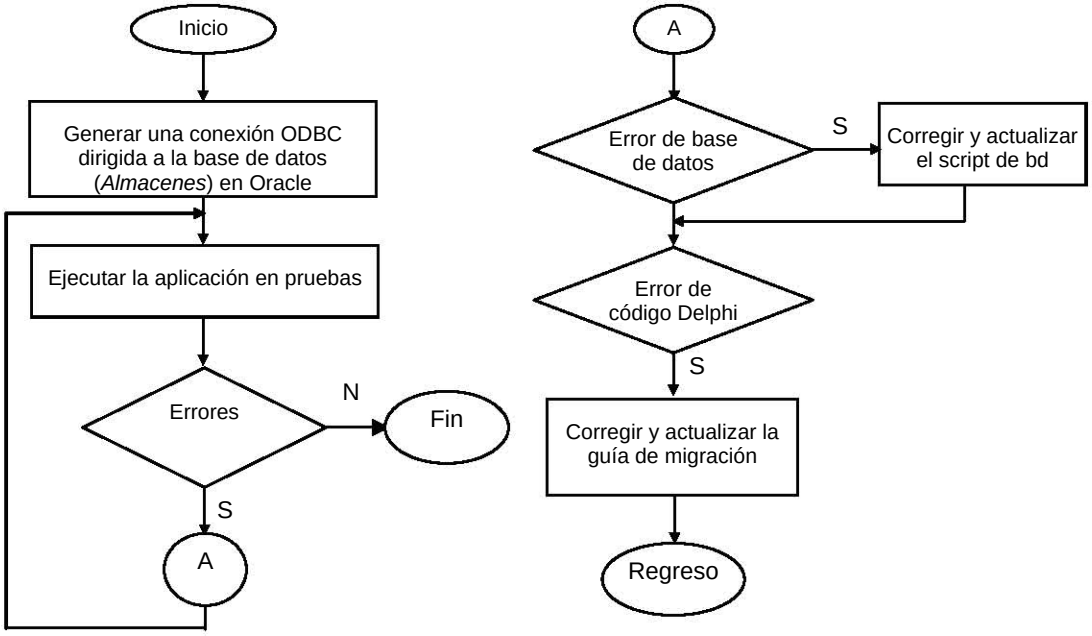


Diagrama 3.3.1

El conjunto inicial de actualizaciones requeridas es el siguiente.

- Actualizar las cadenas de acceso a la base de datos
- Actualización de la sintaxis de las sentencias de consulta o modificación a objetos de la base de datos.
- Los nombres de las estructuras de base de datos, tales como tablas, funciones, procedimientos almacenados, en mayúsculas, para evitar el mensaje de *…"… el objeto no existe*…".
- Los nombres de las tablas, procedimientos almacenados, vistas e índices deberán reescribirse en mayúsculas.

Identificación de elementos

Antes de describir los ajustes al código de las aplicaciones Delphi 5 que son consecuencia del cambio de motor de base de datos, es necesario revisar algunos conceptos básicos asociados a una aplicación desarrollada en Delphi 5.0.

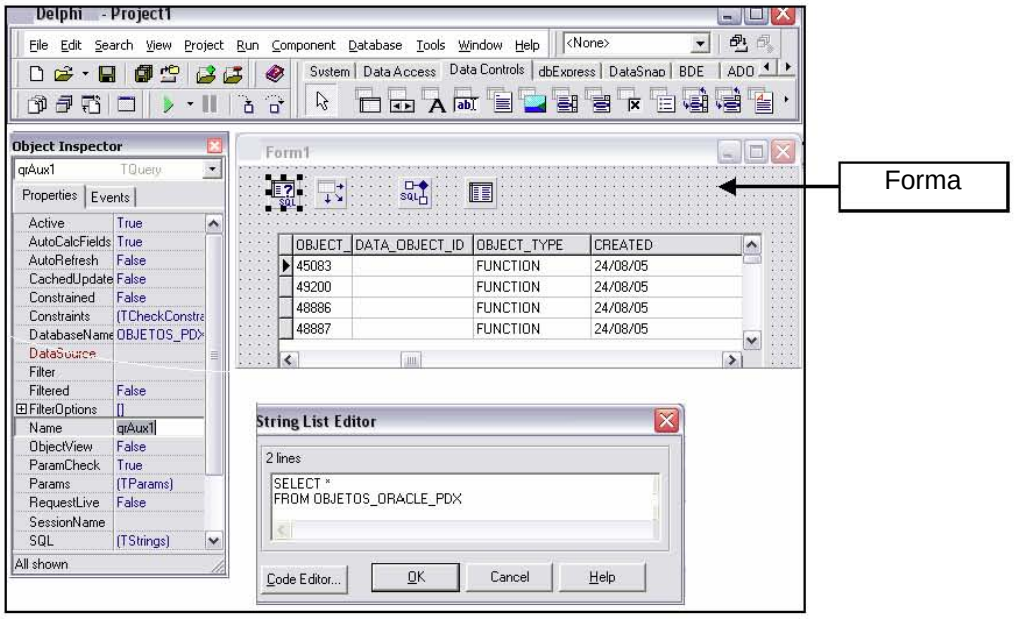


Diagrama 3.3.1

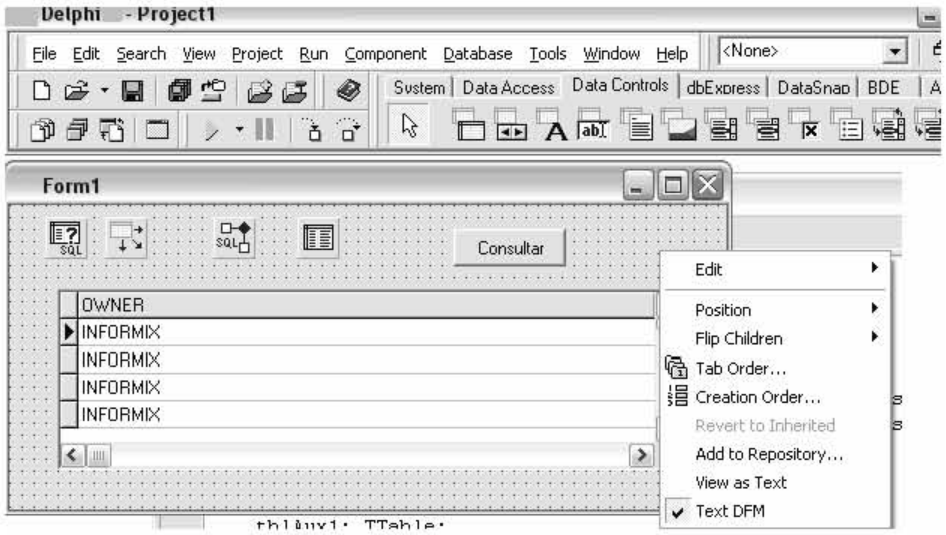
El código fuente de una aplicación desarrollada en Delphi 5.0, está integrada por los siguientes elementos.

3.3.1. ***Formas*****:** son la interfase con la que el Usuario interactúa con el sistema (Diagrama 3.3.1).

La *forma* es el elemento gráfico que proporciona la funcionalidad que el Usuario puede ejecutar, de acuerdo a las tareas asociadas a cada elemento del lienzo o forma.

Una forma se almacena en archivos de extensión .DFM.

Para ver el código del archivo .DFM de una forma, pulse el botón derecho del Mouse, del menú auxiliar seleccione la opción View as Text.

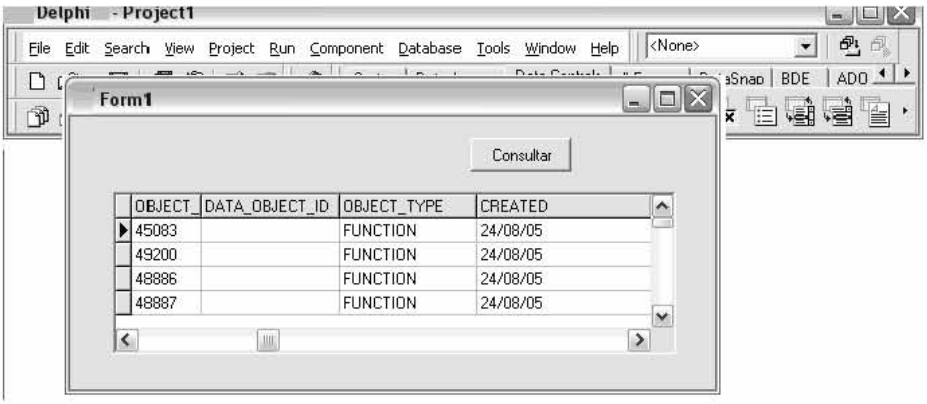



```
  object tblAux1: TTable                OBJETO tTable
    Left = 192
    Top = 8
  end
  object qrAux1: TQuery                 OBJETO TIPO tQuery
    Active = True
    DatabaseName = 'OBJETOS_PDX'
    SQL.Strings = (
      'SELECT * '
      'FROM OBJETOS_ORACLE_PDX')        CONSULTA
    Left = 24
    Top = 8
  end
  object spAux1: TStoredProc            OBJETO tStoredProcedure
    Left = 136
    Top = 8
  end

end
```

Diagrama 3.3.2

En el ejemplo anterior se identifica el código asociado a un objeto tTable, tQuery, tStoredProcedure (Diagrama 3.3.2).

3.3.2. ***Programación de la Forma:*** Para cada archivo con extensión .DFM se tiene al menos un archivo .PAS, en el que se almacena el código asociado a los eventos programados de los elementos gráficos en la forma. Para ver el código asociado a un objeto de la forma, pulse doble click sobre la forma o sobre cualquiera de sus elementos Diagrama 3.3.3.




```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, StdCtrls, Buttons, Grids, DBGrids, DB, DBTables;

type
  TForm1 = class(TForm)
    tblAux1: TTable;
    qrAux1: TQuery;
    spAux1: TStoredProc;
    DsAux1: TDataSource;
    dbgAux1: TDBGrid;
    BitBtn1: TBitBtn;
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

procedure TForm1.BitBtn1Click(Sender: TObject);
begin
qraux1.close;
qraux1.sql.clear;
qraux1.sql.add('SELECT * FROM objetos_oracle_pdx');
qraux1.active := true;
end;

{$R *.dfm}
end.
```

Diagrama 3.3.3

3.3.3. *Componentes TTable*

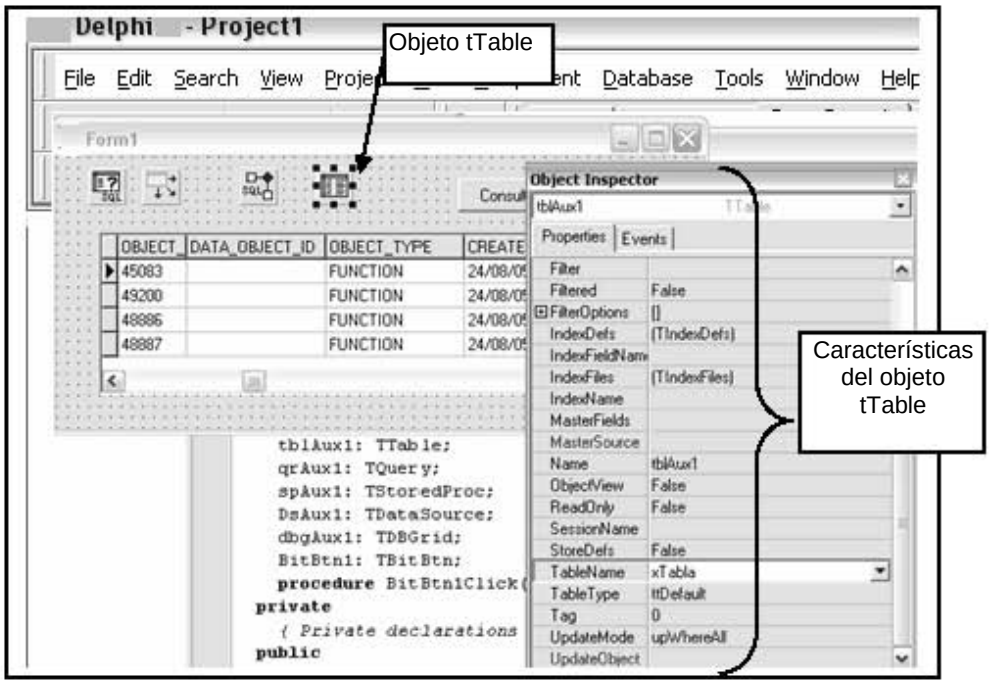


Diagrama 3.3.4

Actualización necesaria

- Modificar el nombre de la tabla o vista a letras mayúsculas.

Es necesario realizar esta modificación debido a que las librerías de Oracle 10gi no cubren completamente el requerimiento de conexión desde aplicaciones Delphi 5, únicamente se tiene documentada la conexión a versiones de Delphi 2005 por parte del proveedor.

Únicamente desde el entorno natural SqlPlus de Oracle es posible referirse a los objetos sin necesidad de cambiar a mayúsculas el nombre de la tabla asociada al objeto tTable.

En el Diagrama 3.3.4 se muestran las características de un objeto tTable; en la línea `TableName : xTabla` la línea deberá quedar como:

```
TableName: XTABLA
```

3.3.4. ***Componentes tStoredProcedure:*** Los componentes tStoredProcedure de Delphi son objetos que proporcionan la funcionalidad necesaria para emplear un stored procedure de la base de datos el Diagrama 3.3.5 muestra un ejemplo.

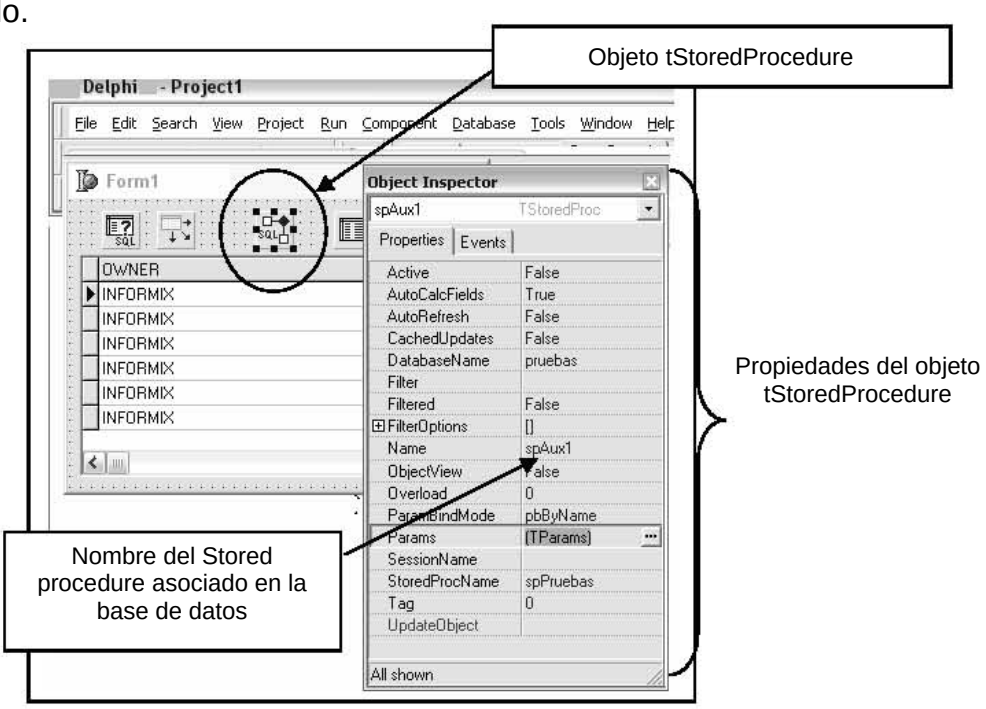


Diagrama 3.3.5

Ajustes al código

Luego de analizar las características del código Delphi 5 en las referencias a la base de datos los ajustes necesarios para actualizar la funcionalidad de las aplicaciones con la base de datos en Oracle son las siguientes.

- Cambiar a mayúsculas el nombre del Procedimiento.
  En el ejemplo anterior, el nombre del procedimiento almacenado es:

  `Name: spAux`; después de los cambios, deberá ser:

  `Name: SPAUX`

  El cambio mencionado es similar a la modificación asociada a los objetos tTable, en general, los nombres a los objetos de base de datos deberán quedar registrados en mayúsculas para evitar los mensajes de error de "…*el objeto no existe…".*

- Revisar en la base de datos en Oracle los nombres de los parámetros del procedimiento y registrarlos en la sección Params de la lista de propiedades del objeto tStoredProcedure.

- Asignar el valor 0 a los parámetros tipo Cursor.

3.3.5. ***Componentes tQuery:*** Los componentes tipo tQuery son objetos a los que se asocia una sentencia DML (SELECT, UPDATE, INSERT o DELETE), en el Diagrama 3.3.6 se muestra un objeto tQuery; el Diagrama 3.3.7 muestra la definición de la consulta asignada al mismo objeto.

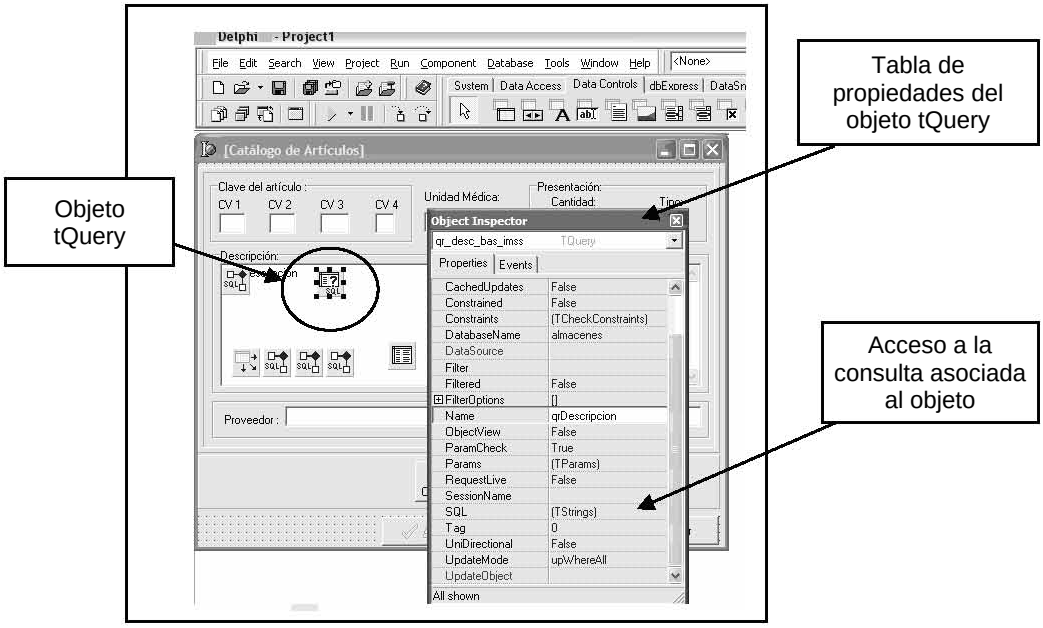


Diagrama 3.3.6

Al pulsar la sección SQL: (TStrings) se muestra el detalle de la consulta asociada al Query qrDescripcion (Diagrama 3.3.7).

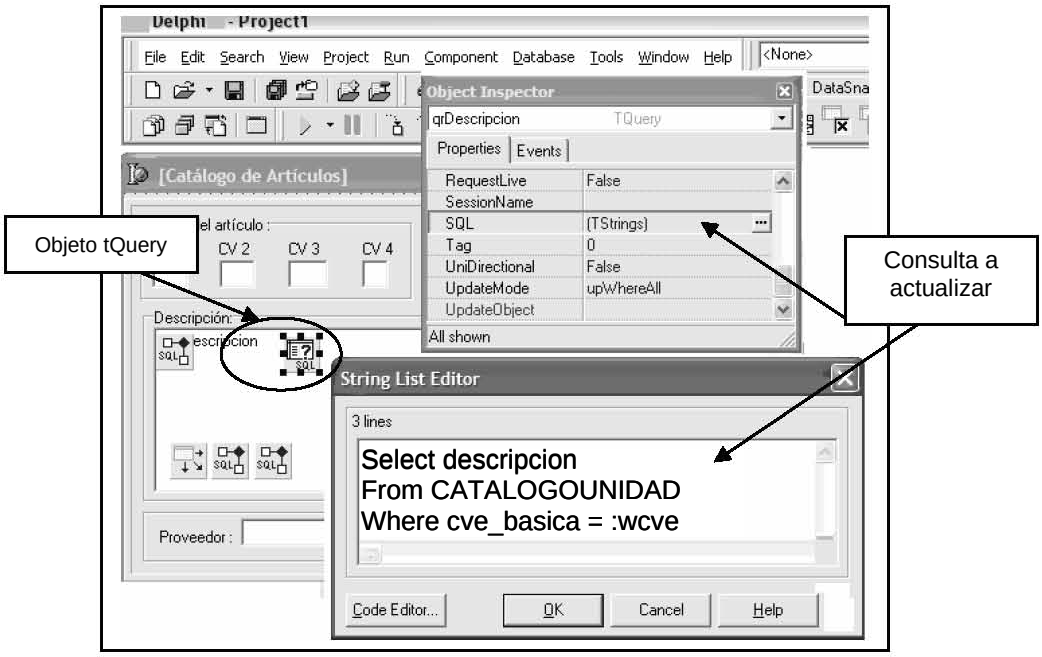


Diagrama 3.3.7

Los ajustes correspondientes al código Delphi 5 después de actualizar la base de datos son los siguientes.

- Actualizar a mayúsculas los nombres de tablas a las que se hace referencia en el código. En el Diagrama 3.3.5 la sentencia original es:

```
SELECT descripcion
FROM   catalogounidades
WHERE  cve_basica = :wcve
```

Al cambiar a mayúsculas los nombres de las estructuras, la consulta debe quedar como sigue:

```
SELECT descripcion
FROM   CATALOGOUNIDADES
WHERE  cve_basica = :wcve
```

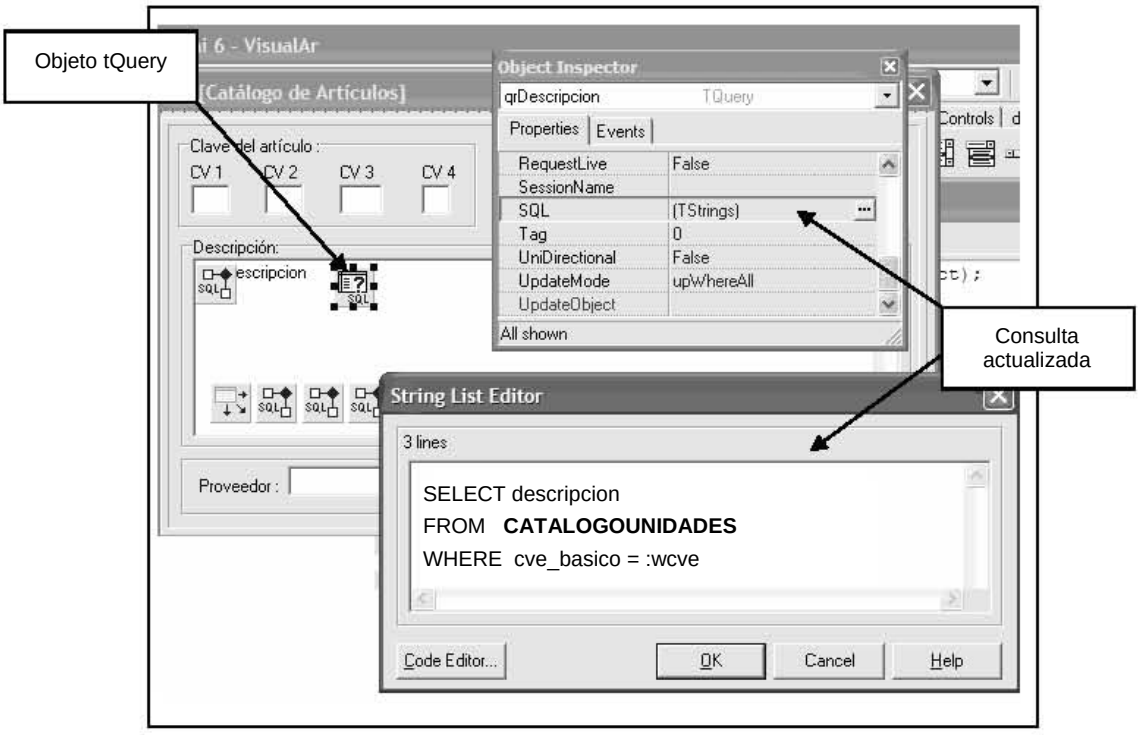


Diagrama 3.3.7

3.3.6. ***Alias para las columnas con funciones de agregación:*** En el caso de consultas con resultados agregados, es decir, que incluyen sentencias como COUNT, AVG, SUM, etc., es necesario asignar a la columna resultante un alias.

El alias a asignar será COL01, COL02, … a los campos calculados de un objeto TQuery y a las variables asociadas a estos.

Ejemplo:

```
SELECT SUM(CAMPO1) * 12, SUM(CAMPO3) – SUM(CAMPO4)
FROM   TABLAX
```

Debe quedar como:

```
SELECT SUM(CAMPO1) * 12  as col01, SUM(CAMPO3) –
SUM(CAMPO4) AS COL02
FROM   TABLAX
```

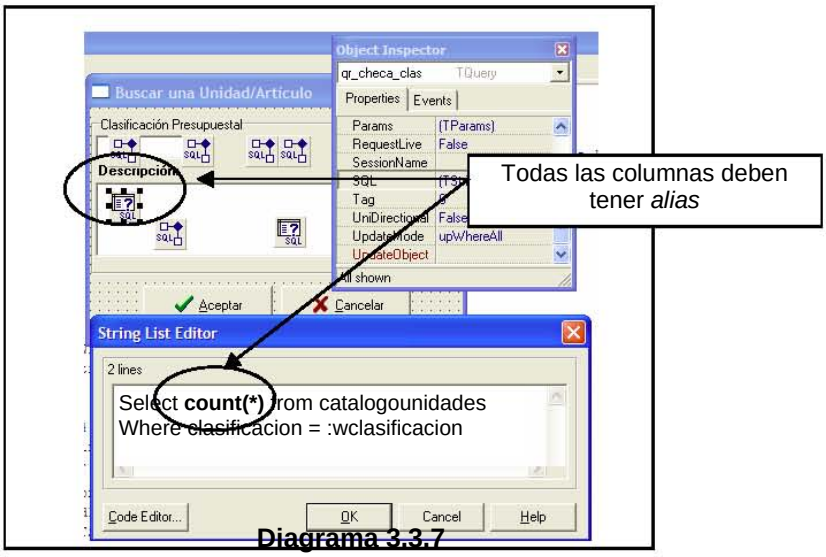


Diagrama 3.3.7

El código resultante queda como se muestra en el Diagrama 3.3.8.

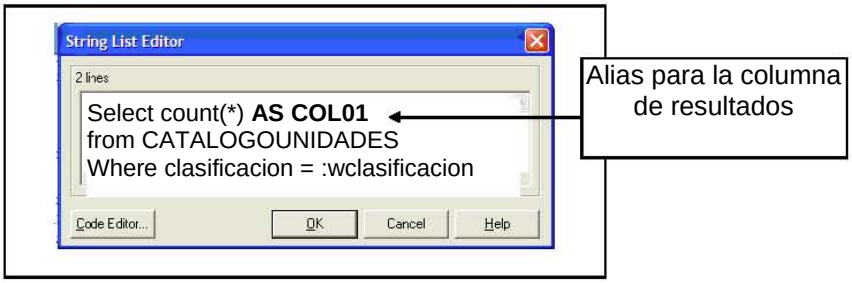


Diagrama 3.3.8

3.3.7. ***Manejo de Tablas Temporales:*** En el código de las aplicaciones Delphi 5 existen procesos para los que se crean tablas temporales de manera dinámica; por ejemplo cuando se está generando un condensado de información.
En Informix las tablas temporales existen mientras está activa la sesión que las creó y son eliminadas automáticamente cuando la sesión termina.

En Oracle 10gi existe un concepto similar, las *Global Temporary Table;* estas tablas se crean una sola vez en el script general de la base de datos; cuando un usuario inserta información en una tabla temporal dicha información únicamente puede ser vista por quien insertó esos datos. La estructura de la tabla es la misma para todos los usuarios.

Para cada tabla temporal que se identifique, se agregará una sentencia al script general de la base de datos como se muestra a continuación.

```
CREATE GLOBAL TEMPORARY TABLE nombre_tabla
(campo1… tipo,…..) ON COMMIT PRESERVE ROWS;
```

El procedimiento para reemplazar el uso de tablas temporales en el código de las aplicaciones en Delphi 5 es el siguiente (Diagrama 3.3.7):

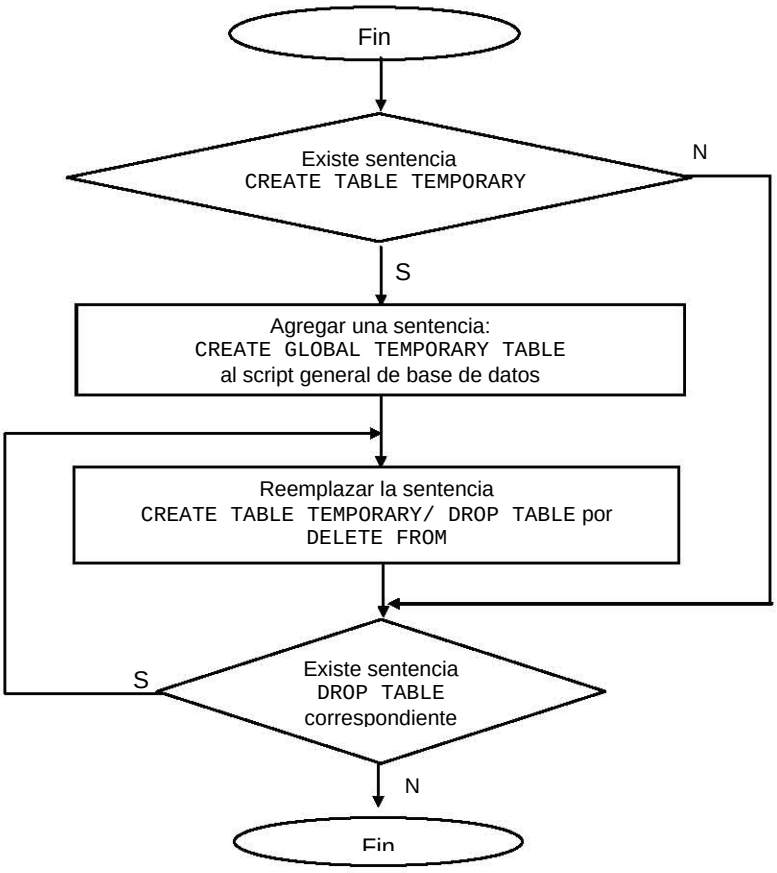


Diagrama 3.3.7

3.3.8. ***Queries dinámicos con parámetros:*** En el caso de localizar consultas dinámicas con parámetros, identificar si se está haciendo referencia a una columna de tipo carácter, en tal caso, se deberá emplear la función TRIM() a cada lado de la condición.

Además, se deberán aplicar los ajustes que se requiera de acuerdo a esta misma guía de actualizaciones como se muestra en el siguiente ejemplo.

```
select cve_alfanumerica, descripcion_producto
from  productos
where  cve_ alfanumerica =:param_ alfa
```

1.- El primer paso consiste en revisar la estructura de la tabla:

```
SQL> DESC productos

NAME                    NULL?     TYPE
----------------------  ------    ---------------
CVE_ALFANUMERICA        NOT NULL  CHAR(15)
DESCRIPCION_PRODUCTO              VARCHAR2(40)
```

2.- El campo cve_alfanumerica es de tipo carácter; esa columna es parte de la condición de búsqueda por lo tanto se deben eliminar los blancos de ambos lados de la condición de manera que el código resultante es el siguiente:

```
select cve_alfanumerica,
       descripcion_producto
from  PRODUCTOS
where  TRIM(cve_ alfanumerica) = TRIM(:param_ alfa)
```

3.3.9. ***Parámetros tipo fecha:*** Para parámetros tipo fecha los ajustes correspondientes son los siguientes:

1.- Identificar en el código de las consultas los parámetros tipo fecha esto se logra generalmente por el nombre del parámetro mismo y en algunas ocasiones por el nombre del campo al que va asociado.

2.- Por ejemplo, para un parámetro llamado param_fecha2, todas las referencias a dicha variable deberán completarse como sigue:

```
To_date(:param_fecha2, 'dd/mm/yyyy')
```

3.- Las líneas de código donde se asigna valor NOW (fecha actual) a los parámetros tipo fecha en un objeto tipo tQuery:

```
Query2.ParamByName('param_fecha2').AsDate := Now;
```

Se modificarán empleando la sentencia DateToStr(NOW) para convertir en string la fecha actual; además, del lado izquierdo de la sentencia el tipo de dato AsDate se sustituye por AsString para mantener la correspondencia de tipo de dato en la asignación.

Ejemplo:

Sentencia original:

```
Query2.ParamByName('param_fecha2').AsDate := Now;
```

Sentencia modificada:

```
Query2.ParamByName('param_fecha2').AsString :=
                                        DateToStr(Now);
```

3.3.10. ***Columnas sin "alias":*** Cuando en Informix no se define un alias específico para cada columna que regresa una consulta, o procedimiento almacenado o vista o cursor, Informix asigna un nombre por default a dichas columnas de resultado.

Los nombres por default asignados por Informix van numerados consecutivamente en el conjunto de resultados como:

```
(expression)
(expression)_1
(expression)_2
   . . .
```

En el código Delphi 5 de las aplicaciones el nombre default de columnas de resultados se cambiará consecutivamente como se muestra a continuación:

- La columna `(expression)` cambia a `col01`
- La columna `(expression)_1` cambia a `col02`
- La columna `(expression)_2` cambia a `col03`
- Y así sucesivamente

Por ejemplo, para el siguiente bloque de código el objeto *spArticulos* regresa un conjunto de 6 columnas sin nombre fijo.

```
. . .
if wAccion = 0 Then
  Begin
   edgpo.Text:=spArticulos.FieldByName('(expression)').AsString;
   edcv1.Text:=spArticulos.FieldByName('(expression)_1').AsString;
   edcv2.Text:=spArticulos.FieldByName('(expression)_2').AsString;
   edcv3.Text:=spArticulos.FieldByName('(expression)_3').AsString;
   edcv4.Text:=spArticulos.FieldByName('(expression)_4').AsString;
   edCantidad.Text     :=     FloatToStrF(spArticulos.FieldByName(
              '(expression)_6').AsFloat ,ffnumber,11,3);
. . .
```

Después del ajuste el código resultante es:

```
. . .
if wAccion = 0 Then
  Begin
   edgpo.Text:=spArticulos.FieldByName('col01').AsString;
   edcv1.Text:=spArticulos.FieldByName('col02').AsString;
   edcv2.Text:=spArticulos.FieldByName('col03').AsString;
   edcv3.Text:=spArticulos.FieldByName('col04').AsString;
   edcv4.Text:=spArticulos.FieldByName('col05').AsString;
   edCantidad.Text   :=   FloatToStrF(   spArticulos.FieldByName(
              'col06').AsFloat ,ffnumber ,11 ,3);
. . .
```

3.3.11. ***Identificadores de caracteres:*** En Informix las cadenas de caracteres se delimitan empleando comillas dobles (“), en Oracle el terminador de caracteres es una comilla sencilla (‘).

En el código Delphi es necesario reemplazar las comillas dobles por un apostrofo en todas las operaciones en que se involucran datos alfanuméricos. Se puede emplear chr(39) como equivalente del apóstrofe.

Ejemplo:

```
. . .
procedure TfrmCatalogo_visual_articulos.edCve2KeyDown(Sender:
TObject;
  var Key: Word; Shift: TShiftState);
Var
  Cadena : String;
begin
  If ((Key = VK_F1) and (edCv1.Text <> '')) Then
    Begin
      cadena:='Select descripcion From almacen_generico ' +
              'where cve1 ="' + edCve1.Text + '"' +
              'order by Cve2;';
. . .
```

Luego de reemplazar las comillas dobles el código resultante es:

```
. . .
procedure TfrmCatalogo_visual_articulos.edCve2KeyDown(Sender:
TObject;
  var Key: Word; Shift: TShiftState);
Var
  Cadena : String;
begin
  If ((Key = VK_F1) and (edCv1.Text <> '')) Then
    Begin
      cadena:='Select descripcion From almacen_generico ' +
              'where cve1 = ' + chr(39) + edCve1.Text + chr(39) +
              'order by Cve2;';
. . .
```

3.3.12. ***Reemplazar datos tipo Integer por Float:*** Es necesario reemplazar la declaración de objetos tipo TIntegerField por objetos TFloatField para evitar el siguiente mensaje de error.

```
Type mismatch for field ‘xxxx’, expecting: Integer
actual: Float’
```

El ajuste aplica para el código y los elementos de la forma.

Ejemplo:

```
...
  qrAux: TQuery;
  tbclasificacion: TStringField;
  tbConfigdias_mes: TIntegerField;
  tbConfigmes_consumo: TIntegerField;
  tbConfiganio_consumo: TIntegerField;
  ListBox1: TListBox;
  GroupBox1: TGroupBox;
...
```

Luego de aplicar los ajustes el resultado es el siguiente.

```
...
  qrAux: TQuery;
  tbclasificacion: TStringField;
  tbConfigdias_mes: TFloatField;
  tbConfigmes_consumo: TFloatField;
  tbConfiganio_consumo: TFloatField;
  ListBox1: TListBox;
  GroupBox1: TGroupBox;
...
```

Estos cambios deben aplicarse también a la vista de código:

```
. . .
  object qrAux: TQuery
    DatabaseName = 'Almacenes'
    Left = 104
    Top = 296
  End
  object tbConfiguracion: TTable
    DatabaseName = 'Almacenes'
    TableName = 'configura'
    Left = 384
    object tbClasificacion: TStringField
      FieldName = 'clasificacion'
      Required = True
      Size = 13
    end
    object tbConfigdias_mes: TFloatField
      FieldName = 'dias_mes'
    end
    object tbConfigmes_consumo: TFloatField
      FieldName = 'mes_consumo'
      Required = True
    end
    object tbConfiganio_consumo: TFloatField
      FieldName = 'anio_consumo'
      Required = True
    end
  end
. . .
```

CAPITULO IV

CASO DE USO

El proyecto de migración de la base de datos se ha dividido en dos fases.

- La fase inicial abarca la migración del esquema y la información de la base de datos.
- La segunda fase del proyecto corresponde a la migración de las aplicaciones en Delphi 5.

En el desarrollo de estas actividades se sigue el modelo de cascada.

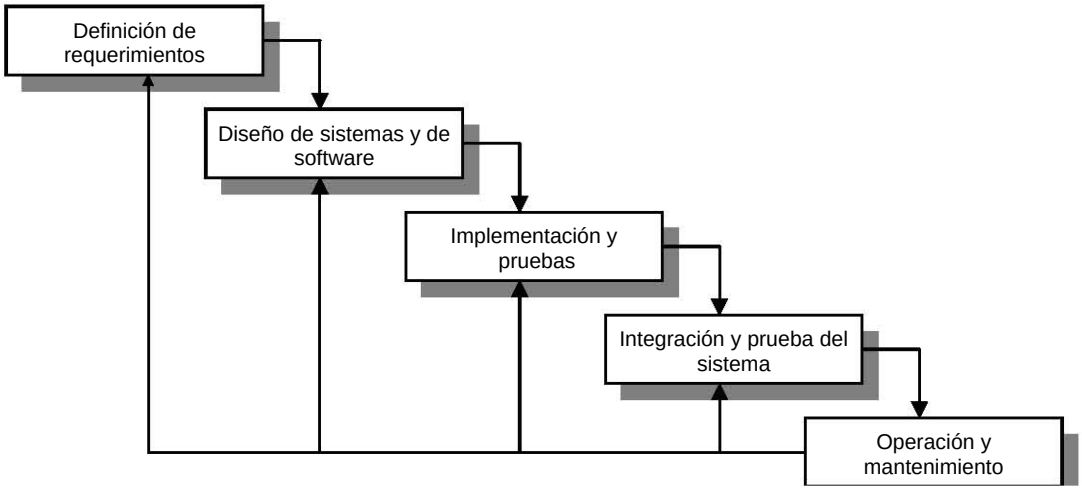


Diagrama 4.1

Durante la etapa de *Definición de requerimientos* se ha identificado la necesidad expresa del departamento de sistemas del área de Administración del Abasto; en la que se especifican los resultados esperados, las condiciones de tiempo y de operación en que ha de desarrollarse el trabajo y las características de los resultados esperados.

En la fase de *Diseño de sistemas y de software,* se analizaron las condiciones del desarrollo se establecieron planes de trabajo considerando las condiciones de tiempo que el Usuario definió.

Se analizaron las correspondencias inmediatas para las estructuras de datos y se evaluó la necesidad de emplear OMW como herramienta auxiliar en la migración de las estructuras de datos, código almacenado e información.

Para estimar la extensión de las modificaciones al código Delphi, se realizó una prueba con el catálogo de Artículos y se generó una guía preliminar de cambios.

La etapa de *Implementación y pruebas* se dividió en dos fases, como se ha mencionado antes. El desarrollo de la Migración de Aplicaciones en Delphi está ligado a los resultados obtenidos durante la Migración de la Base de datos, durante la migración de aplicaciones se pueden identificar errores o problemas con la definición de tablas, tipos de datos o código almacenado.

Estas circunstancias generan eventos de retroalimentación a la fase de migración de la base de datos, de manera que para que una pantalla o los resultados presentados por la misma sean marcados como válidos se requieren ajustes en la definición del modelo físico de la base de datos o ajustes al código almacenado o incluso pueden provocar ajustes en la manera de ejecutar una tarea o proceso en el código Delphi de la aplicación.

Este proceso se repetirá actualizando la guía de modificaciones al código Delphi o a la base de datos, según sea el caso, cada vez que se identifica una nueva diferencia.

El último tercio del calendario de actividades se dedica a las *Pruebas de integración* del sistema; para dichas pruebas se ejecutan los procesos del sistema sobre la base de datos Oracle. El Usuario no define casos de ejemplo ni procesos específicos, las validaciones que el Usuario realiza están basadas en la experiencia de uso de los procesos. Los resultados “sospechosos de falla” se comparan con los resultados de la misma operación sobre la base de datos original en Informix.

Las tareas de *Operación y mantenimiento* son desarrolladas por el personal de sistemas del Instituto.

4.1 Definición de la metodología

Existe una amplia documentación sobre guías de migración de motor de base de datos tanto en libros técnicos, manuales y programas tutoriales en la Internet.

Las consideraciones que determinan el esquema de trabajo que se describe al inicio de este capítulo se basan en el análisis de dos posibilidades principalmente. La primera de ellas siguiendo la metodología descrita por Oracle Migration Maps y la segunda opción describe las circunstancias por las que se ha definido separar en dos fases el proceso de migración que se describe en este capítulo.

Opción 1 Seguir el plan Oracle Migration Maps

Base de datos

En el caso particular de Oracle, existe una extensa metodología descrita en su Migration Map Process. Esta guía sugiere dividir el proceso en diferentes etapas: Definición, Análisis, Diseño, Migración, Transición, Producción. En cada etapa los integrantes del equipo desarrollan tareas específicas basados en roles de responsabilidad: Cliente, Administrador del Proyecto de Migración, Ingeniero de migración de base de datos, Ingeniero de migración de aplicaciones.

Señala puntualmente los documentos, reportes, minutas, planes de trabajo, notas técnicas, etc., que se generan durante cada una de las etapas en que se divide el guión y además proporciona ejemplos para el cálculo del esfuerzo y el tiempo que se ha de dedicar a cada tarea.

Define también la frecuencia de reuniones de trabajo entre el grupo de migración y el área usuaria marcando para cada tipo de reunión una relación de documentos a desarrollar asignados en algunos casos al usuario y en otras ocasiones al grupo de administración del proceso.

El plan de trabajo para la migración generado empleando esta guía define además que la herramienta básica para el desarrollo será OMW.

Aplicaciones

Migración manual del código Delphi aplicación por aplicación después de haber creado y cargado con información la base de datos en Oracle.

Del análisis de requerimientos adicionales que presenta la aplicación de esta metodología, y considerando las condiciones iniciales para la migración, esta guía de trabajo presenta ventajas y desventajas que se listan a continuación.

Base de datos

Ventajas	Desventajas
➢ Requiere la integración de personal dedicado exclusivamente a la administración de las tareas y cumplimiento de metas. ➢ Divide las tareas en roles específicos de responsabilidad. • Cliente • Administrador del Proyecto de Migración • Ingeniero de migración de base de datos • Ingeniero de migRación de aplicaciones ➢ La relación de materiales de análisis por desarrollar en cada etapa del proyecto están definidos específicamente, por ejemplo: Reporte de análisis, Instrucciones de instalación de aplicaciones, Referencia de migración de aplicaciones, Requerimientos de capacitación al Usuario, Diseño de la migración de datos, Diseño del reporte de análisis de migración, Plan de migración de datos, Reporte de migración de datos, etc.	➢ Se requiere tiempo dedicado únicamente a la aplicación de la metodología en el levantamiento de las especificaciones del alcance. ➢ Se requiere personal que cumpla con la definición de los roles y perfiles de la metodología. ➢ El número de especialistas crecerá según el tamaño de la base de datos y el número y complejidad de las aplicaciones asociadas. ➢ Se requiere tiempo del personal del área de sistemas exclusivamente a resolver las solicitudes de información, materiales de ejemplo, reportes, etc., presentadas por el grupo de administración de la migración. ➢ Cuando no exista documentación de alguno de los materiales solicitados al Usuario, se requiere dedicar tiempo a obtener el material en referencia.

Aplicaciones

Ventajas	Desventajas
➢ Migración manual del código Delphi aplicación por aplicación.	➢ A consecuencia de la migración manual del código Delphi se pueden identificar cambios adicionales a la definición de la base de datos, por ejemplo, eliminar el uso de tablas temporales.

Opción 2 Dividir en dos fases la migración

Considerando las restricciones de tiempo definidas por la Institución, no es posible seguir el esquema de trabajo propuesto en el plan del Oracle Migration Maps.

Las restricciones de tiempo han sido marcadas por la Institución y están relacionados con metas de otras áreas por lo que no es posible modificar las condiciones iniciales.

Tiempo
La posibilidad que tiene el grupo de migración para definir el plan de trabajo se limita a la distribución de las tareas a lo largo de las nueve semanas con que se cuenta para la conclusión del proyecto.

Herramientas
Se cuenta con una licencia de OMW.

Personal
El equipo está integrado por cuatro personas, con experiencia en desarrollo en Informix y Delphi 5 y conocimientos intermedios de programación en Oracle, serán los encargados de migrar las aplicaciones Delphi.
Al grupo se suma un administrador de base de datos con conocimientos de Oracle pero sin conocimientos de Informix.

Documentación
Es escasa la documentación del sistema, al arranque del proyecto se tienen los manuales de usuario y de instalación en versiones no actualizadas.
No se cuenta con diagrama entidad/relación actualizado y aún cuando es posible recuperarlo directamente de Informix, las jerarquías y dependencia entre tablas no esta asegurada mediante *constraints* sino directamente por el código Delphi o código almacenado.

Usuarios
Al momento del arranque de este proceso no es posible contar con atención de los usuarios o responsables de las aplicaciones y procesos del Sistema de Administración del Abasto, de manera que las oportunidades para revisión de dudas y justificación de modificaciones son escasas y limitadas en tiempo.
Es posible modificar la programación de procesos para mantener la operación de las aplicaciones siempre y cuando no haya cambios visibles al Usuario.

En este entorno, no es posible aplicar la metodología general propuesta en el Oracle Migration Maps aunque sí es posible emplear OMW como auxiliar en los procesos de migración de la información y estructuras de datos como se describe a continuación.

4.2 Migración del esquema de datos

En la Tabla 4.1.1 se muestra el plan de trabajo definido para la migración de la base de datos.

Plan de trabajo

Actividad	Semana								
	1	2	3	4	5	6	7	8	9
1. Migración del esquema de datos.									
1.1) Inventario inicial.									
2. Ajustes de equivalencias y correspondencias.									
3. Migración de información.									
4. Ajustes y correcciones a código almacenado.									
5. Actualización de aplicaciones Delphi.									

Tabla 4.1.1

El Diagrama 4.1.1 ilustra el procedimiento general para la migración de la base de datos.

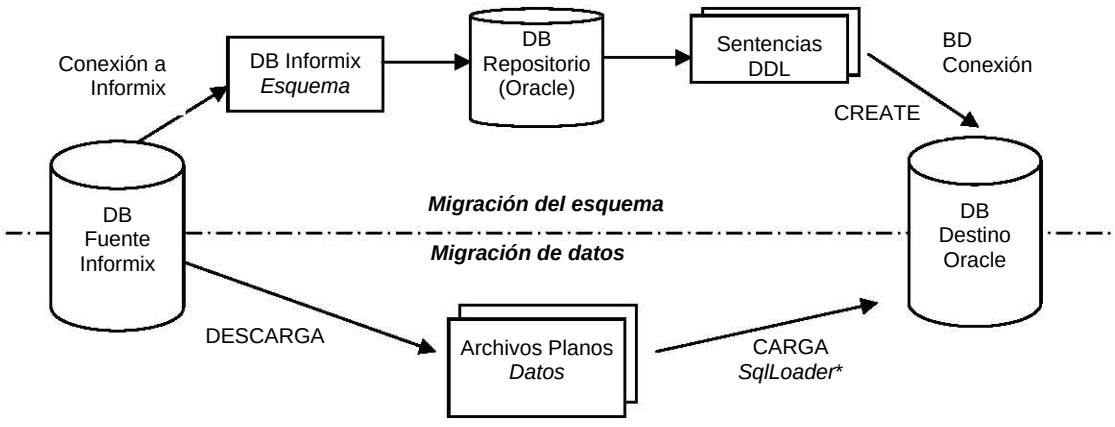


Diagrama 4.1.1

Proceso:

- Instalar OMW en una configuración estándar (Capitulo III).
- Obtener el inventario inicial de la base de datos *almacenes*.
- Ejecutar la conversión estándar de OMW para transformar el esquema original en Informix a la nueva versión en Oracle 10gi.
- Crear la base de datos *almacenes* en Oracle 10gi con el script generado en OMW.
- Obtener el inventario final de objetos de la base de datos *almacenes* en Oracle 10gi.
- Comparar los inventarios inicial y final.
- Identificar y corregir diferencias o problemas originados por tipo de datos.

Al finalizar el proceso descrito se cuenta ya con una base de datos preliminar, misma que generalmente sufrirá actualizaciones tanto en la definición de los atributos en la versión Oracle como en el código de los procedimientos almacenados, vistas, triggers.

Los ajustes son resultado de las pruebas que se realizan sobre las aplicaciones en Delphi 5 una vez que se inicia la actualización del código de las mismas.

Una vez instalado OMW en una configuración estándar, estableceremos un repositorio inicial para la información de la base de datos original.

1 Inventario inicial

El inventario inicial de la base de datos almacenes en Informix es el siguiente.

Tipo de objeto	*Descripción*	*Núm. de Objetos*
1. TABLE	Tablas	2441
2. INDEX	Indices	750
3. VIEW	Vistas	12
4. TRIGGER	Triggers	112
5. PROCEDURE	Procedimientos	3824
6. SERIAL FIELDS	Tablas con campos tipo Serial	41
7. LOB	Objetos tipo LOB	80

Tabla 4.1.2

2 Inventario final en la base de datos migrada a Oracle

Empleando OMW para migrar las estructuras de datos y el código almacenado el inventario final es el siguiente.

Tipo de objeto	*Descripción*	*Número de Objetos*
1. TABLE	Tablas	2441
2. INDEX	Indices	750
3. VIEW	Vistas	12
4. TRIGGER	Triggers	153
5. PROCEDURE	Procedimientos	3523
6. FUNCTION	Funciones	301
7. SEQUENCE	Secuencias	41
8. LOB	Objetos tipo LOB	80
9. TYPE	Tipos de dato adicionales	1

Tabla 4.1.3

En Oracle, es posible obtener el inventario de objetos con la siguiente consulta:

```
SELECT object_type, count(*)
FROM   all_objects
WHERE owner = 'INFORMIX'
GROUP BY object_type
```

3 Análisis de resultados

Tipo de objeto	*Inventario Informix (a)*	*Inventario Oracle (b)*
TABLE	2441	2441
INDEX	750	750
VIEW	12	12
TRIGGER	112	153
PROCEDURE	3824	3523
FUNCTION		301
SEQUENCE		41
LOB	80	80
TYPE		1
SERIAL FIELD	41	
	3	2

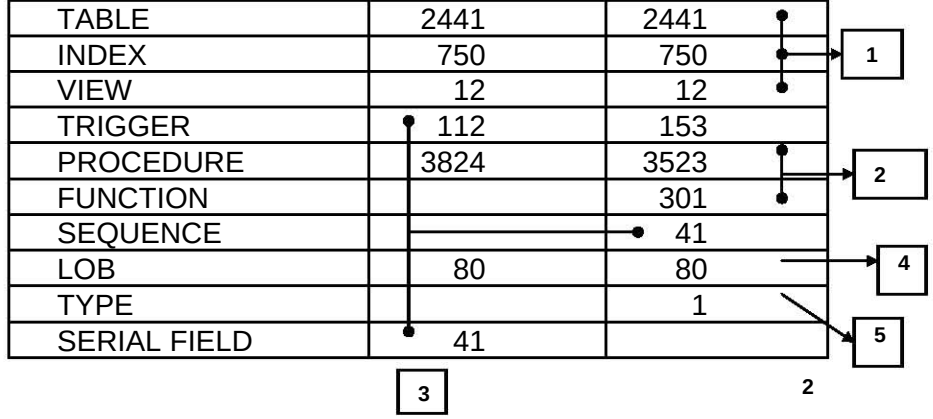


Tabla 4.1.4

Caso 1

El número de estructuras tipo TABLE, INDEX y VIEW se mantiene constante en la versión de Oracle de la base de datos.

Caso 2

El número de objetos tipo PROCEDURE disminuye en 301, pero se crean 301 nuevos objetos, tipo FUNCTION para complementar el número de objetos tipo PROCEDURE originales dado que la funcionalidad de los procedimientos corresponde en Oracle a un objeto tipo FUNCTION. Esto es, los procedimientos que retornaban un valor se convierten en Oracle en objetos tipo FUNCTION.

Caso 3

Los objetos tipo SERIAL FIELD han sido reemplazados por 41 objetos tipo SECUENCE y 41 objetos tipo TRIGGER para mantener la funcionalidad de valores auto incrementales en los atributos asociados.

Caso 4

El objeto tipo TYPE corresponde a un nuevo tipo de datos generado para complementar la funcionalidad del llamado a objetos tipo FUNCTION.

Caso 5

Los objetos tipo LOB han sido migrados en igual número en el correspondiente CLOB de Oracle.

4.3 Migración de datos

Para migrar los datos entre la base de datos fuente en Informix a la base destino en Oracle se requiere cubrir dos etapas.

1 Extraer la información de la base de datos de informix a archivos de texto plano empleando la utilería `unload` de Informix.

1.1 Los archivos de información, uno por cada tabla, se pueden generar desde OMW, desde el menú Object la opción Generate SQL* Loader Scripts.

Otro procedimiento para obtener los archivos planos es ejecutar para cada tabla en la base de datos original en Informix la línea siguiente:

```
unload to <TABLA.dat>
select * from TABLA
```

Ejemplo desde una sesión en informix en el Diagrama 4.2.1.

```
Telnet nxst
MODIFY: ESC    = Done editing      CTRL-A = Typeover/Ins
        CTRL-X = Delete character  CTRL-D = Delete rest o
--------------- almacenes@sistemas_abasto ------ Press CT

Unload to “articulos.dat”
Select * from articulos



9 row(s) unloaded.
```

Diagrama 4.2.1

En el ejemplo, los datos de la tabla Articulos son descargados a un archivo `articulos.dat` separados por pipes.

Para cada tabla obtendremos un archivo con extensión .dat, separado por *pipes* (|) como se muestra a continuación en el caso de la tabla ARTICULOS.

ARTICULOS.DAT

```
010|000|0022|01|CASEINATO DE CALCIO POLVO. CADA 100 G CONTIENEN: PROTEINAS
(*) 86.0 A 90.0 G,  GRASAS 0.0 A 2.0 G MINERALES 3.8 A-6.0 G HUMEDAD 4.0 A
5.5 G (*) LAS PROTEINAS  EQUIVALEN A 13.5-14.1 G DE NITROGENO. LATA CON 100
G .|CASEINATO DE CALCIO POLVO. -CADA 100 G CONTIENEN: PROTEINAS (*) 86.0 A
90.0 G GRASAS0.0 A 2.0 G MINERALES 3.8 A-6.0 G HUMEDAD 4.0 A 5.5 G  -(*) LAS
PROTEINAS  EQUIVALEN-A  13.5-14.1     G  DE  NITROGENO.LATA  CON  100  G
.||||LTA|100.0|GRO|10/21/1987||11|11/16/1988|D.
OFICIAL|071|00|00|00|00|00|0|00|0|0|0301|11100|0000000000|0012|00|
010|000|0022|02|CASEINATO DE CALCIO POLVO. CADA 100 G CONTIENEN: PROTEINAS
(*) 86.0 A 90.0 G  GRASAS 0.0 A 2.0 G MINERALES 3.8 A-6.0 G HUMEDAD 4.0 A
5.5 G (*) LAS PROTEINAS  EQUIVALEN A 13.5-14.1 G DE NITROGENO. LATA CON 100
G.|CASEINATO DE CALCIO POLVO. -CADA 100 G CONTIENEN: PROTEINAS (*) 86.0 A
90.0 G GRASAS0.0 A 2.0 G MINERALES 3.8 A-6.0 G HUMEDAD 4.0 A 5.5 G  -(*) LAS
PROTEINAS  EQUIVALEN-A  13.5-14.1     G  DE  NITROGENO.LATA  CON  100  G
.||||LTA|1.0|PZA|04/26/1988|000|11|11/16/1988|D.
OFICIAL|071|12|11|00|00|10|00|0|00|0|0|0301|00000|0101000100|0000|00|
                                    . . .
010|000|0022|11|CASEINATO DE CALCIO. POLVO. CADA 100 G CONTIENEN: PROTEINAS
86.0 A 90.0 G. GRASAS 0.0 A 2.0 G. MINERALES 3.8 A 6.0 G. HUMEDAD 0.0 A 6.2
G.  LATA  CON  100  G.|CASEINATO  DE  CALCIO.  POLVO.CADA  100  G  CONTIENEN:
PROTEINAS 86.0 A 90.0 G. GRASAS --0.0 A 2.0 G. MINERALES 3.8 A6.0 G. HUMEDAD
0.0           A           6.2           G.-LATA           CON           100
G.||||LTA|100.0|GRO|10/30/1991||13|01/15/1998|325|011|01|03|55|00|0|00|R|00|
0|0|0301|00000|1212000000|1212|00|
```

DIAGRAMA 4.2.2

El procedimiento manual se puede sustituir por un procedimiento *batch* o bien por el procedimiento en línea que OMW proporciona.

A continuación se listan a manera de ejemplo las rutinas en Unix que pueden emplearse como auxiliares en el proceso.

Variables de ambiente

```
#!/bin/bash
# filename: conversion_config.bsh
# Variables de configuración para la conversión
# Lista de tables a convertir

TABLE_LIST='
A_table_name
B_table_name'

AWK="/usr/bin/gawk"
SCHEMA_FILE="schema.sql"
TABLE_DATA_PATH="table_data" # Directory for table data.
mkdir $TABLE_DATA_PATH
TABLE_CONTROL_FILE_PATH="control_file" # Oracle sqlldr control files.
mkdir $TABLE_CONTROL_FILE_PATH
LOAD_LOG_PATH="load_log" # Table loading logs.
mkdir $LOAD_LOG_PATH
```

Descarga de datos

```
#!/bin/bash
# filename: get_data.bsh
# descarga de tablas de Informix tables en archivos .dat
# Nota: las variables de ambiente deben estar registradas

. ./I-O_config.bsh

# Informix unload.
unload_tables () {

for table in ${TABLE_LIST}
do
      dbaccess database_name << EOF
      unload to "${TABLE_DATA_PATH}/${table}/.txt"
      select * from $table;
      EOF
done
}
```

2 Una vez que se han generado los archivos de datos, uno para cada tabla original, el segundo paso del procedimiento es Cargar la información de los archivos planos a las tablas correspondientes en la base de datos destino en Oracle 10gi empleando.

El proceso de carga se realiza empleando la utilería SqlLoader de Oracle.

Para emplear SqlLoader se requiere:

- Un archivo de datos para cada tabla destino, separada la información por pipes (|).
- Un archivo de control (extensión ctl) en el que se especifica el nombre del archivo de datos, el nombre y lista de campos de la tabla destino y el delimitador de campos.

Como resultado de la carga se puede obtener un archivo de Log con los resultados del proceso, un archivo de errores con los registros que presentan error durante el proceso de carga y la descripción del error generado.

2.1 *Archivo de datos*

El orden en que la información se presenta en el archivo de intercambio corresponde a la secuencia de los campos en la tabla original en informix.

Por ejemplo, para la tabla artículos, en Informix la estructura es la que se muestra a continuación.

```
create table "informix".articulos   (
  cv1 char(3) not null constraint "informix".no999_9993,
  cv2 char(3) not null constraint "informix".no999_9994,
  cv3 char(4) not null constraint "informix".no999_9995,
  cv4 char(2) not null constraint "informix".no999_9996,
  descripc   char(254)   not null
                    constraint "informix".no999_9998,
  precio     money(16,2) not null
                    constraint "informix".no999_9999,
  clave_unica   char(3) not null
                     constraint "informix".no999_1001,
  cve_tipo      char(3) not null
                     constraint "informix".no999_1111,
  cve_basicos   char(2) not null
                     constraint "informix".no999_2222,
  cantidad      decimal(11,3),
  cve_part      char(4) not null
                    constraint "informix".no999_3333,
  fecha_rec date not null constraint "informix".no999_4444,
  inventariable      char(1),
  nivel_compra       char(1),
  linea              char(3),
  fecha_incl         date,
  comision           char(2),
  cve_estado         char(2),
  fecha_dto_fte      date,
  num_oficina        char(40),
  cve_incl           char(2),
  uso_hospitalario   char(1),
  tipo_alta          char(1),
  cve_tipo_adq       char(1),
  cve_basico_estdr   char(2),
  cve_control        char(1)     default '0',
  dias_a_caducar     integer
                    constraint "informix".no111_1111,
  obsoletos          char(1)
                     constraint "informix".no111_1113,
  ftes_alter_abast   char(1),
  fec_ult_ent        date,
  fec_ult_sal        date,
  primary key (cv1,cv2,cv3,cv4)
                    constraint "informix".pko_articulos
  );
```

DIAGRAMA 4.2.1

2.2 *Archivo de control*

A continuación se muestra el archivo de control para la tabla articulos.

ARTICULOS.CTL

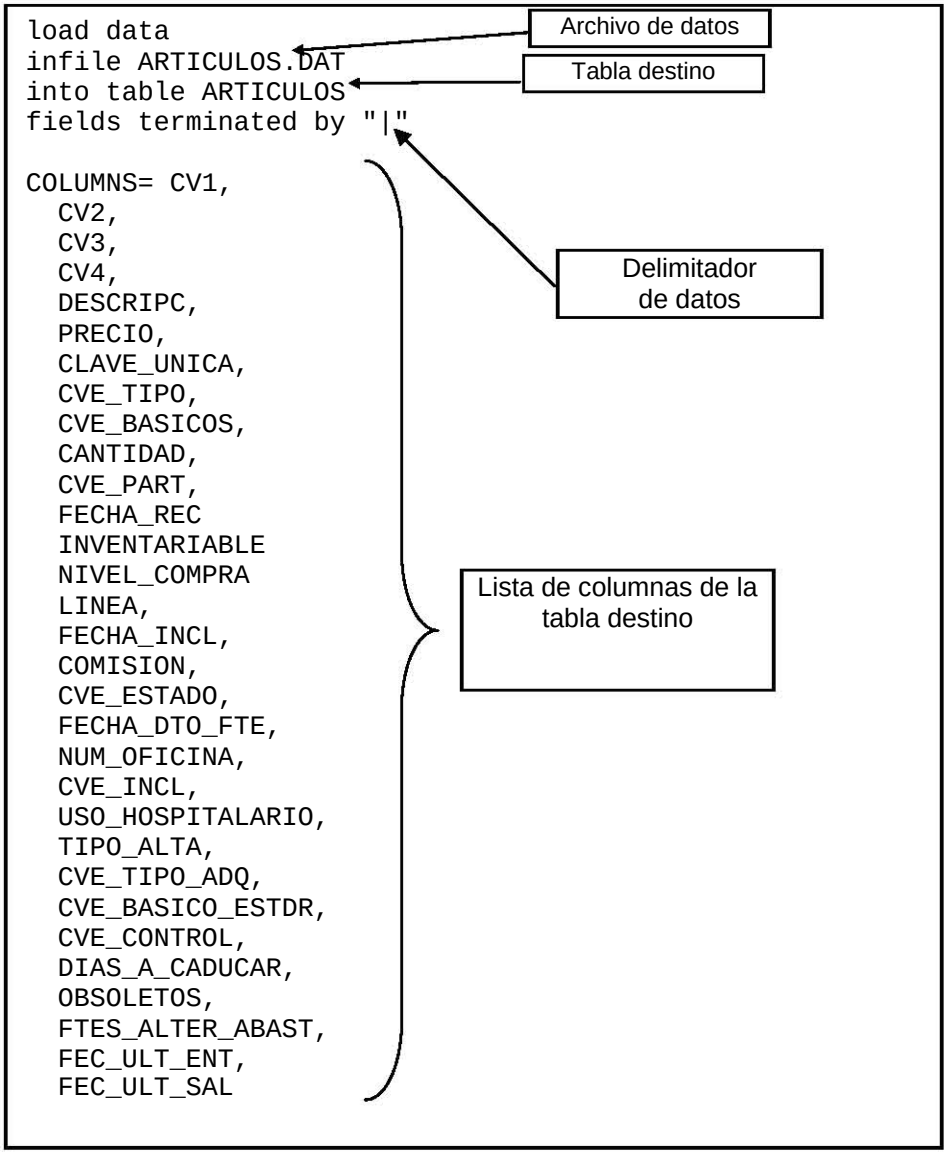


DIAGRAMA 4.2.2

Para obtener los archivos de control (.ctl) de todas las tablas se puede emplear una secuencia de instrucciones como la siguiente que accesa la base de datos en Informix.

```
# Creación de los archivos de control de Oracle
make_control_file () {
for table in ${TABLE_LIST}
do
cat << EOF > $TABLE_CONTROL_FILE_PATH/$table.ctl

load data
infile '${TABLE_DATA_PATH}/${table}/.DAT'
into table $table
fields terminated by "|"
EOF

  echo '(' >> $TABLE_CONTROL_FILE_PATH/$table.ctl
  COLUMNS=$(get_table_columns)
  echo $COLUMNS |
  ${AWK} '{ gsub (" ",","); print}' >>
$TABLE_CONTROL_FILE_PATH/$table.ctl
  echo ')' >> $TABLE_CONTROL_FILE_PATH/$table.ctl
done
}
```

Donde :

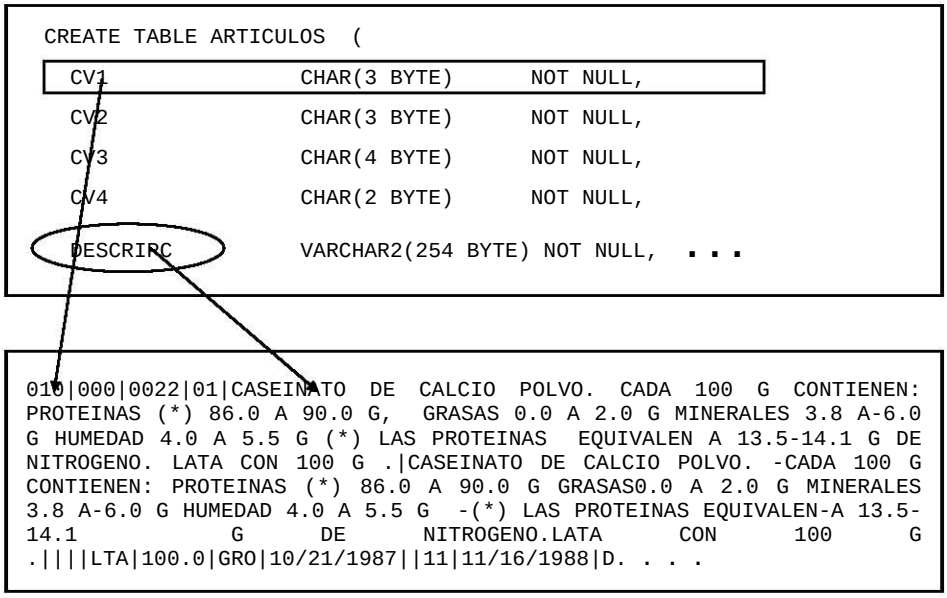
```
get_table_columns () {

dbaccess database_name << EOF | grep -v '^$'
   output to pipe "cat" without headings
   select colname from syscolumns, systables
   where
   systables.tabname = "$table"
   and systables.tabid = syscolumns.tabid;
EOF
}
```

A manera de ejemplo veamos el caso del archivo de control para la tabla ARTICULOS donde se señala que la información de cada campo está delimitada por un carácter especial, en este caso un pipe (|).

```
CREATE TABLE ARTICULOS  (
  CV1               CHAR(3 BYTE)      NOT NULL,
  CV2               CHAR(3 BYTE)      NOT NULL,
  CV3               CHAR(4 BYTE)      NOT NULL,
  CV4               CHAR(2 BYTE)      NOT NULL,
  DESCRIPC          VARCHAR2(254 BYTE) NOT NULL,
  PRECIO            NUMBER(16,2)      NOT NULL,
  CLAVE_UNICA       CHAR(3 BYTE)      NOT NULL,
  CVE_TIPO          CHAR(3 BYTE)      NOT NULL,
  CVE_BASICOS       CHAR(2 BYTE)      NOT NULL,
  CANTIDAD          NUMBER(11,3),
 CVE_PART          CHAR(4 BYTE)       NOT NULL,
  FECHA_REC         DATE              NOT NULL,
  INVENTARIABLE     CHAR(1 BYTE,
  NIVEL_COMPRA      CHAR(1 BYTE),
  LINEA             CHAR(3 BYTE),
  FECHA_INCL        DATE,
  COMISION          CHAR(3 BYTE),
  CVE_ESTADO        CHAR(2 BYTE),
  FECHA_DTO_FTE     DATE,
  NUM_OFICINA       CHAR(40 BYTE),
  CVE_INCL          CHAR(2 BYTE),
  USO_HOSPITALARIO  CHAR(1 BYTE),
  TIPO_ALTA         CHAR(1 BYTE),
  CVE_TIPO_ADQ      CHAR(1 BYTE),
  CVE_BASICO_ESTDR  CHAR(2 BYTE),
  CVE_CONTROL       CHAR(1 BYTE)  DEFAULT '0',
  DIAS_A_CADUCAR  NUMBER(10) DEFAULT 0 NOT NULL,
  OBSOLETOS   CHAR(1 BYTE) DEFAULT '0' NOT NULL,
  FTES_ALTER_ABAST  CHAR(1 BYTE),
  FEC_ULT_ENT       DATE,
  FEC_ULT_SAL       DATE )
```

DIAGRAMA 4.2.3

La correspondencia entre la tabla y los datos en el archivo .DAT.



2.3 *Secuencia de carga*

La secuencia de carga para tablas que no son catálogos, fue definida por el personal del área de sistemas, basados en la experiencia y conocimiento de la base de datos original, esto es debido a que no se tiene un diagrama entidad/relación actualizado que describa las dependencias entre estructuras de la base de datos.

Adicionalmente, se tiene un alto porcentaje de tablas para las que no existen *constraints* que permitan identificar las jerarquías entre estructuras. Para la mayoría de estos casos la integridad referencial se asegura por pantalla o por los procedimientos almacenados que se emplean para la actualización de la información en la base de datos.

2.4 *Análisis de resultados*

Durante los procesos de carga masiva se encontraron algunos reportes de error en el tipo de datos. Para cada caso se analizó el tipo de error y se implementó la solución correspondiente tanto en el script de creación de la base de datos como en la definición de los archivos de control correspondientes, de manera que el proceso de creación y carga de datos se repitió en varias ocasiones.

La secuencia de carga de las tablas se verificó también analizando los mensajes de error generados por SQL Loader.

La carga de información se realizó desactivando los triggers, mismos que fueron activados al terminar los procesos de carga, esta consideración obedece a que los triggers asociados a la operación de INSERT no debían realizar los procesos asociados durante la carga masiva pues esos *insert's* no obedecen a la operación normal de las aplicaciones.

También fue necesario verificar los valores iniciales de las secuencias que complementan el reemplazo de los atributos *Serial Field.*

El ciclo de ajuste a las características del tipo de dato de algunas columnas se repitió algunas veces más como resultado de solicitudes específicas que se originaban durante la migración del código fuente.

4.4 Migración de aplicaciones

1 Inventario Inicial de aplicaciones

El sistema de Sistema de Administración del Abasto está organizado como se muestra en la Diagrama 4.3.1.

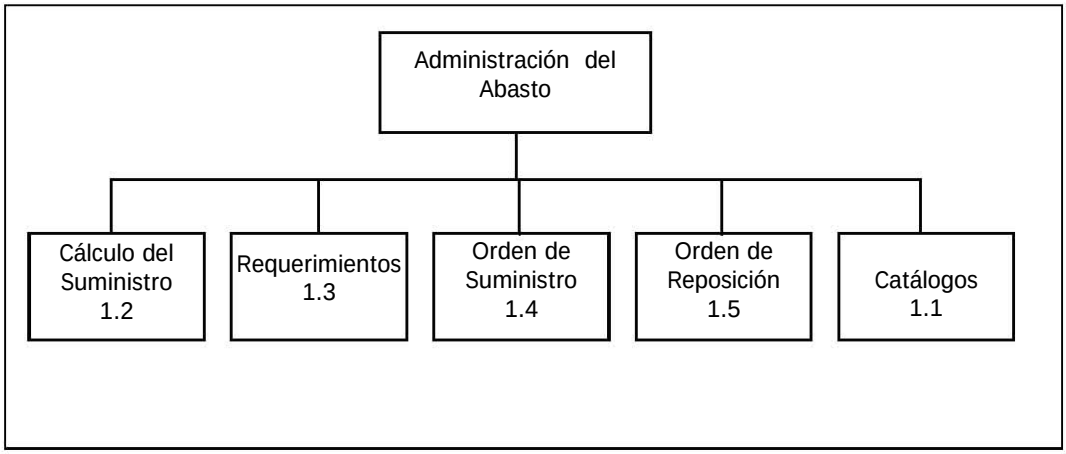


Diagrama 4.3.1.

El siguiente plan de trabajo muestra el tiempo estimado para la actualización de las aplicaciones que integran el Sistema de Administración del Abasto considerando un equipo de trabajo de 4 personas con experiencia en el desarrollo de aplicaciones en Delphi 5 y conocimientos intermedios de programación en Oracle.

Proyectos Delphi	Semana								
	1	2	3	4	5	6	7	8	9
1 Catálogos									
1.1 Artículos									
1.2 Unidades									
1.3 Clasificación-Servicios									
2 Cálculo de Suministro									
2.1 Consumo Promedio Mensual									
2.2 Frecuencias									
2.3 Máximos y Mínimos									
3 Requerimientos									
4 Orden de Suministro									
5 Orden de Reposición									
6 Pruebas de integración									

1 Pruebas

A continuación se muestra el procedimiento de validación de los ajustes aplicados al código de las aplicaciones en Delphi, será necesario repetir el proceso de validación y ajustes hasta obtener los resultados requeridos para cada una de las aplicaciones (Diagrama 4.3.1).

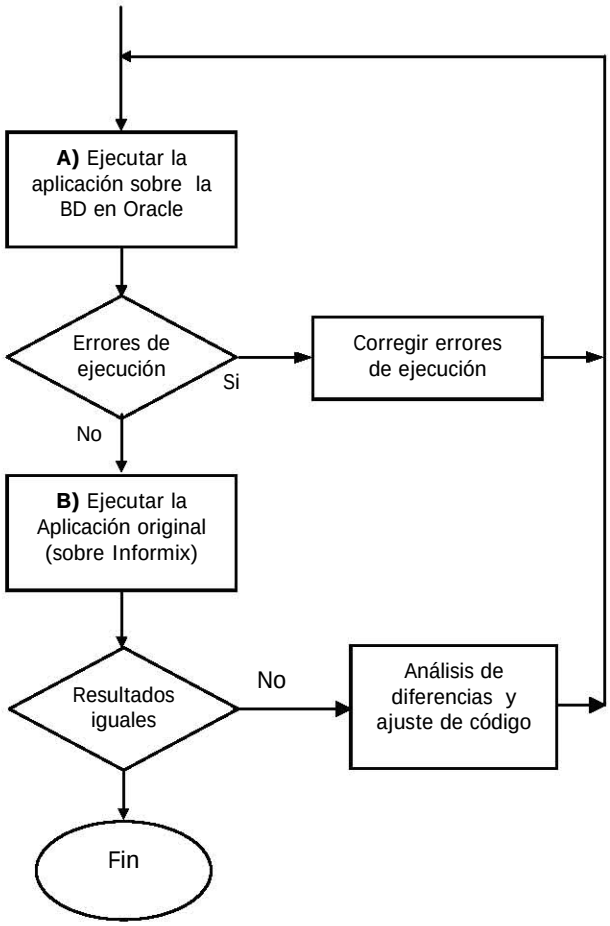


Diagrama 4.3.1

Para probar los ajustes, es necesario contar con una conexión ODBC a la base de datos generada en Oracle, configurada como se muestra a continuación.

Diagrama 4.3.2.a

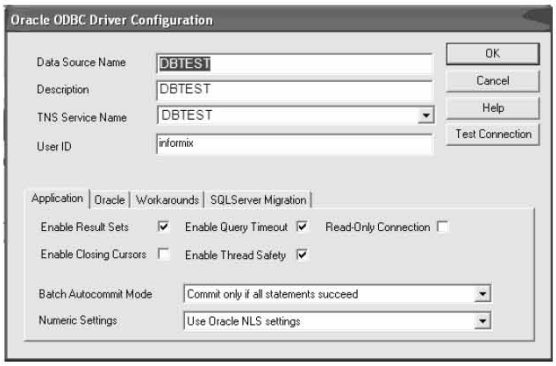


Diagrama 4.3.2.b

Complementando el uso de la conexión ODBC se requiere una conexión BDE configurada como se muestra a continuación.

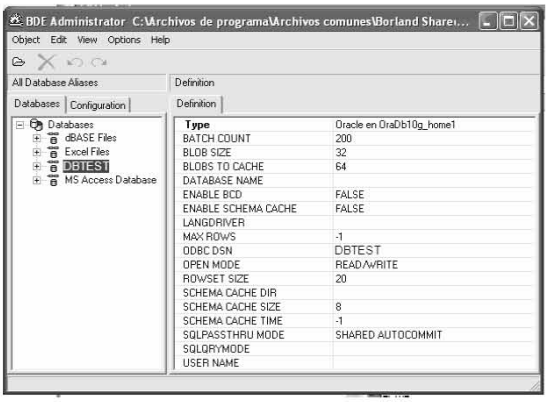
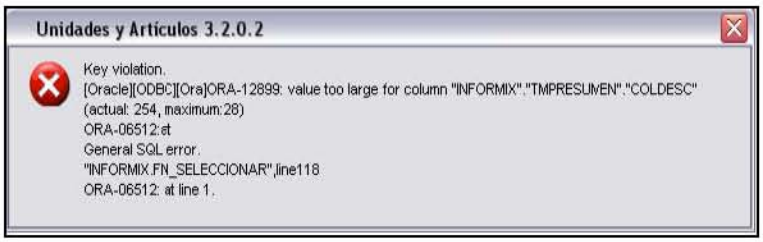


Diagrama 4.3.2.c

2 Análisis de resultados

A continuación se listan las circunstancias de error que se identificaron durante la etapa de ajustes a la codificación y pruebas de las aplicaciones señalando además la acción correctiva correspondiente.

… value too large for column …



Observaciones

En la línea número 118 del procedimiento o función FN_SELECCCIONAR, se realiza una operación de asignación de un valor o variable de tipo carácter de longitud 254 caracteres a la columna COLDESC de la tabla TMPRESUMEN propiedad del Usuario INFORMIX; dicha columna tiene una extensión máxima de 28 caracteres.

Informix no marca errores en este tipo de asignación, al asignar valores, registra únicamente los primeros 28 caracteres de la cadena en la columna COLDESC y no se producirá un error.

Sugerencia

Modificar la sentencia que asigna la cadena de 254 caracteres por:

```
. . .
   SUBSTR(variable_coldesc, 1, 28).
. . .
```

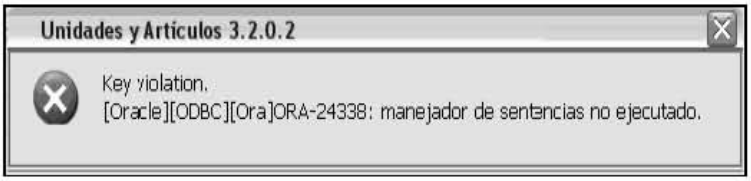
Este tipo de diferencias se presenta únicamente al momento de ejecutar la sentencia, en Informix no se presenta el error debido a que el dato registrado en la tabla se corta automáticamente a la longitud definida para la tabla como se muestra a continuación.

```
Telnet nxst
MODIFY: ESC    = Done editing      CTRL-A = Typeover/Ins
        CTRL-X = Delete character  CTRL-D = Delete rest o
--------------- almacenes@sistemas_abasto ------ Press CT

Drop table xx;
Create table xx (col1 char(5));
Insert into xx values ('12345');
Insert into xx values ('1234567890');
Select length(col1),  col1, length(trim(col1)) from xx;
```

El resultado es el siguiente, note la longitud de los dos datos insertados:

```
 Telnet nxst
SQL: _  New  Run  Modify Use-editor Output Cho
Run the current SQL statements.
--------------- almacenes@sistemas_abasto ----

(expression)    col1   (expression)
          5     12345              5
          5     12345              5

2 row(s) retrieved.
```

… manejador de sentencias no ejecutado …



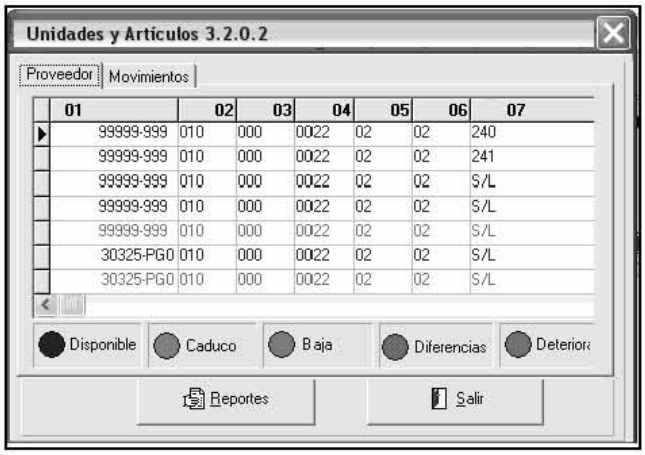
Observaciones

Se hace referencia a un objeto en la base de datos que no está activo o que está inválido.

Sugerencia

Verificar que el objeto de base de datos que se accesa existe y está activo.
Prevenir el acceso a objetos de base de datos no activos o no válidos.

… tablas de datos sin título



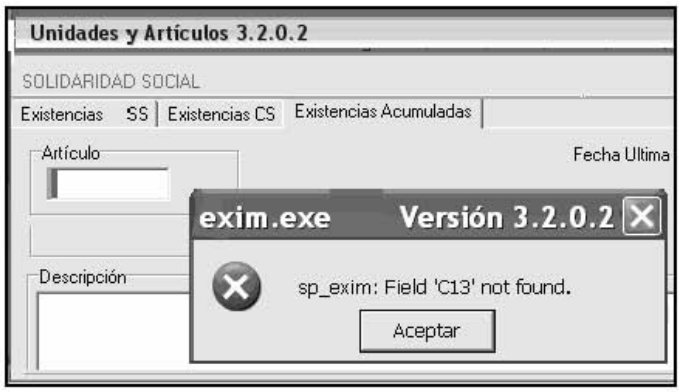
Observaciones

Los datos en la pantalla proceden de una consulta de columnas agregadas o bien, de un procedimiento almacenado, al que no se asignó el alias correspondiente a cada columna.

Sugerencia

Actualizar el procedimiento almacenado, o en su caso, la consulta asignando el alias respectivo a cada columna de regreso.

… field “XXX” not found.



Observaciones

El objeto sp_exim, que en este caso es un objeto tipo TStoredProcedure, hace referencia a la columna C13, dicha columna no es parte del conjunto de resultados que regresa el stored procedure de la base de datos.

En el ejemplo la columna C13 no existe en el conjunto de datos de retorno.

Sugerencia

Verificar los nombres de columnas que el stored procedure está regresando a la llamada desde la pantalla.

… Couldn't perform the edit because another user changed the record

.

Observaciones

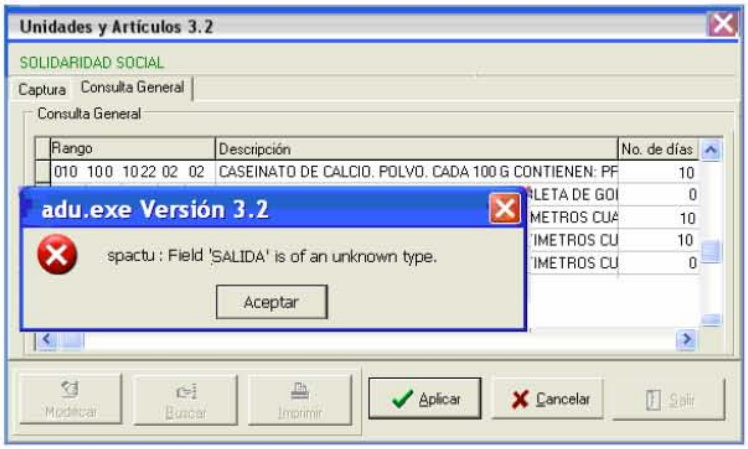
El origen de datos, un objeto TQuery o un TStored Procedure o un objeto TTable está habilitado en modo edición en Informix. La asociación a la base de datos en Oracle no permite el modo edición automáticamente.

Sugerencia

Agregar la sentencia <Nombre_Objeto>.edit al inicio de la forma o inmediatamente después de activar o abrir el dataset correspondiente.

```
<Nombre_Objeto>.open
<Nombre_Objeto>.edit
```

… XXX: Field "XXXX" is of an unknown type.



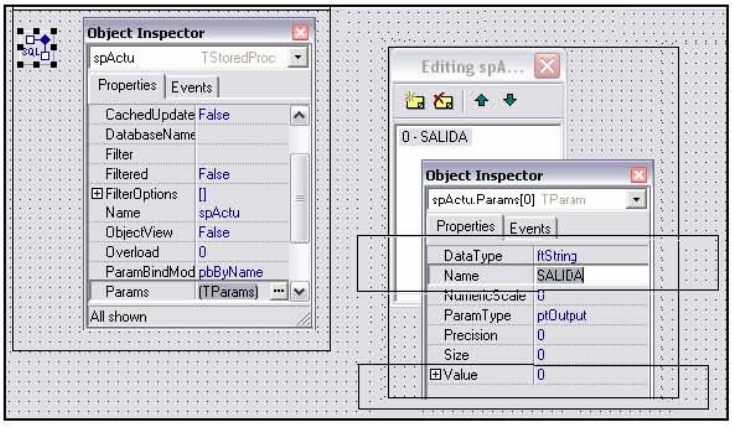
Observaciones

Desde la vista de diseño de la forma, al objeto de tipo TStoredProcedure llamado *spactu*, no se han asignado completamente las características de cada uno de los parámetros de IN/OUT.

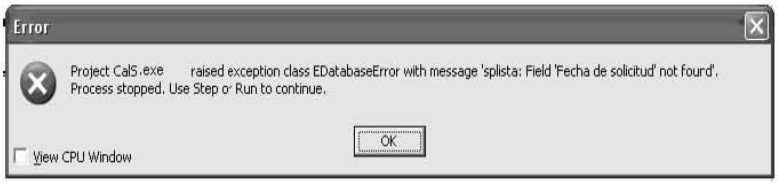
En el ejemplo el procedimiento spactu en la columna SALIDA no se ha registrado el output value 0 .

Sugerencia

Desde la pantalla de diseño de Delphi, actualizar los parámetros del objeto spactu que es de tipo TStoredProcname, y cuando el parámetro es de tipo OUT en Oracle, asignar un 0 (cero) en la columna Value del parámetro.



… XXX: Field "XXXX" is of an unknown type.



Observaciones

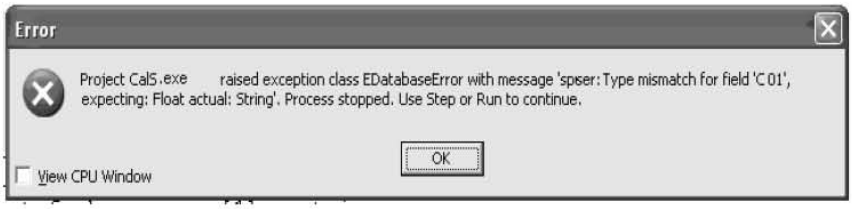
Se emplea un objeto tipo TStoredProcedure, al cual no se han asignado completamente los parámetros de retorno, o no se han actualizado.

En el ejemplo, al objeto splista no se han actualizado las características de los parámetros de salida, en este caso, el campo Fecha de Solicitud no existe.

Sugerencia

Desde la pantalla de diseño de Delphi, eliminar los parámetros del objeto solista, conectarse a la base de datos y actualizar los parámetros de la versión más reciente del stored procedure asociado.

… Type mismatch for field 'XXX', expecting:Float actual:String …



Observaciones

El tipo de dato definido para el objeto al que se asocia la columna C01 que recupera el objeto spser es Float; el stored procedure está regresando un dato de tipo String.

En el ejemplo, para el objeto spser el dato contenido en la columna C01 es de tipo caracter mientras que se espera un valor numérico.

Sugerencia

Desde la pantalla de diseño de Delphi, modificar la asignación del dato C01 de TFloatField a TStringField.

… [Oracle][ODBC][Ora]ORA-0904:"02":invalid identifier. …

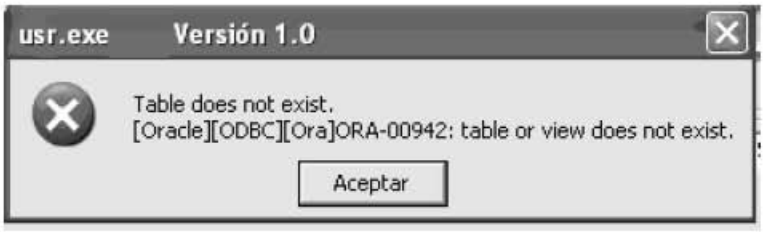
Observaciones

La cadena delimitada por comillas dobles es inválida.

Sugerencia

Reemplazar las comillas dobles de la consulta o asignación de parámetros por apóstrofe.

… [Oracle][ODBC][Ora]ORA-0942: table or view does not exist. …



Observaciones

La tabla o vista a la que se hace referencia no existe.

Sugerencia

Reescribir el nombre de la tabla o vista en mayúsculas y verificar que el objeto existe en la base de datos.

CAPITULO V

CONCLUSIONES

Del desarrollo de las diferentes actividades del proceso de migración de la base de datos podemos decir, que existen factores culturales, económicos, de estrategia institucional y técnicos que determinan en buena medida la forma en que se desarrolla el proyecto de migración de una base de datos.

Podemos decir que es posible migrar una base de datos de Informix SE2000 a Oracle 10gi y mantener en operación las aplicaciones Delphi 5 asociadas a la misma, cumpliendo con las características de operación originales.

En este trabajo probamos el proceso de migración del esquema de datos, de la información y finalmente el ajuste a las aplicaciones Delphi y los resultados obtenidos nos permiten establecer esta guía de trabajo como viable para procesos de migración similares.

Como requerimiento técnico, es necesario contar con conocimientos intermedios sobre las características de las herramientas de programación de la plataforma original; pero es indispensable conocer las características del DBMS destino, aún cuando se cuente con herramientas *case* de apoyo para la conversión de las estructuras de datos y el manejo de altos volúmenes de información.

El apoyo del personal de sistemas del área usuaria, para validar los procesos y resultado generados por las aplicaciones migradas es indispensable.

Para terminar diremos que es factor importante contar con un grupo de trabajo altamente integrado y orientado a alcanzar el objetivo común, solo de esta manera se pueden superar los retos técnicos y administrativos que la tarea presenta.

GLOSARIO DE TÉRMINOS

Alias: Es el nombre alternativo generalmente aplicado a una columna resultado de una consulta a la base de datos.

ANSI: Siglas de American National Standards Institute, organización encargada de definir estándares de tecnología; colabora con grupos industriales y es miembro de la International Organization for Standardization (ISO) y la International Electrotechnical Commission (IEC).

Bienes de consumo: Todos aquellos artículos que por su utilización en el desarrollo de las actividades, se desgastan o extinguen en su uso primario y por lo tanto, no son susceptibles de ser utilizados nuevamente**.**

BLOB: Estructura de datos que almacena imágenes, sonido o textos que por su extensión no pueden ser almacenados, cargados, exportados en la base de datos en estructuras más simples.

Byte: Es una unidad de datos de 8 dígitos binarios de longitud. En algunos sistemas de cómputo representa un carácter como una letra, número o símbolo tipográfico. En algunos sistemas de cómputo, cuatro bytes constituyen una palabra, la unidad para la que el procesador se diseñó para procesar eficientemente cada instrucción.

CASE: Computer-Aided Software Engineering; se refiere al uso de un método asistido por computadora para organizar y controlar el desarrollo de software. El uso de herramientas case facilita compartir una vista común del estado del proyecto de desarrollo de software en todo momento y para todos los integrantes del grupo de trabajo. Algunas herramientas case soportan los conceptos de programación estructurada. Debido a que el proceso de desarrollo hace énfasis en las pruebas y rediseño, el costo de desarrollo de un producto basándose en herramientas case se reduce c0onsiderablemente.

Control: Es un mecanismo preventivo y correctivo adoptado por la administración de una Dependencia o Entidad que permite la oportuna detección y corrección de desviaciones, ineficiencias o incongruencias en el curso de la formulación, procurar el cumplimiento de la normatividad que las rige y las estrategias, políticas, objetivos, metas y asignación de recursos.

Cuadro básico institucional: Documento que relaciona los bienes por grupo de suministro determinados como fundamentales e indispensables en la operación del Instituto.

DBMS: Data Base Management System.

Embedded SQL: Esquema de programación basada en el lenguaje C, que incluye sentencias propias de un manejador de base de datos.

Encapsulamiento: En general es la inclusión de una cosa dentro de otra de manera que el objeto incluido no es visible.

Hardware: Son los elementos físicos que componen un equipo de cómputo, procesador, monitor, Mouse, teclado, etc.

Inventario: La cantidad de propiedad disponible en cualquier momento dado; una lista desglosada por artículo de las cantidades de propiedad indicada como disponible en un momento determinado. Anotación ordenada de los bienes de una persona o de una entidad. Registro llevado en conexión con las actividades diarias de la dependencia.

Mainframe: Equipos de alta capacidad de procesamiento y almacenamiento de información; generalmente son empleados por corporativos de amplios recursos técnicos y económicos.

Metadata: Metadata son datos sobre los datos; explican los datos de manera que éstos puedan ser comprendidos y administrados apropiadamente. Los datos que se emplean para controlar y administrar a los datos desde una perspectiva técnica.

Proveedor: Es aquella persona física o moral que se encuentra disponible para vender los insumos que comercia y a quién se le puede encomendar mediante contrato el abastecimiento de bienes en favor del Instituto.

RDBMS: Relational Data Base Management System.

Repository (Repositorio): Es el almacenamiento central o lugar en el que se almacenan datos de manera organizada. Puede estar accesible directamente para los usuarios o puede ser un espacio de almacenamiento donde bases de datos, archivos o documentos se mantienen para una distribución posterior a través de la red.

Requester: Programa que solicita servicios o respuesta a un programa o proceso de cómputo.

Schema: Es la estructura u organización de la base de datos.

Script: Es un programa o secuencia de instrucciones que son interpretadas o seguidas por otro programa.

Software: Son los programas o procedimientos necesarios para habilitar una computadora para desarrollar una tarea específica; es diferente de los elementos físicos *(hardware)* del sistema. Por ejemplo un procesador de textos y software de sistema como el sistema operativo. El término "software" fue empleado por primera vez por John W. Tukey en 1957.

Wizard: Un wizard es un programa de computadora que actúa como interfase para guiar al usuario a través de una tarea compleja por medio de pantallas paso a paso; Microsoft Windows 95 fue el primer sistema operativo en el que se hizo uso de wizards. Es diferente de un sistema experto, que guía al usuario a través de una serie de preguntas con respuestas tipo Si/No para resolver un problema.

REFERENCIAS

Beginning Relational Data Modeling Second Edition.
Sharon Allen and Evan Terry
Apress

Local Area Network in Libraries
Kenneth E. Marks y Steven P. Nielsen
Meckler

Redes de Ordenadores
Andrew S. Tenenbaum
Printence Hall

Diseño y gestión de sistemas de base de datos
Ángel Lucas Gómez
Paraninfo

Ingeniería de Software
Rogers S. Pressman
Mc. Graw Hill

Informática Básica
Eduardo Alcalde
Mc. Graw Hill

LEVINE, G. GUILLERMO
Computación y programación moderna.
Perspectiva integral de la informática.
Pearson educación, México, 2001.
640 Págs.

Sampieri,Roberto.
Metodología de la investigación.
Mc Graw Hill; 3ª. Ed. México, 2003. 705 Págs.

Weitzenfeld ,Alfredo.
Ingeniería de software orientada a objetos con Uml, Java e Internet.
THOMSON, México, 2004. 678 Págs.

O'neil, schrader et al
Oracle Data Warehousing.
First edition.
Sams Publishing, 1997. 699 Págs.

Oracle 7 Server Concepts, Release 7.3
1996 Oracle Corporation

SITIOS CONSULTADOS

www.virtual.unal.edu.co/cursos/sedes/ manizales/4100010/lecciones/cap4/entidrelac.htm - 21k - cached – similar pages
www.virtual.uanl.edu.co/cursos/sedes/manizales/4100010/lecciones/cap4/entidrelac.htm
www.dcc.uchile.cl/ ccollazo/cc20a/e-r.html
http://www.oracle.com/technology/tech/migration/workbench/htdocs/quicktour/index_3.html
http://www.oracle.com/technology/documentation/migration.html