UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

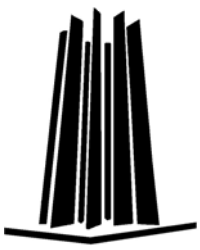
FACULTAD DE ESTUDIOS SUPERIORES
ARAGÓN

“ACTUALIZACIÓN EN TÓPICOS DE ADMINISTRACIÓN DE BASES DE DATOS”

T R A B A J O E S C R I T O
EN LA MODALIDAD DE SEMINARIOS Y CURSOS DE ACTUALIZACIÓN Y CAPACITACIÓN PROFESIONAL
QUE PARA OBTENER EL TÍTULO DE:
INGENIERO EN COMPUTACIÓN
P R E S E N T A:

M I N E R V A P A R E D E S D U R A N

ASESOR: M. EN C. MARCELO PÉREZ MEDEL



MÉXICO, 2009.

Dedicatoria

A Dios por cuidar mis pasos y permitirme llegar hasta este momento.

A mi amado esposo por su amor y apoyo incondicional.

A mi pequeño Gabriel por ser la inspiración más grande de mi vida.

A mis padres por darme la vida y entregarme su mayor esfuerzo.

A mi hermano Daniel, porque siempre estás en mi corazón.

A mi hermano Julio, porque sé que juntos llegaremos muy lejos... Cuentas conmigo.

A mi tía Socorrito y prima July por su cariño y buen ejemplo.

Agradecimientos

En primer lugar a mi asesor el M. en C. Marcelo Pérez Medel, por todo su tiempo, apoyo y compresión, ya que sin él no hubiera sido posible cumplir esta meta.

Al Mat. Luis Ramírez Flores, por todos aquellos consejos que contribuyeron en mi crecimiento personal y profesional.

A los Ing. Silvia Vega Muytoy, Juan Gastaldi Pérez y Gabriel Ortiz Cordero, por su valioso tiempo, observaciones y sugerencias para el mejoramiento de este trabajo.

A la UNAM y todos mis profesores que contribuyeron en mi desarrollo profesional.

Índice

Antecedentes

Hoy en día la información es uno de los activos más importantes para las organizaciones ya sean estas de carácter público o privado, por lo cual requieren administrarla y asegurarla ante todo tipo de amenaza que ponga en riesgo las características que le dan valor.
La implementación de Bases de Datos permite a dichas organizaciones administrar la información de forma segura y eficiente.

El objetivo principal del presente trabajo consiste en proporcionar los conocimientos necesarios para una adecuada administración de las Bases de Datos; Así como mostrar una de las principales tendencias en cuanto al tema.

En el capítulo I, Modelo de Datos Relacional y Sistemas Manejadores de Bases de Datos Relacionales, se explican los conceptos fundamentales de las Bases de Datos, entre ellos encontramos la definición de Bases de Datos, así como los elementos que las conforman; Otro tema que también será tratado en este capítulo es el Sistema Manejador de Bases de Datos, su función, componentes y características.

En el capítulo II, Lenguaje SQL, se describirán los elementos básicos para el desarrollo de consultas a la Base de Datos utilizando el lenguaje SQL para el manejo de datos (insertar, modificar, eliminar y extraer la información requerida).

En el capítulo III, Administración de Bases de Datos, se describen las tareas básicas de administración en SYBASE Adaptive Server Enterprise 15.0.

En el capítulo IV, Tópicos Avanzados de Bases de Datos, se describe brevemente el concepto y características del Data Warehouse, el cual es una de las principales tendencias de las Bases de Datos, debido a que es una valiosa herramienta para la toma de decisiones.

Modelo de Datos Relacional y Sistemas Manejadores de Bases de Datos Relacionales

1.1 Introducción a las Bases de Datos

Anteriormente el administrar la información representaba una labor lenta y complicada; Conforme fue pasando el tiempo, fueron evolucionando los sistemas para administrar la información, desde los Sistemas de Gestión de Archivos[1] hasta lo que hoy conocemos como las Bases de Datos.
Para poder explotar toda la funcionalidad que ofrecen las Bases de Datos es necesario contar con un software especial que es el Sistema Manejador de Bases de Datos el cual permite administrar de forma segura y eficiente una Base de Datos.

Ahora bien para poder entender la importancia de las Bases de Datos en cuanto a la administración de la información tenemos que conocer los conceptos básicos que son desarrollados en este capítulo.

1.1.1 Dato

Un dato es la unidad mínima de información, de forma genérica se dice que un dato se puede definir como un hecho aislado y en bruto que por sí sólo no tiene significado ni valor.
Relacionando el concepto dato a las Bases de Datos, se puede decir que un dato es la representación simbólica (numérica, alfabética, etc.) de un atributo o característica de un objeto o entidad.

1.1.2 Información

Se define como un conjunto de datos relacionados entre sí y que dentro de un contexto determinado tienen un valor y significado; Es posible realizar una toma de decisiones a partir de la información.

La obtención de información formal genera gastos, su valor sólo puede ser comparado con el valor que tendrá para el receptor final.

El costo de producir información es tangible, se puede medir gracias a algunos dispositivos y medios utilizados, pero la información es conceptual por naturaleza y no tiene características tangibles únicamente representaciones simbólicas.

Para dar valor a la información se utilizan las siguientes características:

- *Accesible:* Facilidad y rapidez con que se obtiene la información
- *Clara:* Entendimiento de la información sin ambigüedades
- *Precisa:* Lo más exacta posible
- *Propia:* Relación entre el usuario y lo solicitado por el mismo

[1] Sistema que consistía en almacenar la información en varios archivos de texto, dichos archivos no tenían relación entre ellos, lo que resultaban en información redundante.

- *Oportuna:* Disponible cuando se requiera
- *Flexible:* Adaptable a la toma de decisiones
- *Verificable:* Permite el Análisis
- *Imparcial:* Modificación por el dueño de la información únicamente
- *Cuantificable:* Cualquier dato procesado produce información

1.2 Base de Datos

Una Base de Datos es un conjunto de datos interrelacionados, que tienen un objetivo en común y una estructura basada en las relaciones y restricciones reales existentes entre cada dato. Una Bases de Datos puede verse como un único repositorio central de información al cual tienen acceso múltiples usuarios al mismo tiempo.
El objetivo de las Bases de Datos es almacenar y administrar la información de las organizaciones.

1.2.1 Tipos de Bases de Datos

Una Base de Datos puede clasificarse ya sea por su contenido o bien por la variabilidad de su contenido, a continuación enlisto los tipos de Base de Datos según su criterio de clasificación:

Según la variabilidad de los datos almacenados:

- *Bases de Datos estáticas*: su contenido no puede ser modificado ya que son Bases de Datos solo de consulta, por lo general almacenan datos históricos que sirven como fuente de información para realizar una toma de decisiones o bien crear modelos de comportamiento

- *Bases de Datos dinámicas*: su contenido puede ser modificado a través del tiempo agregando, modificando o eliminando datos, dentro de esta categoría se encuentran las Bases de Datos Transaccionales, las cuales almacenan la información generada en el día a día de una organización

Según su contenido:

- *Bibliográficas*
- *Numéricas*
- *Directorios*
- *Bancos de imágenes, audio, video, multimedia, etc*.

1.2.2 Arquitectura de las Bases de Datos

Con la finalidad de separar la Base de Datos física de los programas de aplicación que acceden a ella, las Bases de Datos se alinean con la arquitectura definida por el grupo ANSI/SPARC[2] la cual divide a la Base de Datos en tres niveles: externo, conceptual e interno.

- *Nivel interno*: En este nivel se define como se almacenará físicamente la información, así como los métodos para acceder a ella. También es conocido como *Nivel físico*

- *Nivel conceptual*: Cada organización cuenta con diferentes áreas que soportan su operación diaria, cada uno de estas áreas tiene ciertos requerimientos de información y aplicaciones, por lo cual la arquitectura de Bases de Datos debe abstraer todos estos requerimientos y plasmarlos en la Base de Datos, el nivel conceptual hace que esto sea posible gracias a que representa todos estos requerimientos incluyendo la definición de los datos y las relaciones entre ellos

- *Nivel externo*: Los usuarios y aplicaciones pueden acceder a la información contenida en la Base de Datos por medio de este nivel

- *Las correspondencias*: Se pueden definir como una asociación de distintas representaciones para un mismo dato

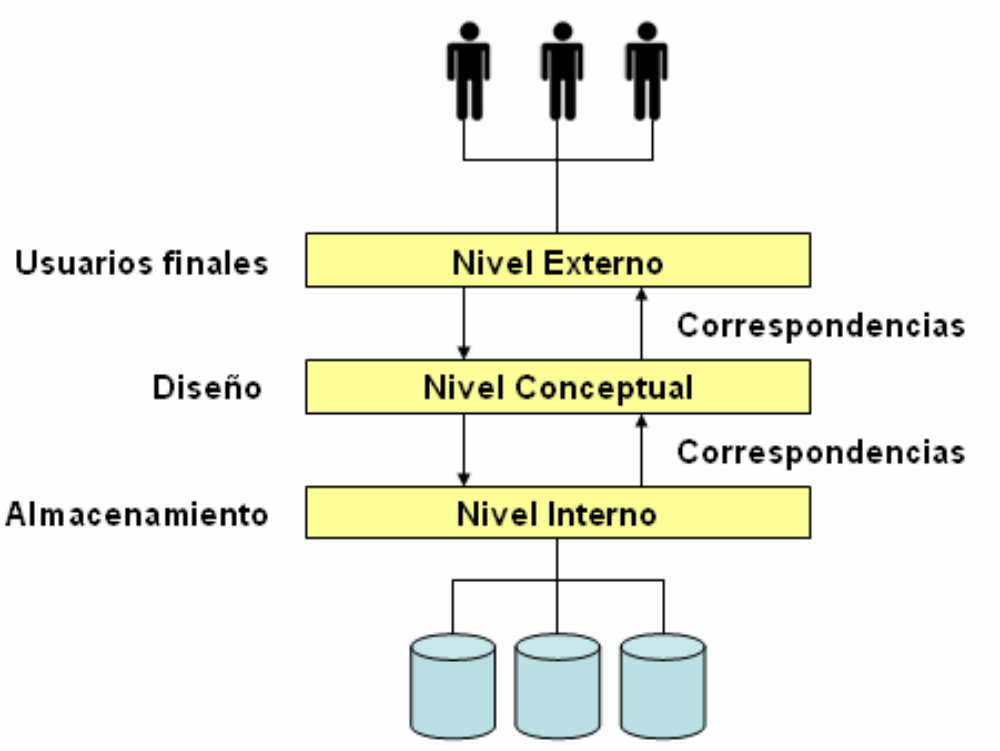


Fig. 1-1 Arquitectura ANSI/SPARC

[2] ANSI/SPARC es un grupo de estudio del Standard Planning and Requirements Committee (SPARC) del ANSI (American National Standards Institute), que se ocupa de todo lo relacionado a computadoras e informática

1.2.3 Independencia de datos

Partiendo de la arquitectura de tres niveles se puede conceptualizar a la independencia de datos como aquella que hace posible modificar el esquema en un nivel sin necesidad de modificar el esquema del nivel superior inmediato.

Existen dos tipos de independencia de datos:

- *Independencia lógica*: al modificar una Base de Datos ya sea al agregar o eliminar una entidad, se dice que se está modificando el esquema en un nivel conceptual, y gracias a la independencia lógica es posible realizar esta modificación sin tener que actualizar los esquemas del nivel externo que no tengan relación con dicha entidad

- *Independencia física*: es posible modificar el esquema interno sin tener que modificar el esquema en el nivel conceptual o externo. Por ejemplo, para mejorar el rendimiento al ejecutar una consulta o actualización de datos, puede ser necesario modificar la ubicación de los archivos físicos, la independencia física permita realizar esta modificación sin tener que realizar modificaciones adicionales a los niveles superiores

1.2.4 Arquitectura Cliente-Servidor

La arquitectura Cliente-Servidor se define como la relación entre una máquina llamada “cliente” quien realiza una solicitud a otra máquina llamada “servidor” que será la encargada de realizar las tareas necesarias para responder a la solicitud del cliente y así enviarle un resultado. Para que esta relación sea posible ambas máquinas deben estar conectadas a la misma red.
Gracias a la arquitectura Cliente-Servidor se optimiza el rendimiento, ya que el procesamiento de una solicitud se divide entre el cliente y servidor.

Hablando de Bases de Datos, se observa que esta arquitectura es de gran utilidad ya que el cliente es el encargado de la interface con el usuario (pantallas, reportes, etc.), dejando al servidor de Bases de Datos a cargo de la ejecución de consultas a la Base de Datos, el almacenamiento e integridad de los datos.

Durante el procesamiento de una consulta o actualización a la Base de Datos se pueden identificar los siguientes pasos:

1. El usuario crea solicitud ya sea de consulta o actualización.

2. El cliente adecua la solicitud para que pueda ser entendida por el servidor de Base de Datos, y se la envía a través de la red.

3. El servidor de Base de Datos procesa la solicitud, el resultado obtenido lo envía al cliente a través de la red.

4. El cliente recibe el resultado y lo despliega al usuario.

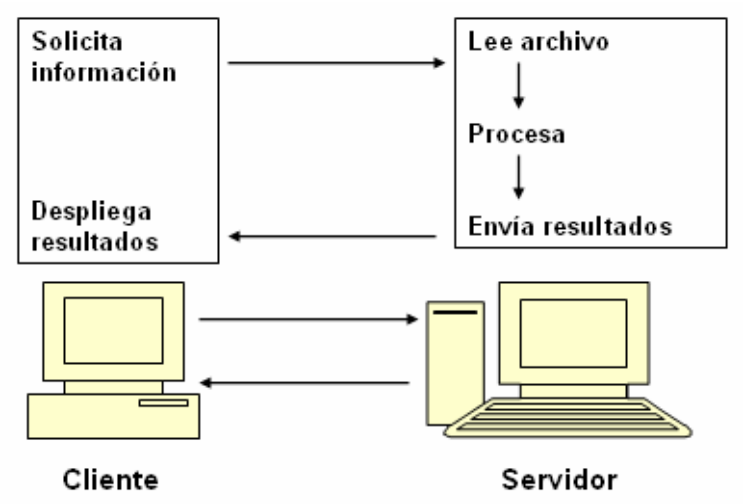


Fig. 1-2 Arquitectura Cliente-Servidor

1.2.5 Características de las Bases de Datos

Redundancia

La redundancia de datos se refiere a la existencia de información duplicada en diferentes tablas dentro de una Base de Datos, lo cual requiere múltiples procedimientos de entrada y actualización, así mismo conduce a muchos problemas que tienen que ver con la integridad y consistencia de los datos. Dentro de una Base de Datos la redundancia debe ser mínima y controlada, en ocasiones existirán motivos válidos de negocios o técnicos para mantener redundancia.

Consistencia

Decimos que la información es consistente cuando su valor es idéntico en todos los lugares en donde se hace referencia a ella, frecuentemente los problemas de consistencia de información se deben a la redundancia, es muy probable que surjan incongruencias al almacenar la misma información en más de un lugar; ya que al modificarla, eliminarla o agregarla se debe repetir el mismo proceso para cada uno de los campos en donde es almacenada con el riesgo de no realizarlo en su totalidad, generando en este caso la inconsistencia.

Integridad

La integridad de una Base de Datos se refiere no sólo a que los datos sean consistentes dentro de la Base de Datos, sino que además, los valores que posean los datos sean válidos de acuerdo a las dependencias funcionales entre tablas y de acuerdo a las políticas de negocio.

Seguridad

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

Además de las características anteriores una Base de Datos debe cumplir las siguientes condiciones:

- Los datos han de estar almacenadas juntos

- Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento ya que lo importante para estos es la información contenida

- Los datos son compartidos por diferentes usuarios y programas de aplicación; por lo tanto debe existir un mecanismo común para inserción, actualización, borrado y consulta de datos

- Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes sistemas manejadores de Base de Datos

1.3 Modelo de Datos

Los modelos son utilizados para representar la realidad, ya sean hechos, comportamientos o simplemente objetos o conceptos.
Para poder crear una Base de Datos, es necesario contar con una representación grafica que describa las entidades y las relaciones existentes entre ellas, a esta representación se le conoce como “Modelo de Datos”.

Podemos encontrar diferentes definiciones de Modelo de Datos, pero opino que la definición de Jeffrey Ullman describe mejor el concepto:

“Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. Es formal pues los objetos del sistema se manipulan siguiendo reglas perfectamente definidas y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar.”

Clasificación de Modelos

Los modelos de datos se dividen en tres grupos:

- Modelos lógicos basados en objetos

- Modelos lógicos basados en registros

- Modelos físicos de datos

1.3.1 Modelos lógicos basados en objetos

Como se definió anteriormente la abstracción del mundo real la podemos representar en los niveles conceptual y externo, los modelos lógicos basados en objetos son utilizados para realizar esta abstracción, son fáciles de estructurar ya que permiten representar los datos, sus relaciones y restricciones tal como los captamos en el mundo real.
Existen diferentes modelos de este tipo, pero el más utilizado es el modelo Entidad-Relación por su fácil estructuración.

Modelo E-R (Entidad-Relación)

Las entidades son objetos con características propias llamadas atributos, las entidades se diferencian de otras gracias a sus atributos y relaciones con otras entidades. Con el modelo E-R podemos representar a estas entidades y sus relaciones.
Los símbolos utilizados en el modelo E-R son los siguientes:

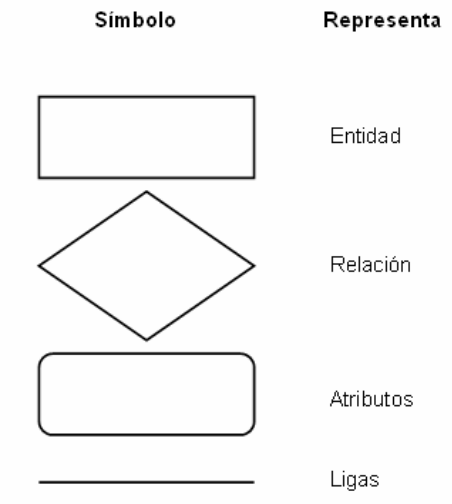


Fig. 1-3 Simbología Entidad-Relación

Para entender mejor el concepto expondré el siguiente ejemplo:
En una librería, los vendedores ganan una comisión extra por cada venta realizada; utilizando el modelo E-R identificamos como entidades al vendedor y el libro, dejando la venta como la relación existente entre estas 2 entidades; gráficamente se ilustra:

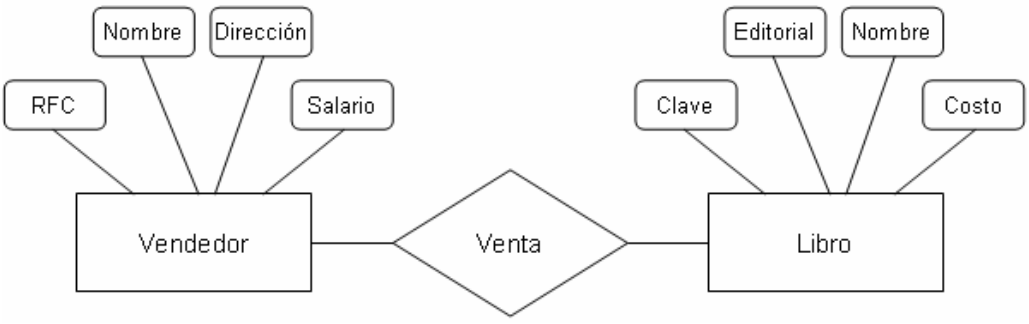


Fig. 1-4 Modelo Entidad-Relación

1.3.2 Modelos lógicos basados en registros

Para representar los niveles conceptual e interno se utilizan este tipo de modelos; que para representar la realidad emplean registros e instancias, así como ligas o apuntadores que representan las relaciones existentes entre estos registros. Son utilizados para especificar la estructura lógica de la Base Datos.

Los tres modelos lógicos basados en registros más comunes son:

- Modelo relacional
- Modelo de red
- Modelo jerárquico

Modelo relacional

Para representar los datos y sus relaciones, este tipo de modelo utiliza tablas, en donde los registros son representados por los renglones (tuplas) y las características (atributos) de estos registros corresponden a las columnas dentro de las tablas.

RFC	Nombre	Dirección	Salario
MADG771003	Gabriel Macías Duran	Graciela #38	5,000
MESC790523	Carlos Mena Suárez	Oniquina #13	5,000

Tabla 1-1 Vendedores

Clave	Editorial	Nombre	Costo
A1124	Trillas	Cómo aprovechar el tiempo	135
B0032	Planeta	El beso de la virreina	250

Tabla 1-2 Libros

Para representar las relaciones en este modelo, primero debemos entender el concepto de llave primaria, que se define como el campo que representa un atributo principal y que permite identificar de forma única a una entidad.

Existen dos formas de representar las relaciones en este modelo:

1.2.1 Crear una tabla con cada una de las llaves primarias de las entidades para las que se pretende representar una relación.

RFC	Clave
MADG771003	A1124
MESC790523	B0032

Tabla 1-3 Relación tipo a

2.2.1 Incluir en una de las tablas (entidades) involucradas, la llave primaria de la otra tabla.

RFC	Nombre	Dirección	Salario	Clave
MADG771003	Gabriel Macías Duran	Graciela #38	5,000	A1124
MESC790523	Carlos Mena Suárez	Oniquina #13	5,000	B0032

Tabla 1-4 Relación tipo b

Modelo de red

Para representar los datos el modelo de red utiliza registros, y las ligas o enlaces representan las relaciones entre los registros. Las ligas o enlaces pueden verse como punteros.
Una Base de Datos de red, está formada por un grupo de registros, y sus respectivos enlaces.
Un *registro* es un conjunto de campos (atributos), cada campo tiene almacenado un valor, el *enlace* es la relación existente entre dos registros.

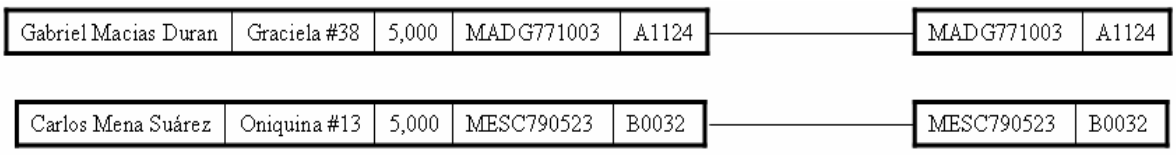


Fig. 1-5 Modelo de red

Modelo jerárquico

Este tipo de modelo representa al igual que el modelo de red a los datos y sus relaciones por medio de registros y ligas. A diferencia del modelo de red, el modelo jerárquico organiza los datos y sus relaciones en un conjunto de árboles y no de graficas arbitrarias. Utiliza un nodo ficticio para representar la raíz del árbol.
Por lo tanto una Base de Datos jerárquica será un grupo de árboles del siguiente tipo.

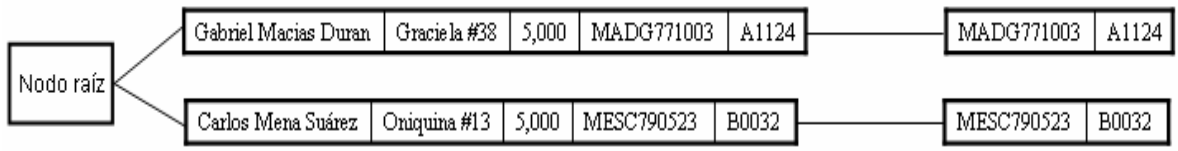


Fig. 1-6 Modelo jerárquico

1.3.3 Modelos físicos de datos

Se usan para representar el nivel interno o físico, existen pocos modelos de este tipo.
Describen la forma en que será almacenada la Base de Datos a nivel hardware (discos).

Se clasificación en dos tipos:

- Modelo unificador
- Memoria de elementos

1.4 Sistema Manejador de Bases de Datos Relacionales (RDBMS)

Para administrar el acceso a los datos almacenados en una Base de Datos, es utilizado un software especial llamado Sistema Manejador de Bases de Datos o Data Base Management System (DBMS). El cuál es la interface entre los datos físicos (es decir, los datos tal y como están almacenados en el hardware) y los usuarios y/o aplicaciones de la Base de Datos.

Ahora bien un RDBMS es un Sistema Administrador de Bases de Datos Relacionales o Relational Data Base Management System (RDBMS), el cual administra las Bases de Datos y las relaciones entre cada una de las tablas contenidas en ella.

1.4.1 Reglas de Codd

Existen doce reglas para definir si un DBMS es relacional o no; estas reglas son el pilar de los RDBMS y fueron públicas por el Dr. Edgar Frank Codd en 1985.

Regla 1: Regla de la información
Toda la información en una Base de Datos Relacional se representa explícitamente en el nivel lógico exactamente de una manera, como valores en una tabla.

Regla 2: Regla del acceso garantizado
Para todos y cada uno de los datos (valores atómicos) de una Base de Datos Relacional se garantiza que son accesibles a nivel lógico utilizando una combinación de nombre de tabla, valor de clave primaria y nombre de columna.

Regla 3: Tratamiento sistemático de valores nulos
Los valores nulos (que son distintos de la cadena vacía, blancos y 0) se soportan en los DBMS totalmente relacionales para representar información desconocida o no aplicable de manera sistemática, independientemente del tipo de datos.

Regla 4: Catalogo dinámico en línea basado en el modelo relacional
La descripción de la Base de Datos se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, igual que lo aplican a los datos normales.

Regla 5: Regla del sub-lenguaje de datos completo
Se debe contar con un sub-lenguaje que contemple la definición de datos, la definición de vistas, la manipulación de datos, las restricciones de integridad, la autorización, el inicio y fin de una transacción.

Regla 6: Regla de actualización de vistas
Todas las vistas que son teóricamente actualizables se pueden actualizar por el sistema.

Regla 7: Inserción, actualización y borrado de alto nivel
La capacidad de manejar una relación base o derivada como un solo operando se aplica no sólo a la recuperación de los datos (consultas), sino también a la inserción, actualización y borrado de datos.

Regla 8: Independencia física de datos
Los programas de aplicación y actividades en terminales permanecen inalterados a nivel físico cuando quiera que se realicen cambios en las representaciones de almacenamiento o métodos de acceso.

Regla 9: Independencia lógica de datos
Los programas de aplicación y actividades en terminales permanecen inalterados a nivel lógico cuandoquiera que se realicen cambios a las tablas base que preserven la información

Regla 10: Independencia de integridad
Los limitantes de integridad específicos para una determinada Base de Datos relacional deben poder ser definidos en el sublenguaje de datos relacional, y almacenables en el catálogo, no en los programas de aplicación.

Regla 11: Independencia de distribución
Debe existir un sub-lenguaje de datos que pueda soportar Bases de Datos distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos por primera vez o se redistribuyan éstos posteriormente.

Regla 12: Regla de la no subversión
Si un sistema relacional tiene un lenguaje de bajo nivel (un sólo registro cada vez), ese bajo nivel no puede ser utilizado para suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

1.4.2 Características del RDBMS

Los RDBMS deben facilitar la integridad, seguridad y acceso a los datos; así como reducir la redundancia en su almacenamiento.
Deben mantener las aplicaciones independientes del almacenamiento físico de los datos.

Un RDBMS debe permitir las siguientes condiciones en una Base de Datos:

- Los datos han de estar almacenados juntos
- Tanto los usuarios finales como los programas de aplicación no necesitan conocer los detalles de las estructuras de almacenamiento
- Los datos son compartidos por diferentes usuarios y programas de aplicación, por tal motivo debe existir un mecanismo común para la inserción, actualización, borrado y consulta de los datos
- Tanto datos como procedimientos pueden ser transportables conceptualmente a través de diferentes RDBMS

1.4.3 Arquitectura del RDBMS

Los componentes de un RDBMS son ilustrados con la siguiente figura:

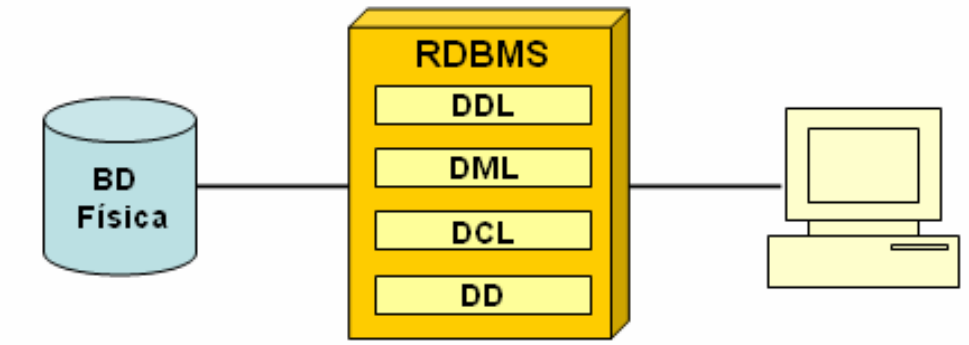


Fig. 1-7 Arquitectura del RDBMS

Lenguaje de Definición de Datos o Data Definition Language (DDL)

Permite crear, modificar y eliminar estructuras de datos como: tablas, Bases de Datos, índices, etc.; es decir, permite definir la estructura de la Base de Datos mediante comandos como *Create* (crear), *Drop* (eliminar), o *Alter (modificar)*.

Lenguaje de Manipulación de Datos o Data Manipulation Language (DML)

Se utiliza para realizar consultas y edición de la información contenida en la Base de Datos, esto implica: seleccionar (select), insertar (insert), borrar (delete) y modificar (update).

Lenguaje de Control de Datos o Data Control Language (DCL)

Se utiliza para la definición de los privilegios de control de acceso y edición a los elementos que componen la Base de Datos (seguridad), es decir, permitir (grant) o revocar (revoke) el acceso.
Los permisos a nivel Base de Datos pueden otorgarse a usuarios para ejecutar ciertos comandos dentro de la base o para que puedan manipular objetos como roles y los datos que puedan contener estos.

Diccionario de Datos o Data Dictionary (DD)

El diccionario de datos contiene la definición de los objetos almacenados en la Base de Datos, a esta definición se le conoce como metadatos, que por lo general se describen como “datos acerca de los datos”.
En ocasiones el diccionario de datos puede ser muy completo, ya que puede incluir las referencias que describen los datos utilizados por un programa o bien los informes o reportes que requiere cada área dentro de una organización, etc.

1.4.4 Esquema de seguridad del RDBMS

El acceso al RDBMS se administra por 3 niveles:

- Seguridad a nivel Servidor: El usuario final debe tener una cuenta válida dentro de la capa del servidor RDBMS
- Seguridad a nivel de Base de Datos: El usuario final debe tener una cuenta valida dentro de una Base de Datos
- Seguridad a nivel de Permisos sobre Objetos y Comandos: El usuario final debe tener privilegios sobre los objetos de la Base de Datos y comandos

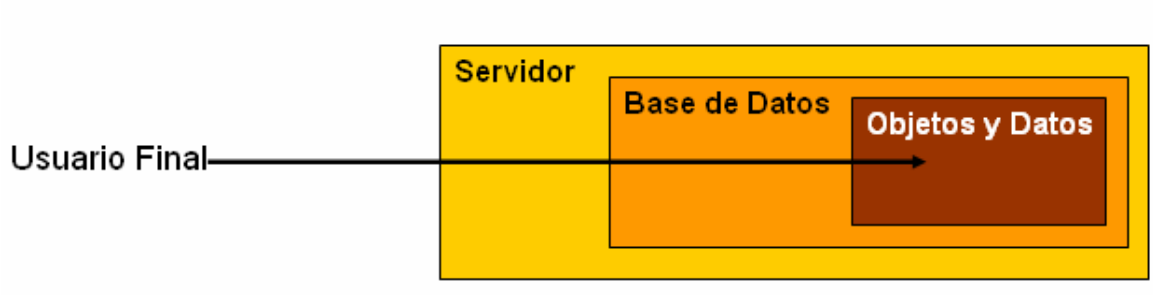


Fig. 1-8 Esquema de seguridad del RDBMS

1.4.5 Clasificación de los RDBMS

Los RDBMS pueden ser clasificados por ámbito o volumen de información que pueden almacenar.

Las siguientes tablas muestran los RDBMS más populares y su clasificación:

Ámbito	
Comerciales	**Software libre**
• SQL Server • Sybase • Oracle • Informix • DB2	• PostgresSQL • Mysql • Sybase(Linux)

Tabla 1-5 RDBMS por ámbito

Volumen de Información	
Corporativo	**Departamental**
• Oracle • Informix • Sybase DB2 • PostgreSQL	• SQL Server • Mysql

Tabla 1-6 RDBMS por volumen de información

Lenguaje SQL

El Lenguaje de Consunta Estructurado o Structured Query Language (SQL) es un lenguaje de consulta para Bases de Datos, fácil de entender ya que su estructura utiliza palabras en inglés, lo que lo hace fácil de aprender y utilizar, las instrucciones se enfocan a qué buscar, dejando al RDBMS la tarea de cómo recuperar la información.

La ventaja de la adopción del ANSI SQL, es que los diversos RDBMS tienen que acoplarse al estándar, permitiendo así una mayor compatibilidad entre ellos. Esto implica que conociendo una variante del lenguaje SQL, se tienen los conocimientos necesarios para poder utilizarlo en todos los RDBMS (MS SQL Server, Oracle, Sybase, MySQL, PostgreSQL, DB2, entre otros). Aunque los distintos fabricantes tratan de acoplarse al estándar ANSI SQL, es cierto que cada uno implementa funcionalidades extra que le dan un valor agregado a su producto pero sacrificando un poco la compatibilidad, por lo cual se podrán notar ciertas diferencias entre distintos RDBMS.

2.1 Estándares SQL ANSI 89, 92 y 99

El lenguaje SQL, fue adoptado como estándar de la industria en 1986. Desde entonces se han realizado revisiones al estándar para incorporar nueva funcionalidad conforme la industria de las Bases de Datos lo va requiriendo. Una de las revisiones más importantes fue la de 1992, conocida como ANSI SQL92, actualmente la versión soportada por la mayoría de los RDBMS es el ANSI SQL99 también conocido como SQL3.
A continuación enlisto algunas de las características de las diferentes versiones de SQL:

2.1.1 SQL ANSI 89

- Agregan la capacidad conocida como integridad referencial y la descripción de todo el modelo relacional
- Se definió que el lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado
- Estos elementos se combinan para definir y manipular la Base de Datos
- Se establecen los elementos de un DBMS (DDL, DML y DCL), así como las instrucciones y sintaxis relacionadas con cada uno de ellos
- Establecimiento de las cláusulas del comando ***SELECT,*** las cuales son: FROM, WHERE, GROUP BY, HAVING, ORDER BY
- Definición de los operadores lógicos: AND, OR y NOT y de comparación
- Se determinan las funciones de agregado, tales como: AVG, COUNT, SUM, MAX, MIN

2.1.2 SQL ANSI 92

- Toma todas características definidas en el estándar ANSI SQL 89
- Permite la definición de esquemas
- Permite la definición de dominios por parte de los usuarios, es decir, tipos de datos definidos por el usuario
- Menciona las consideraciones para realizar consultas sencillas, multi-tablas y sub-consultas
- Incluye los operadores EXISTS y NOT EXISTS
- Contempla el uso de la palabra DISTINCT en una consulta
- Menciona algunas consideraciones para el uso de las cláusulas GROUP BY y HAVING
- Especifica la definición de vistas en una Base de Datos

2.1.3 SQL ANSI 99

- Toma todas características definidas en los estándares ANSI SQL 89 y 92
- Incluye nuevos tipos de datos escalares: BOOLEAN, CLOB (objeto de caracteres largo) y BLOB (objeto binario grande)
- Presenta dos nuevos operadores de totales: EVER y ANY
- Incorpora generadores de tipo de dato: REF, ARRAY y ROW
- Incluye los operadores EXISTS y NOT EXISTS
- Contempla el uso de la palabra DISTINCT en una consulta
- Soporta una opción LIKE en CREATE TABLE, lo cual permite que todas o algunas definiciones de columna de una nueva tabla sean copiadas a partir de otra ya existente
- Incluye la cláusula WITH para introducir nombres abreviados para determinadas expresiones
- Incorpora una nueva expresión de condición IS DISTINCT para la cláusula FROM

2.2 Definición de datos

Antes de comenzar a trabajar con SQL, es necesario conocer los elementos que intervienen en la definición de la información en una Base de Datos, para poder manipularla de manera adecuada.
La tabla es el elemento fundamental de una Base de Datos Relacional, la cual consiste de una serie de renglones (registros) que representan la información. Cada renglón está dividido en columnas (campos) los cuales deben de tener un tipo de dato establecido.

2.2.1 Tipos de datos del sistema

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

Los tipos de datos que se manejan en una base, pueden variar ligeramente entre diferentes RDBMS, sin embargo el estándar ANSI, asegura que cierto tipo de datos estará presente en cualquier RDBMS asegurando así la compatibilidad. Algunos RDBMS implementan sinónimos para los tipos de datos, de manera que puedan cumplir con el ANSI SQL99, sin embargo internamente son convertidos a un tipo de dato que si esté soportado. Por ejemplo MS SQL Server acepta el tipo de dato DOUBLE PRECISION pero lo convierte y maneja como un FLOAT.

En la siguiente tabla se muestra en la primera columna, el tipo de dato como se especifica en el lenguaje ANSI SQL y en las demás columnas se indica el tipo de datos equivalente por RDBMS que cumple con dicho estándar.

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

Tabla 2-1 Tipo de datos

2.2.2 El Valor Nulo

Es importante conocer el concepto de valor nulo, en el contexto de una Base de Datos, debido a que frecuentemente un valor nulo es confundido con un valor numérico de 0 o una cadena vacía. Un valor nulo se representa en SQL con la cláusula NULL y representa la ausencia de información.

Por ejemplo si en un listado de empleados algunos aparecen en el dato de comisión como NULL, esto indica que no se tiene dicha información aunque erróneamente se podría interpretar que estos empleados tienen comisión del 0%. Ahora, suponiendo que los datos que aparecen como NULL corresponden a la edad, indican que el dato no fue proporcionado y por lo tanto se carece de dicha información y claramente se observa que no es lo mismo que una edad de 0 años.
Hasta este momento se ha hablado únicamente de tipos de datos numéricos, aunque los valores nulos también se utilizan en cualquier otro tipo de dato, por ejemplo en texto: No es lo mismo una cadena vacía que un valor nulo.

2.3 Tablas

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

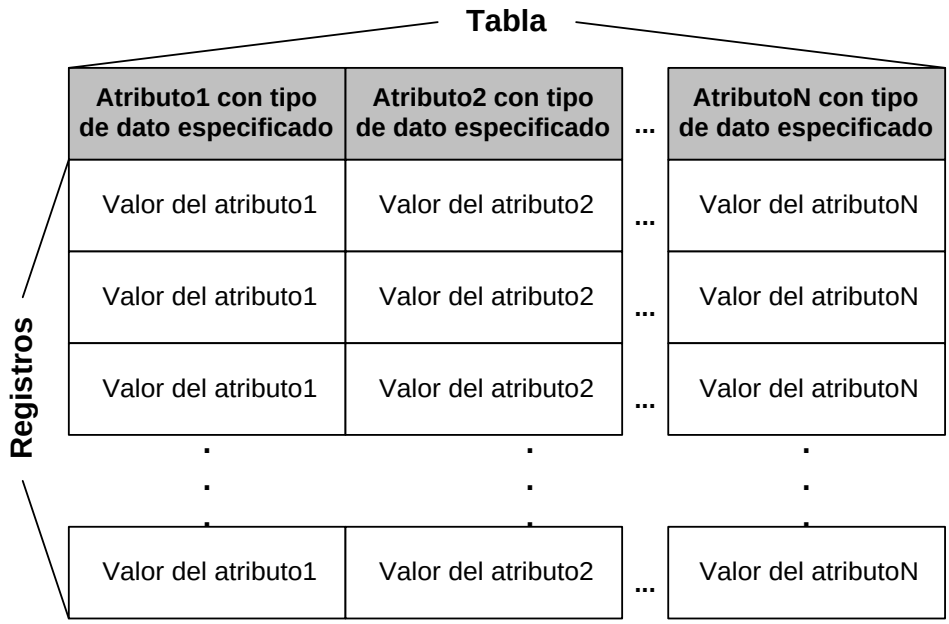


Fig. 2-1 La Tabla

2.3.1 Convención de nombres

Los nombres que se le asignan a los objetos de una Base de Datos, están sujetos a ciertas reglas que varían entre RDBMS, incluso de una versión a otra del mismo. Algunas de las reglas que deben observarse se muestran en la siguiente tabla:

RDBMS	Reglas
Todos	• No pueden usarse palabras reservadas del servidor SQL. • Es aconsejable usar nombres descriptivos. • No pueden existir dos objetos (aunque sean de distinto tipo) con el mismo nombre para un usuario en particular
Sybase	• Deben empezar con una letra. • Pueden contener letras: A-Z, a-z, números: 0-9, _, #, $.
Oracle	• Deben empezar con una letra. • Pueden contener letras: A-Z, a-z, números: 0-9, _, #, $.
MySQL	• Puede empezar con los símbolos _, $, letras o números • Pueden contener letras: A-Z, a-z, números: 0-9, _, $.
PostgreSQL	• Puede empezar con el símbolo _, y letras • Pueden contener letras: A-Z, a-z, números: 0-9, _, $
Ms SQL Server	• Puede empezar con el símbolo #,_, letras • Pueden contener letras: A-Z, a-z, números: 0-9, _, $

Tabla 2-2 Convención de Nombres

2.3.2 Creación de tablas

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

Sintaxis	CREATE TABLE <*nombre_tabla*> (<*nombre_campo1*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>], <*nombre_campo2*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>], … <*nombre_campoN*> <*tipo_de_dato*> [NULL | NOT NULL] [DEFAULT <*val_predeterminado*>])
Ejemplo	CREATE TABLE pais (id_pais numeric(3) NOT NULL, nombre varchar(100) NOT NULL)

Tabla 2-3 Sintaxis creación de tablas

2.3.3 Modificación de tablas

Dependiendo del RDBMS esta modificación de estructura de la tabla ofrece mayor o menor flexibilidad. Por ejemplo, MySQL es de las más flexibles, porque permite cambiar incluso el tipo de dato de una columna mientras que en Sybase o MS SQL Server es necesario eliminar y volver a crear la tabla con la estructura deseada.

Características soportadas:	MySQL	PostgreSQL	Oracle	Sybase	MS SQL Server
Agregar campo	Si	Si	Si	Si	Si
Eliminar campo	Si	Si	Si	No	Si
Renombrar campo	Si	Si	Si	No	No
Cambiar el tipo de dato del campo	Si	No	Si	No	Si
Definir llave primaria	Si	Si	Si	Si	Si
Definir llave foránea	Si	Si	Si	Si	Si
Renombrar tabla	Si	Si	Si	No	No

Tabla 2-4 Los RDBMS y el soporte para modificación de tablas

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

La sintaxis general para agregar un campo es la siguiente:

Sintaxis	ALTER TABLE <*nombre_tabla*> add <*nombre_campo*> <*tipo_dato*> [DEFAULT <val*_predeterminado*>] [NOT NULL | NULL]
Ejemplo	ALTER TABLE pais add superficie numeric(6) NULL

Tabla 2-5 Sintaxis modificación de tablas

Cabe señalar que siempre que se agregue un campo a una tabla, éste debe permitir valores NULOS.

2.3.4 Eliminación de tablas

Para eliminar una tabla y los datos que contiene, se utiliza la siguiente instrucción:

Sintaxis	DROP TABLE <*nombre_tabla*>
Ejemplo	DROP TABLE pais

Tabla 2-6 Sintaxis eliminación de tablas

2.4 Reglas

Las reglas dentro de la Base de Datos permiten definir condiciones que debe cumplir la información para que sea válida. Por ejemplo se puede definir una regla que especifique que los ingresos de un empleado no sobrepasen los $10,000 pesos.

En la práctica, muchas de estas reglas no se definen en la Base de Datos sino mediante la lógica de una aplicación desarrollada en algún lenguaje, que tiene acceso a la información, ya que el exceso de reglas puede disminuir el rendimiento del RDBMS en los procesos de inserción y modificación de información.

En Sybase y MS SQL Server las reglas se pueden crear como objetos independientes que se vinculan a distintas tablas. En cambio en Oracle y PostgreSQL no son objetos independientes y sólo pueden ser definidas como restricciones que afecta a una sola tabla.

	MySQL	PostgreSQL	Oracle	Sybase	MS SQL Server
Soporta definición de reglas	No	Si	Si	Si	Si

Tabla 2-7 Los RDBMS y el soporte para reglas

En las siguientes tablas describo la sintaxis para crear reglas dependiendo el RDBMS:

Sintaxis para Sybase y MS SQL Server	CREATE RULE <*nombre*_regla> AS <*condicion*>
Ejemplo	CREATE RULE regla_salario AS @sueldo <= 10000

Tabla 2-8 Sintaxis creación de reglas Sybase y Ms SQL Server

Sintaxis para Oracle y PostgreSQL	ALTER TABLE <*nombre_tabla*> ADD CONSTRAINT <*nombre_restriccion*> CHECK <condicion>
Ejemplo	ALTER TABLE empleado ADD CONSTRAINT regla_salario CHECK @sueldo <= 10000

Tabla 2-9 Sintaxis creación de reglas Oracle y PostgreSQL

2.5 Defaults

Los defaults establecen que valor será registrado de manera predeterminada para una columna, en caso de que no se especifique al momento de introducir los datos.
En algunos RDBMS los defaults son objetos que se pueden emplear en diferentes tablas, mientras que en los demás, están ligados a la definición de la tabla. Al igual que con las reglas, la funcionalidad de los defaults pueden implementarse utilizando la lógica de la aplicación.
Nuevamente en Sybase y MS SQL Server es posible definir un DEFAULT como un objeto independiente que se puede vincular a varias campos de una o más tablas, mientras que en Oracle, MySQL y PostgreSQL los defaults están ligados a un solo campo.

	MySQL	PostgreSQL	Oracle	Sybase	MS SQL Server
Soporta definición de defaults	Si	Si	Si	Si	Si

Tabla 2-7 Los RDBMS y el soporte para defaults

En Todos los RDBMS, los valores predeterminados se pueden especificar al momento de generar una tabla, o en algunos casos es posible modificar la tabla para agregar estos valores predeterminados:

Sintaxis	<*nombre_campo*> <*tipo_dato*> DEFAULT <val_*predeterminado*>
Ejemplo al crear una tabla	CREATE TABLE cliente (id_cliente NUMERIC(10),**fecha_afiliacion DATE DEFAULT SYSDATE**)
Ejemplo al modificar una tabla	ALTER TABLE cliente add **fecha_afiliacion DATE DEFAULT SYSDATE**

Tabla 2-8 Sintaxis definición de defaults

2.6 Llaves e Índices

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

Es un error frecuente confundir entre índices y llaves, quizá sea el hecho de que al crear una llave primaria se genera un índice, lo cual no sucede para una llave foránea. Hay que tener presente que los índices tienen como función acelerar el proceso de recuperación de la información.

Tanto una llave primaria como una foránea establecen restricciones sobre el valor que pueda tener una columna. Las restricciones pueden tener un nombre asignado por el usuario o si este no se especifica, entonces el RDBMS será el encargado de asignarle un nombre interno a dicha restricción.

2.6.1 Llaves primarias

Una llave primaria permite identificar de manera única un registro dentro de una tabla. Al crear una llave de este tipo se genera automáticamente un índice de valores únicos, por lo que los valores de los campos que involucra la llave primaria no se pueden repetir ni ser nulos. Existen varias formas de declarar una llave primaria:

Sintaxis	CONSTRAINT <*nombre_llaveprimaria*> **PRIMARY KEY (**<*nombre_campo1*>,<*nombre_campo2*>,…,<*nombre_campon*>**)**
Ejemplo al crear una tabla con un campo como llave primaria	CREATE TABLE pais (id_pais numeric(3) NOT NULL **PRIMARY KEY**, nombre varchar(100))
Ejemplo al crear una tabla con dos campos como llave primaria	CREATE TABLE pedido (id_cliente numeric(10) NOT NULL, id_producto numeric(10) NOT NULL, fecha date, CONSTRAINT pedido_pk **PRIMARY KEY (id_cliente, id_producto)**)
Ejemplo al modificar una tabla	ALTER TABLE pedido **ADD CONSTRAINT pedido_pk PRIMARY KEY (id_cliente, id_producto)**

Tabla 2-9 Sintaxis creación de llave primaria

2.6.2 Llaves foráneas

Las llaves foráneas son atributos de una tabla que hacen referencia a la llave primaria de otra tabla. Estas llaves foráneas permiten establecer relaciones entre las distintas tablas que existen dentro de la Base de Datos. La mayoría de los RDBMS implementan restricciones (constraints) cuando se genera una llave foránea, de este modo asegura la integridad de la información almacenada en la Base de Datos.

Las versiones 3 y anteriores de MySQL no implementaban estas restricciones. Actualmente se puede decidir entre manejarlas o no manejarlas. Quizá en un futuro de manera predeterminada se manejara el soporte para llaves foráneas. Lo que implica la falta de este tipo de restricciones puede ser un problema de integridad en la información, pero por otro lado las restricciones se pueden implementar con lógica de aplicación en caso de que exista (Por ejemplo una aplicación escrita en Java puede realizar verificación de integridad) liberando así a la Base de Datos de esta tarea permitiendo que MySQL trabaje más rápido que cualquier otro RDBMS.

Dependiendo del tipo de información y de aplicación a desarrollar se puede sacrificar la característica a cambio de rapidez en respuesta.

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

Sintaxis	CONSTRAINT <*nombre_llaveforanea*> **FOREIGN KEY (**<*nombre_campo*>*)* REFERENCES <nombre_tabla_referenciada>(<nombre_llaveprimaria_referenciada>)
Ejemplo al crear una tabla	CREATE TABLE resultado (id_pais1 NUMBER(3) NOT NULL, id_pais2 NUMBER(3) NOT NULL, resultado VARCHAR2(15) NOT NULL, **CONSTRAINT pais_01_fk FOREIGN KEY (id_pais1) REFERENCES pais(id_pais),** **CONSTRAINT pais_02_fk FOREIGN KEY (id_pais2) REFERENCES pais(id_pais)**)
Ejemplo al modificar una tabla	ALTER TABLE resultado ADD CONSTRAINT pais_01_fk **FOREIGN KEY (id_pais1) REFERENCES pais(id_pais)**; ALTER TABLE resultado ADD CONSTRAINT pais_02_fk FOREIGN KEY (id_pais2) REFERENCES pais(id_pais);

Tabla 2-10 Sintaxis creación llave foránea

2.6.3 Índices

Un índice es una estructura de almacenamiento físico que ocupa un espacio. Los índices ayudan al Servidor SQL a localizar datos y son transparentes para el usuario.

El propósito principal de un índice es proporcionar un acceso más rápido a los datos, aunque en algunos casos su propósito es asegurar que el contenido de un campo sea único.

Abusar del empleo de índices puede llevar a que se degrade el tiempo de respuesta del servidor en lugar de mejorarlo, esto se debe a que en operaciones que involucran inserción, modificación o eliminación de datos, los índices deben de ser actualizados lo cual puede consumir un tiempo considerable.

Por lo general se suele crear un índice únicamente cuando una columna es empleada frecuentemente en una cláusula WHERE y la tabla tiene un tamaño considerable.

La sintaxis para creación y eliminación de índices se muestra en las siguientes tablas:

Sintaxis	CREATE [UNIQUE] INDEX <*nombre_índice*>
Ejemplo	CREATE INDEX idx_empleado ON empleado(fecha _ contratación)

Tabla 2-11 Sintaxis creación de índices

Sintaxis	DROP INDEX <*nombre_índice*>
Ejemplo	DROP INDEX idx_empleado

Tabla 2-12 Sintaxis eliminación de índices

2.7 Manipulación de datos

La mayor parte del trabajo con SQL gira en torno a cuatro comandos:

- SELECT: Permite seleccionar (recuperar) información de una tabla
- INSERT: Permite agregar información a una tabla
- DELETE: Permite eliminar información de una tabla
- UPDATE: Permite actualizar información que existe en una tabla

Para el empleo de cualquiera de los comandos mencionados anteriormente es indispensable tomar en cuenta dos puntos:

- Para expresar un valor de tipo alfanumérico o fecha, es requisito entrecomillarlo con comillas simples
- Todo valor que no se especifique entre comillas simples, será interpretado como tipo de dato numérico

2.7.1 Algebra relacional

SQL se basa en el álgebra relacional, por ello es importante conocer las operaciones del álgebra relacional y su relación con SQL.

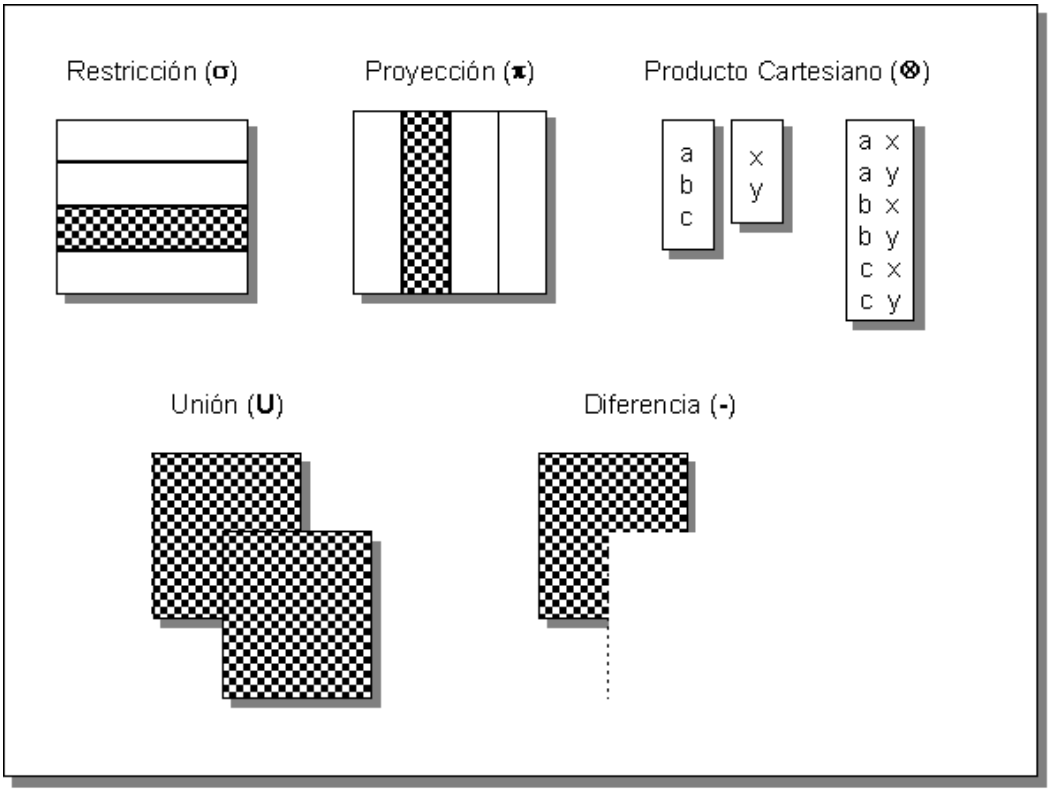


Fig. 2-2 Algebra relacional

Selección (Restricción)

La operación de selección genera un subconjunto de los renglones de una tabla, con base en un criterio (restricción) establecido.
Esta operación del álgebra relacional es realizada por la cláusula WHERE de SQL.

Proyección

La proyección selecciona y genera un subconjunto con los atributos (columnas) indicados de una tabla. También es conocida como operación vertical.
Esta operación es realizada por la cláusula SELECT de SQL.

Unión

La operación unión realiza la misma acción que en el álgebra de conjuntos, es decir {1,4,5,10} U {1,4,3,9} = {1,3,4,5,9,10}.
Esta operación se realiza con la cláusula UNION de SQL.

Producto Cartesiano

El producto cartesiano es el producto cruz entre 2 tablas:

{a, b} X {1, 2} = {a1, a2, b1, b2}

El resultado es la combinación de cada renglón de una tabla con cada renglón de la otra tabla.

En SQL esta operación se lleva a cabo cuando se ocupa la cláusula FROM especificando 2 o más tablas, y no se especifica una restricción que indique la relación entre las tablas.

Join

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

2.7.2 Selección de datos

Las tablas dentro de una Base de Datos son las estructuras que tienen almacenada la información en forma de registros. Para poder recuperar esa información almacenada, se requiere del comando SELECT de SQL.

El comando SELECT es sumamente útil, ya que a través de él es posible realizar desde una consulta simple que sólo involucra una tabla, hasta una consulta compleja donde intervienen dos o más tablas, varias condiciones, agrupaciones de datos y ordenamientos.

Sintaxis	SELECT {* | [DISTINCT] <*campo*>, <*campo*> … } FROM <*nombre_tabla*>, [<*nombre_tabla*>] [WHERE <*condición*>] [GROUP BY <*campo*>, <*campo*>,…] [HAVING <*condición*>] [ORDER BY <*campo*> [ASC|DESC], <*campo*> [ASC|DESC], …]
Seleccionar el nombre del empleado de la tabla empleado y el nombre de departamento de la tabla departamento.	**SELECT empleado.nombre, departamento.nombre** FROM empleado, departamento WHERE departamento.id_departamento = empleado.id_departamento
Seleccionar todos los campos de la tabla empleado.	**SELECT *** FROM empleado
Seleccionar todos los campos de la tabla empleado y el nombre del departamento de la tabla departamento.	**SELECT empleado.*, departamento.nombre** FROM empleado, departamento WHERE departamento.id_departamento = empleado.id_departamento

Tabla 2-13 Sintaxis selección de datos

Lo que se puede apreciar en la estructura de la instrucción SELECT, es que nunca debe faltar ni la palabra SELECT, ni FROM. Todos los demás elementos son opcionales.

Comando SELECT

Este comando indica que la instrucción a ejecutar es una consulta a la Base de Datos. SELECT permite indicar el nombre de los campos que queremos mostrar en la consulta. En caso de querer mostrar todos los campos de una tabla se emplea el comodín asterisco: *.

Cuando se realiza una consulta que involucra dos o más tablas, al nombre de cada campo se le antepone el de la tabla a la que pertenece.

Sintaxis	<*nombre_tabla*>.<*nombre_campo*>
Ejemplo	SELECT **empleado.nombre, departamento.nombre** FROM empleado, departamento

Tabla 2-14 Sintaxis selección de datos multi-tablas

Uso de alias para columnas

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

Sintaxis	<*nombre_campo*> AS <*alias*>
Ejemplo	SELECT **nombre AS nombre_empleado,** FROM empleado

Tabla 2-15 Sintaxis selección de datos empleando alias para columnas

Uso de alias para tablas
Cuando una consulta involucra más de una tabla, es necesario especificar de cual tabla es el campo que se desea mostrar. Esto lo podemos ver en el ejemplo c), que involucra a las tablas empleado y departamento. Es frecuente el uso de seudónimos para las tablas, con la finalidad de simplificar la escritura de la consulta. El seudónimo se coloca inmediatamente después del nombre de la tabla.

Sintaxis	<*tabla*> [seudónimo]
Ejemplo	SELECT **e**.nombre, **d**.nombre FROM **empleado e, departamento d**

Tabla 2-16 Sintaxis selección de datos empleando alias para tablas

Cláusula FROM
La cláusula FROM sirve para indicar las tablas de las cuales se desea mostrar la información.

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

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

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

Las expresiones más frecuentes son las que involucran una comparación entre dos elementos, a continuación se enlistan los operadores utilizados para realizar dicha comparación.

Comparación	Operador	Ejemplo
Igualdad	=	empleado.id_departamento = departamento.id_departamento
Desigualdad	<> ó ¡=	nombre_cargo ¡= 'Director'
Mayor que	>	sueldo < 15000
Menor que	<	edad > 35
Mayor o igual que	>=	sueldo >=15000
Menor o igual que	<=	edad <= 35
Similar a	LIKE	nombre_empleado like 'Al%' (% es un comodín)
Es	IS	edad IS NULL
No es	IS NOT	edad IS NOT NULL

Tabla 2-17 Operadores de comparación

La comparación de similitud que se hace mediante el uso de la cláusula LIKE, requiere de incluir comodines, que sustituyan uno o varios caracteres. El signo "%" representa cero o más caracteres y el signo "-" representa un carácter.

En SQL es posible abreviar la forma de escribir ciertas condiciones:

Expresión	Operador	Quedando de la siguiente manera:
Sueldo >= 10000 AND sueldo <=15000	**BETWEEN**	Sueldo between 10000 and 15000
Nombre_cargo = 'Gerente' OR nombre_cargo = 'Presidente' OR nombre_cargo = 'Vicepresidente' OR nombre_cargo = 'Director'	**IN**	nombre_cargo in ('Gerente', 'Presidente', 'Vicepresidente', 'Director')
Nombre_cargo ¡= 'Jefe de departamento' AND nombre_cargo ¡= 'Vendedor'	**NOT IN**	nombre_cargo not in ('Jefe de departamento', 'Vendedor')

Tabla 2-18 Operadores de condición

Cláusula GROUP BY

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

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

Tabla 2-19 Funciones de agregado

Algunos ejemplos de la utilización de las funciones de agregados serian:

Descripción	Sentencia
Mostrar la clave de departamento y el dinero total empleado para pagar a los empleados de dicho departamento.	SELECT id_departamento, SUM(sueldo) FROM empleado GROUP BY id_departamento
Mostrar la edad promedio por cada clave de departamento.	SELECT id_departamento, AVG(edad) FROM empleado GROUP BY id_departamento
Mostrar el nombre de departamento y el dinero total empleado para pagar a los empleados de dicho departamento.	SELECT d.nombre, SUM(e.sueldo) FROM empleado e, departamento d WHERE e.id_departamento = d.id_departamento GROUP BY d.nombre

Tabla 2-20 Uso de funciones de agregado

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

Ejemplo	Sentencia
Mostrar la clave de departamento y sueldo promedio de sus empleados, de aquellos departamentos cuyo salario promedio sea mayor que 12000	SELECT id_departamento, AVG(sueldo) FROM empleado GROUP BY id_departamento HAVING AVG(sueldo)>12000

Tabla 2-21 Uso cláusula HAVING

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

Ejemplos	Sentencia
Mostrar todos los datos de los empleados ordenados por nombre de manera descendente.	SELECT * FROM empleado ORDER BY nombre DESC
Mostrar clave, nombre del empleado y salario ordenados por salario.	SELECT id_empleado, nombre, sueldo FROM empleado ORDER BY 3

Tabla 2-22 Uso cláusula ORDER BY

Uso de Joins
Un join se emplea para definir la relación que existe entre dos tablas. El INNER JOIN es una cláusula que permite definir estas relaciones. Este tipo de join puede expresarse de otra manera utilizando restricciones en la sección del WHERE en una consulta.

Ejemplo con INNER JOIN	SELECT e.nombre, d.nombre FROM empleado e **INNER JOIN** departamento d **ON** e.id_departamento = d.id_departamento
Ejemplo con WHERE	SELECT e.nombre, d.nombre FROM empleado e, departamento d WHERE e.id_departamento = d.id_departamento

Tabla 2-23 Uso de Joins

El resultado de las dos consultas mostradas es el mismo. Lo único que varía es la forma de especificar la relación entre las tablas.

Uso de la Unión
Permite realizar la unión del resultado de dos consultas. El proceso de UNION elimina los registros duplicados, aunque algunos RDBMS proporcionan una cláusula que permite mostrar todos los registros resultantes de la unión.

Sintaxis	*<instrucción SELECT_1>* **UNION** *<instrucción SELECT_2>*
Ejemplo	SELECT MAX(sueldo) FROM empleado **UNION** SELECT MIN(sueldo) FROM empleado

Tabla 2-24 Uso de la Unión

2.7.3 Inserción de datos

A través del comando INSERT de SQL, se introduce información a una tabla.

Sintaxis	INSERT INTO <*tabla*> [(<*nombreCampo1*>, <*nombreCampo2*>, <*nombreCampo3*> *…*)] {VALUES (<*valorCampo1*>, <*valorCampo2*>, <*valorCampo3*> …) | <*Expresión select*> }
Ejemplo	INSERT INTO empleado (id_empleado id_departamento, id_cargo, nombre, edad, sexo, sueldo, fecha_contratacion) VALUES (20, 2, 1, 'Lorena Aguilar', 'M', 7000.00, '2002-04-26')

Tabla 2-25 Sintaxis inserción de datos

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

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

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

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

2.7.4 Eliminación de datos

El comando **DELETE** elimina registros de la tabla indicada con la posibilidad de indicar un criterio, en caso de omitirlo, se eliminan todos los registros de la tabla.

Sintaxis	DELETE FROM <*nombre_tabla*>[WHERE <*condición*>]
Ejemplo	DELETE FROM empleados WHERE id_empleado=7

Tabla 2-26 Sintaxis eliminación de datos

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

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

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

Cláusula WHERE
La cláusula WHERE permite delimitar el conjunto de registros a eliminar, si no se especifica esta condición, se eliminaran todos los registros que contenga la tabla especificada. Para mayor detalle de las condiciones consulte el contenido de la sección 2.7.2.

2.7.5 Actualización de datos

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

Sintaxis	UPDATE <*nombre_tabla*> SET <*campo1*> = <*valor1*>, <*campo2*> = <*valor2*>, …. [WHERE <*condición*>]
Ejemplo **Actualizar el salario a 27,000 pesos y edad a 27 años del empleado con clave 12.**	**UPDATE empleado** SET sueldo = 27000, edad = 27 WHERE id_empleado = 12

Tabla 2-27 Sintaxis actualización de datos

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

Cláusula SET
Esta cláusula permite especificar los campos que se desean modificar y su nuevo valor. La cláusula se coloca una sola vez aunque sean varios campos los que se deseen modificar.

Cláusula WHERE
Esta cláusula permite especificar un criterio para delimitar el conjunto de registros a modificar. Para mayor detalle de las condiciones consulte el contenido de la sección 2.7.2.

2.8 Vistas

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

Sintaxis	CREATE VIEW <*nombre_vista*> AS <*instrucción SELECT*>
Ejemplo	CREATE VIEW empleados_h AS SELECT * FROM empleados WHERE sexo = 'H'

Tabla 2-28 Sintaxis creación de vistas

Sintaxis	DROP VIEW <*nombre_vista*>
Ejemplo	DROP VIEW empleados_h

Tabla 2-29 Sintaxis eliminación de vistas

Las vistas pueden ser utilizadas como si fueran tablas, pero con ciertas restricciones Algunas consideraciones que se deben tomar en cuenta al implementar vistas son:

La instrucción SELECT que define una vista puede:

- Incluir criterios de selección (cláusula WHERE)
- Usar funciones de agrupamiento (COUNT, AVG, SUM,…)
- Usar columnas calculadas
- Consultar más de una tabla
- Consultar otra vista

La instrucción SELECT que define una vista NO puede utilizar la cláusula ORDER BY. Sin embargo, la cláusula ORDER BY se puede aplicar a la vista una vez creada, siendo consultada como cualquier tabla.

2.9 Transacciones

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

	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server
Soporta transacciones	Si	Si	Si	Si	Si

Tabla 2-30 Los RDBMS y el soporte para transacciones

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

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

2.9.1 Commit y Rollback

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

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

Ejemplo	BEGIN; UPDATE cuenta SET saldo = saldo - 1000.00 WHERE cliente = 'Fernanda'; UPDATE cuenta SET saldo = saldo + 1000.00 WHERE cliente = 'Alfredo'; COMMIT;

Tabla 2-31 Uso de transacciones

En el ejemplo anterior, se muestra las operaciones necesarias para realizar una transacción monetaria de $1,000 pesos de la cuenta de Fernanda a la cuenta de Alfredo. Si por alguna razón alguna de las dos instrucciones fallara todo el proceso sería cancelado.
La cláusula COMMIT, realiza los cambios de manera permanente. Si se diera el caso de requerir cancelar la operación, en lugar de emplear la cláusula COMMIT se debería emplear la cláusula ROLLBACK.

2.10 Procedimientos almacenados

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

Cada RDBMS tiene procedimientos almacenados predeterminados, para simplificar las tareas de administración de Base de Datos, la nomenclatura para identificarlos en la mayoría de los RBDMS es "**sp**_<nombre_de funcion>", sp del acrónimo en inglés store procedure.

	MySQL	PostgreSQL	Oracle	Sybase	Ms SQL Server
Soporta Procedimientos Almacenados	No	Si	Si	Si	Si

Tabla 2-32 Los RDBMS y el soporte para procedimientos almacenados

El lenguaje que se emplea para programar los procedimientos almacenados, varía de un RDBMS a otro, y existen algunos que permiten programar en más de un lenguaje.
Para crear procedimientos almacenados se utilizan las siguientes sentencias:

RDBMS	Sintaxis	Ejemplo
Oracle	CREATE PROCEDURE <*nombre_procedimiento*> [(<*nombre_parámetro*> [IN | OUT] <*tipo_de_dato*>)] AS BEGIN <*instrucciones SQL*> END	CREATE PROCEDURE nombre_empleados AS BEGIN SELECT nombre FROM empleados ORDER BY nombre END
Sybase y MS SQL Server	CREATE PROCEDURE <*nombre_procedimiento*> [(@<*nombre_parámetro*> <*tipo_de_dato*>)] AS <*instrucciones SQL*> RETURN	CREATE PROCEDURE nombres_empleados AS SELECT nombre FROM empleado ORDER BY nombre RETURN

Tabla 2-33 Sintaxis creación de procedimientos almacenados

Para ejecutar los procedimientos almacenados la sintaxis es la siguiente:

RDBMS	Sintaxis	Ejemplo
Oracle	CALL <nombre_procedimiento> [<parámetro>,…]	CALL nombre_empleados
Sybase y MS SQL Server	EXEC <nombre_procedimiento> [<parámetro>, …]	EXEC nombre_empleados

Tabla 2-34 Sintaxis ejecución de procedimientos almacenados

Para eliminar un procedimiento almacenado, se dispone de la instrucción DROP PROCEDURE.

Sintaxis	DROP PROCEDURE <*nombre_procedimiento*>
Ejemplo	DROP PROCEDURE nombre_empleados

Tabla 2-35 Sintaxis eliminación de procedimientos almacenados

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

RDBMS	Creación	Ejecución

Oracle	CREATE PROCEDURE
	sueldo_empleado
	(sueldo IN number(8,2))
	AS
	BEGIN
	SELECT nombre, sueldo
	FROM empleado
	WHERE sueldo <= sueldo
	ORDER BY nombre;
	END;
		call sueldo_empleado 10000

Sybase y MS SQL Server	CREATE PROCEDURE sueldo_empleado @sueldo numeric(8,2) AS SELECT nombre, sueldo FROM empleado WHERE sueldo <= @sueldo ORDER BY nombre RETURN	exec sueldo_empleado 10000

Tabla 2-36 El uso de parámetros en los procedimientos almacenados

Administración de Bases de Datos

Los procedimientos de administración de las Base de Datos dependen del fabricante y la versión del RDBMS por tal motivo en este capítulo solo me enfocaré al manejador de Base de Datos SYBASE Adaptive Server Enterprise 15.0 (en adelante ASE).
Con los temas de este capítulo trato de describir las funciones básicas del Administrador de Base de Datos las cuales incluyen:

- Instalar, configurar y actualizar el RDBMS
- Decidir la estructura de almacenamiento: Monitorea constantemente el crecimiento de la Base de Datos y administra los medios de almacenamiento de la Base de Datos
- Brindar servicio tanto a los usuarios finales como a los programadores y diseñadores de aplicaciones para garantizar la existencia de los datos necesarios y requeridos dentro de la Base de Datos
- Definir los controles de acceso a la Base de Datos: Involucra la asignación de privilegios a los usuarios para conservar integridad y calidad de los datos almacenados dentro de la Base de Datos
- Definir una estrategia de respaldo y recuperación, para asegurar una completa recuperación ante posibles fallas del sistema o pérdida de información
- Mantener la Base de Datos en un estado óptimo de operación: Monitorear el rendimiento de la Base de Datos para brindar un servicio de buena calidad en un tiempo aceptable

3.1 Instalación y actualización de ASE

Debido a que ASE es compatible con diferentes sistemas operativos no se puede unificar el procedimiento de instalación y/o actualización, se deben descargar las guías correspondientes dependiendo el sistema operativo y la versión de ASE que requiera, las cuales están disponibles en el portal de ayuda SYBASE:

http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc35130_0400/html/SySearch_rb/title.htm

3.2 Ambiente de Trabajo

Una vez instalado ASE podemos realizar todas las tareas de administración, así como consultar, crear, modificar y eliminar Bases de Datos y su contenido, para poder realizar todas estas actividades es necesario ingresar a ASE por medio de la utilidad isql, que se encuentra almacenada en el directorio:

Plataforma	Directorio	Programa
Unix	$SYBASE/$SYBASE_OCS/bin	*isql.sh*
Windows	%SYBASE%\%SYBASE_OCS%\bin	*isql.exe*

Tabla 3-1 Directorios de isql

La utilidad isql puede ser ejecutada desde línea de comando de sistema operativo o bien en su forma grafica, en este capítulo solo describiré el procedimiento para ejecutar isql por línea de comando.

Para iniciar isql desde línea de comando, se debe teclear en el prompt de sistema operativo el siguiente comando:

Sintaxis	Ejemplo
isql –U<*nombre_usuario*>	isql -Umparedes

Tabla 3-2 Sintaxis de isql

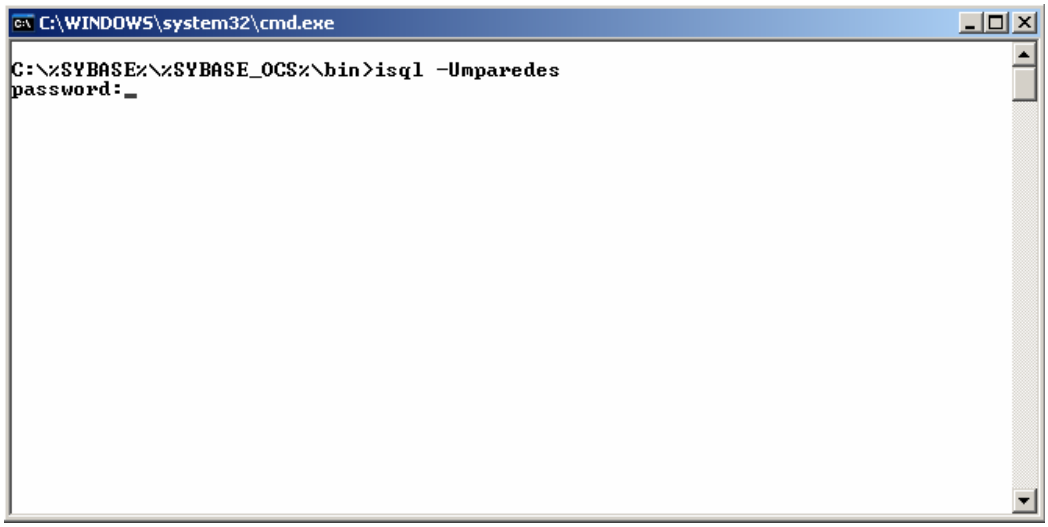


Ilustración 3-1 Ejecución isql en windows

Automáticamente ASE solicitara la contraseña, si la contraseña es correcta se desplegara el prompt de ASE, a partir de ese momento se pueden ejecutar sentencias de SQL para realizar las tareas de administración y todas las actividades antes mencionadas.

Las sentencias que se ingresen en isql pueden ocupar varias líneas, isql no procesa las sentencias hasta que se teclee la palabra “go” . Por ejemplo:

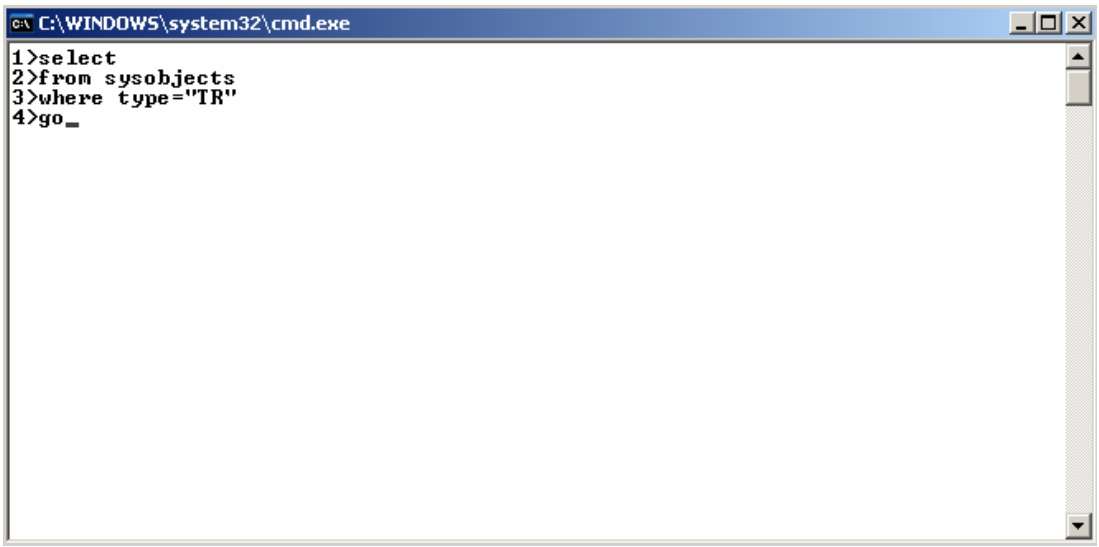


Ilustración 3-2 Ejecución de sentencias en isql

3.2.1 Bases de Datos de sistema

Para que ASE funcione correctamente y pueda administrar las Bases de Datos de usuario, son creadas durante la instalación de ASE las siguientes Bases de Datos de sistema:

Master: La Base de Datos master controla la operación de ASE y almacena información acerca de todas las Bases de Datos de usuario y sus medios de almacenamiento. A continuación enlisto cada una de las tablas contenidas en esta Base de Datos:

Información almacenada	**Tabla de sistema**
Cuentas de usuario	syslogins
Cuentas remotas de usuario	sysremotelogins
Servidores remotos con los que puede interactuar	sysservers
Procesos	sysprocesses
Variables de ambiente configurables	sysconfigures
Mensajes de error de sistema	sysmessages
Bases de datos creadas en ASE	sysdatabases
Espacio de almacenamiento ocupado por cada Base de Datos	sysusages
Cintas y discos disponibles en el sistema	sysdevices
Bloqueos activos	syslocks
Grupo de caracteres	syscharsets
Lenguajes	syslanguages
Usuarios con roles del servidor	sysloginroles
Roles del servidor	syssrvroles
Servidores ASE que están en linea	sysengines

Tabla 3-3 Tablas de sistema

Model: Esta Base de Datos es una plantilla o prototipo, para las nuevas Bases de Datos de usuario. Cada vez que un usuario crea una Base de Datos ASE realiza una copia de la Base de Datos model y extiende el tamaño de la nueva Base de Datos con el tamaño que se especifico en el comando de creación. Una Base de Datos nueva no puede ser de menor tamaño que la Base de Datos model.

La Base de Datos model contiene las tablas de sistema requeridas para cada Base de Datos de usuario. Se puede modificar la Base de Datos model para personalizar la estructura de las nuevas Bases de Datos.

Sybsystemprocs: Los procedimientos de sistema Sybase son almacenados en la Base de Datos sybsystemprocs. Cuando un usuario en cualquier Base de Datos ejecuta un procedimiento almacenado, ASE primero verifica si el procedimiento almacenado existe en esa Base de Datos. Si no encuentra el procedimiento almacenado, ASE consulta si existe en la Base de Datos sybsystemprocs. Si el procedimiento no se encuentra en sybsystemprocs, por ultimo ASE consulta la Base de Datos master.

Tempdb: ASE tiene una Base de Datos temporal, tempdb. Esta proporciona un área de almacenamiento para tablas temporales y otras necesidades del proceso de almacenamiento temporal. El espacio en tempdb es compartido entre todos los usuarios de todas las Bases de Datos en el servidor. La mayoría de las tablas creadas en esta Base de Datos son eliminadas al terminar la sesión, las tablas que son creadas con el prefijo “tempdb” permanecen en la Base de Datos hasta que son borradas por su dueño o bien al reiniciar el servidor ASE.

Cada vez que el servidor ASE es reiniciado, la Base de Datos model es copiada en la Base de Datos tempdb, lo cual limpia la Base de Datos, eliminando así todas las tablas temporales.

Opcionalmente se pueden crear otras Bases de Datos estándar como:

Sybsecurity: La Base de Datos sybsecurity contiene la auditoria de sistema para ASE, Esta consiste de:

- Las tablas de sistema, sysaudits_01, sysaudits_02, ... sysaudits_08, las cuales contienen los rastros de auditoría
- La tabla sysauditoptions, la cual contiene renglones describiendo las opciones globales de auditoría
- Todas las otras tablas de sistema estándar que son derivadas desde la Base de Datos model

Pubs2 y Pubs3: Estas Bases de Datos son proporcionadas como una herramientas de aprendizaje para ASE. La Base de Datos ejemplo pubs2 es utilizada por muchos de los ejemplos en la documentación de ASE, excepto por los ejemplos en donde se utiliza la Base de Datos pubs3.

3.2.2 Log de Bases de Datos

ASE utiliza transacciones para realizar cambios en la Base de Datos. Las transacciones son unidades de trabajo de ASE. Una transacción consiste de una o más sentencias de SQL que se ejecutan (exitosamente o fallan) como una unidad.

Cada Base de Datos tiene su propio log (bitácora) de transacciones, que es la tabla de sistema syslogs. El log de transacciones automáticamente graba cada una de las transacciones utilizada por cada usuario de la Base de Datos.
Cuando un usuario ejecuta una sentencia que modificara la Base de Datos, ASE escribe los cambios en el log. Después que todos los cambios para una sentencia han sido grabados en el log, son escritos en memoria cache mediante la copia de página de datos.
La página de datos permanece en memoria cache hasta que la memoria es necesitada por otra página. En ese momento es escrita a disco, liberando así la memoria cache.

Si cualquier sentencia fallara para completar la transacción, ASE reversa todos los cambios hechos por la transacción. ASE escribe en el log la línea: “end transaction” al final de cada transacción y guarda el estatus (fallo).

3.3 Dispositivo de Base de Datos

Un dispositivo de Base de Datos puede ser cualquier espacio en disco duro, tales como particiones, file system o bien archivos de sistema. En un dispositivo se almacenan Bases de Datos y sus objetos.

Cada dispositivo de Base de Datos debe ser preparado antes que pueda ser utilizado para el almacenamiento de una Base de Datos, este proceso es conocido como inicialización.
Después de que un dispositivo de Base de Datos ha sido inicializado, este puede ser asignado a la lista predeterminada de dispositivos para ser utilizado para crear o extender una Base de Datos y para almacenar logs de transacciones de la Base de Datos.

El administrador de Base de Datos inicializa un nuevo dispositivo con el comando **disk init**, con el cual se puede:

- Asignar el nombre de dispositivo de Base de Datos al disco físico especificado o el archivo de sistema operativo
- Listar los nuevos dispositivos en la tabla *sysdevices*
- Preparar los dispositivos para el almacenamiento de Base de Datos

Sintaxis	Ejemplo UNIX	Ejemplo Windows
disk init name = "nombre_dispositivo", physname = "ruta _ física", size = “tamaño en Kilobytes (K), Megabytes (M) o Gigabytes (G)”	disk init name= "disp_datos1", physname ="/dev/rxy1a", size = "10M"	disk init name= "disp_datos1", physname =“d:\devices\userdisk.dat", size = "10M"

Tabla 3-4 Sintaxis inicialización dispositivos

Antes de ejecutar el comando **disk init** se debe:

- Respaldar la Base de Datos master
- Verificar el espacio disponible en disco
- Asegurar que el archivo o dispositivo no haya sido inicializado
- Asegurar que la cuenta de administrador sybase tenga permisos de escritura sobre ese dispositivo
- Verificar la cantidad de dispositivos creados previamente, el máximo número de dispositivos configurables es de 255

Después de ejecutar el comando **disk init** se debe respaldar la Base de Datos master.

Cuando se crea una Base de Datos con el comando **create database**, se especifica el dispositivo sobre el cual la Base de Datos deberá ser creada, si no se especifica un dispositivo, la Base de Datos será creada sobre algún dispositivo definido como predeterminado.

Una vez que un dispositivo es inicializado, se puede asignar dicho dispositivo como predeterminado usando el comando **sp_diskdefault.**

Sintaxis	Ejemplo
sp_diskdefault <nombre_dispositivo>, defaulton	sp_diskdefault disp_datos1, defaulton

Tabla 3-5 Sintaxis configuración de dispositivos predeterminados

El comando **sp_diskdefault** marca los dispositivos como dispositivos predeterminados en la tabla sysdevices.

Cuando ASE es instalado, el dispositivo master es un dispositivo predeterminado, es recomendable configurar el dispositivo master para que deje de ser predeterminado y así evitar que las Bases de Datos de usuario sean almacenadas junto a las Bases de Datos del sistema. Para realizar esta configuración ejecutar el siguiente comando:

exec sp_diskdefault master, defaultoff

Se pueden definir varios dispositivos predeterminados, son utilizados en orden de alfabético, cuando el primer dispositivo se ha usado completamente, el segundo dispositivo es utilizado y así sucesivamente.

Para eliminar un dispositivo se debe tomar en cuenta que no se puede eliminar un dispositivo que esta almacenando una Base de Datos, primero se debe eliminar la Base de Datos y después eliminar el dispositivo. El comando para eliminar dispositivos es **sp_dropdevice.**

Sintaxis	Ejemplo
sp_dropdevice <*nombre_dispositivo*>	sp_dropdevice disp_datos1

Tabla 3-6 Sintaxis eliminación de dispositivos

Consultar información de un dispositivo

Cuando se ejecuta el comando **disk init**, un nuevo registro es agregado a la tabla sysdevices de la Base de Datos master.
Para obtener información de los dispositivos creados se puede utilizar el comando **select * from sysdevices** o bien el procedimiento **sp_helpdevice** que de igual forma consulta la tabla sysdevices. La información que es desplegada por estos comandos se ilustra a continuación:

```
device_name  physical_name  description
-----------  -------------  ----------------------------------------
master        d_master         special, default disk, physical disk, 30 MB

status      cntrltype    vdevno     vp_low   vpn_high
------      ----------   -------    ------   -------
3           0            0          0        10239
```

Ilustración 3-3 Información de dispositivos

Se deben considerar las siguientes mejores prácticas para la asignación de dispositivos:

- Usar nombres significativos
- Asegurar que las particiones especificadas realmente estén disponibles; siga las restricciones de la plataforma específica sobre que partes del disco pueden ser usadas
- Crear y respaldar scripts de comandos de asignación de los dispositivos
- Mantener una copia de la tabla sysdevices
- Mantener una copia de la asignación física y lógica

3.4 Base de Datos en ASE

Antes de crear una Base de Datos es necesario definir:

- El nombre de la Base de Datos
- El tamaño de la Base de Datos
- El dispositivo de Base de Datos en donde será creada
- Si un dispositivo de log es necesario y, si lo es, definir el tamaño y ubicación

Las consideraciones que se deben tener al definir el tamaño de una Base de Datos son:

- El tamaño mínimo permitido es de 2MB
- Se pude aumentar fácilmente su tamaño pero reducirlo no
  - Para reducir el tamaño de la Base de Datos es necesario eliminarla y crearla de nuevo copiando los datos con una utilidad especial de ASE
- Estimar el número y tamaño de:
  - Tablas
  - Índices
  - Log de transacciones

  Utilizar el comando sp_estspace para estimar el tamaño de las tablas y sus Índices

Las consideraciones que se deben tener al definir el tamaño del log de una Base de Datos son:

- El tamaño del Log depende de la actividad (tipo y cantidad de transacciones) y la frecuencia de los respaldos
- Todos los inserts, deletes, y updates son registrados
- El log es fácil de extender, imposible encoger

Algunas buenas prácticas al momento de crear una Base de Datos:

- Proyectar el crecimiento de la Base de Datos a cinco años
- Almacenar el log en un dispositivo diferente a la Base de Datos
- Asignar al log del 10-30% del tamaño global de la Base de Datos

El comando **create database** es utilizado para crear Bases de Datos.

Sintaxis	Ejemplos
create database <nombre_bd> on <nombre_dispositivo> = <tamaño> log on <nombre_dispositivo> = <tamaño>	(1) create database ventas (2) create database ventas on disp_datos1=5 (3) create database ventas on disp_datos1=5 log on disp_log1 = 2

Tabla 3-7 Sintaxis creación de Bases de Datos

Nota: El parámetro “tamaño” debe ser expresado en megabytes

Cuando una Base de Datos es creada se copia la estructura de la Base de Datos model y se agrega un nuevo registro en la Base de Datos master. La nueva Base de Datos es almacenada en un dispositivo previamente inicializado.

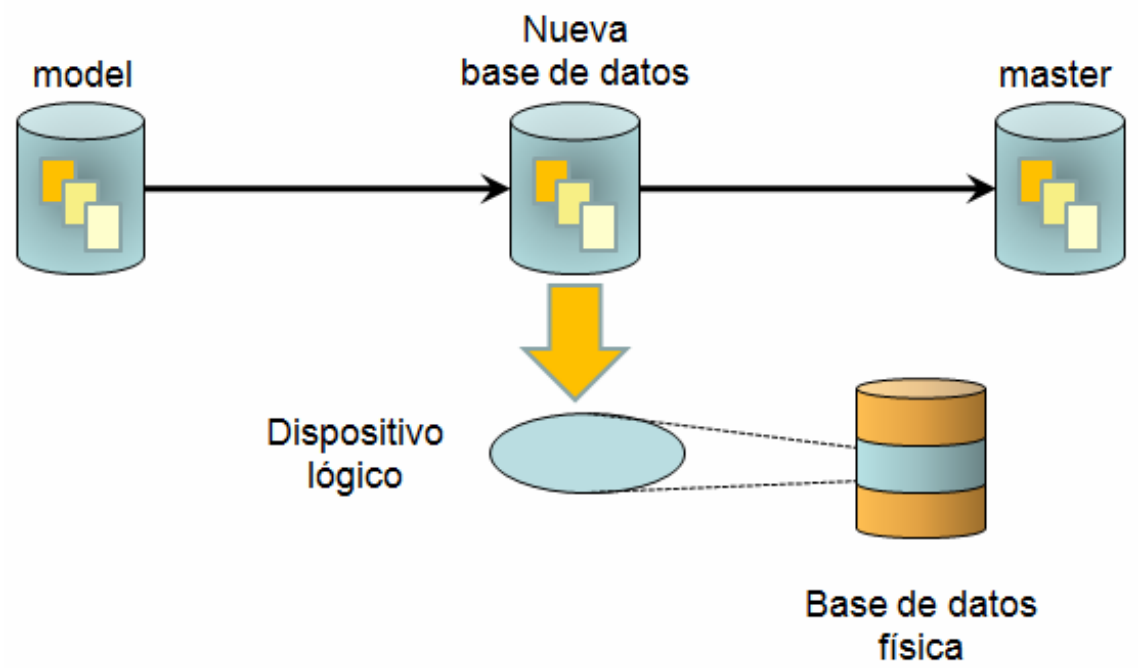


Fig. 3-1 Creación de Base de Datos

Desplegando información de las Bases de Datos

Cuando se crea una Base de Datos, las siguientes tablas de la Base de Datos master son afectadas:

- **sysdatabases:** Contiene un registro por cada Base de Datos creada, incluyendo el nombre de la Base de Datos y su propietario

- **sysusages:** Contiene un registro para cada fragmento del dispositivo para cada Base de Datos, indicando el tamaño y la dirección del disco del comienzo lógico para ese fragmento

Para desplegar esta información, se puede realizar una consulta sobres las tablas o bien ejecutar el comando sp_helpdb.

```
name           db_size  owner dbid created      status
-------------- -------- ----- ---- ------------ ------------------
master           3.0 MB sa       1 Jan 01, 1900 no options set
model            2.0 MB sa       3 Jan 01, 1900 no options set
mydata           4.0 MB sa       7 Aug 25, 1997 no options set
pubs2            2.0 MB sa       6 Aug 23, 1997 no options set
sybsecurity     20.0 MB sa       5 Aug 18, 1997 no options set
sybsystemprocs 10.0 MB sa       4 Aug 18, 1997 trunc log on chkpt
tempdb           2.0 MB sa       2 Aug 18, 1997 select into/
                                                bulkcopy/pllsort
```

Ilustración 3-4 Información de las Bases de Datos

Modificación del tamaño de una Base de Datos

El comando alter database, es utilizado para asignar espacio adicional a una Base de Datos sobre el mismo o diferentes dispositivos.

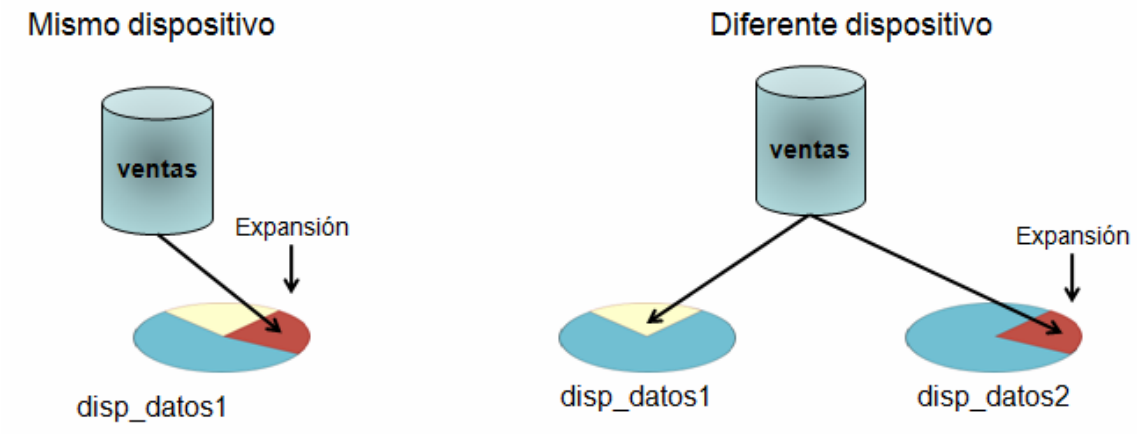


Fig. 3-2 Expansión de Base de Datos

Sintaxis	Ejemplos
alter database <nombre_bd> on <nombre_dispositivo> = <tamaño>	(1) alter database ventas (2) alter database ventas on disp_datos1 = 3

Tabla 3-8 Sintaxis modificación del tamaño de Base de Datos

Consideraciones al modificar el tamaño de una Base de Datos:

- El tamaño es especificado en megabytes.
- Si no se especifica el dispositivo ni el tamaño, se asignara el espacio mínimo predeterminado que es de 1MB y se almacenara en el dispositivo predeterminado de la Base de Datos.
- Si la Base de Datos y el log son almacenados en el mismo dispositivo, y no es especificado el tamaño ni el dispositivo para la expansión del log, el espacio agregado será utilizado por la Base de Datos.

Modificación del tamaño del log

El comando alter database, es utilizado para asignar espacio adicional al log de Base de Datos sobre el mismo o diferentes dispositivos. El tamaño debe ser especificado en megabytes.

Sintaxis	Ejemplos
alter database <nombre_basededatos> log on <nombre_dispositivo> = <tamaño>	alter database ventas log on disp_log1 = 1 alter database ventas log on disp_log2 = 4

Tabla 3-9 Sintaxis modificación del tamaño del log

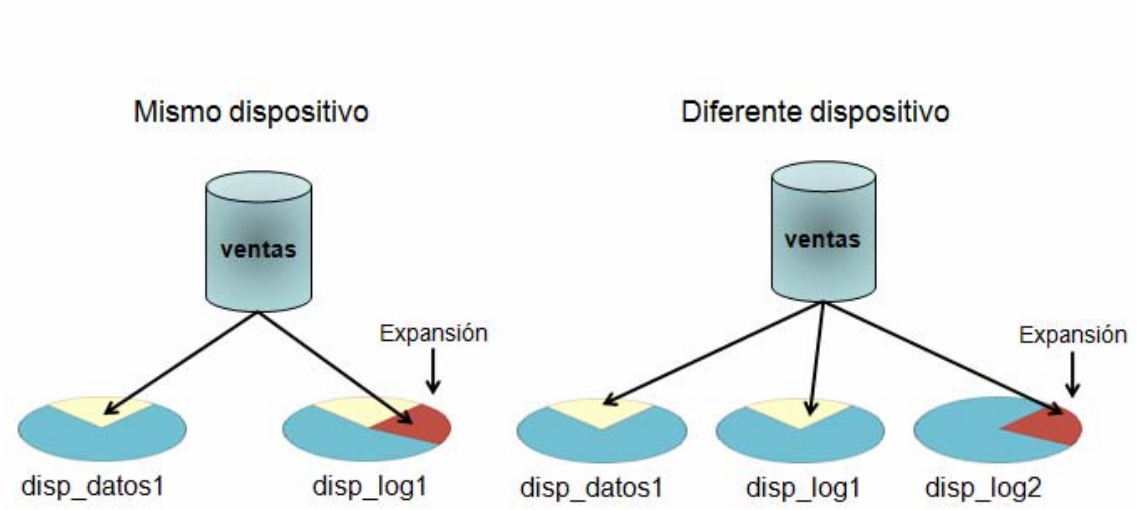


Fig. 3-3 Expansión del log

Eliminación Bases de Datos

El comando **drop database**, es utilizado para eliminar Bases de Datos, este elimina la Base de Datos y todos los objetos dentro de ella. Este comando:

- Libera el espacio en donde estaba almacenada la Base de Datos
- Elimina las referencias a la Base de Datos desde las tablas de sistema en la Base de Datos master
- Solo el dueño de Base de Datos puede eliminar la Base de Datos
- No se puede eliminar una Base de Datos que este siendo utilizada por un usuario
- Se pueden eliminar más de una Base de Datos en una sentencia

Sintaxis	Ejemplos
drop database <nombre_bd>, <nombre_bd>, …	drop database ventas drop database ventas, nomina

Tabla 3-10 Sintaxis Eliminación de Base de Datos

3.5 Control de Acceso

Para controlar los accesos a la Base de Datos ASE utiliza el Control de Acceso Discrecional DAC del acrónimo en inglés Discretionary Access Control.
DAC permite restringir el acceso a objetos y comandos basados en la identidad de los usuarios o miembros de un grupo. El control es “discrecional” porque un usuario con ciertos permisos de acceso, tales como dueño de objeto, puede elegir si transfiere estos permisos de acceso a otros usuarios.

Como todo RDBMS ASE utiliza el esquema de seguridad descrito en el capítulo I, el cual administra la seguridad en 3 niveles (Servidor, Base de datos y objetos).

Ahora bien para entender mejor el esquema de seguridad de ASE es necesario conocer algunos conceptos básicos:

Login: Para poder ingresar a ASE es necesario tener una cuenta dentro del sistema, esta cuenta de acceso es llamada login, debe tener asociados un identificador único y una contraseña.

Usuario de Base de Datos: Una vez que un login ha ingresado en ASE, es necesario contar con un usuario para poder realizar consultas y modificaciones dentro de una Base de Datos, a este usuario se le conoce como usuario de Base de Datos.

Grupo: Sirve para agrupar a los logines por sus características.

Rol: Es la agrupación de autorizaciones que permiten realizar determinadas actividades, los roles pueden ser asignados a un usuario o grupos de usuarios.

Objetos: Refiriéndonos a Bases de Datos un objeto puede ser una tabla, vista, comando, procedimiento almacenado, etc.

3.5.1 Privilegios del administrador de sistema sa

Cuando ASE es instalado, se crea el login administrador del sistema **sa** (system administrator), este login tiene las siguientes características:

- Ha sido asignado con tres roles de sistema (sa_role, sso_role, y oper_role)
- Ejecuta todos los comandos SQL
- Es propietario de la Base de Datos master
- Es tratado como Dueño de la Base de Datos (dbo) en todas las Bases de Datos
- Tiene acceso a todas las Bases de Datos y objetos de Base de Datos

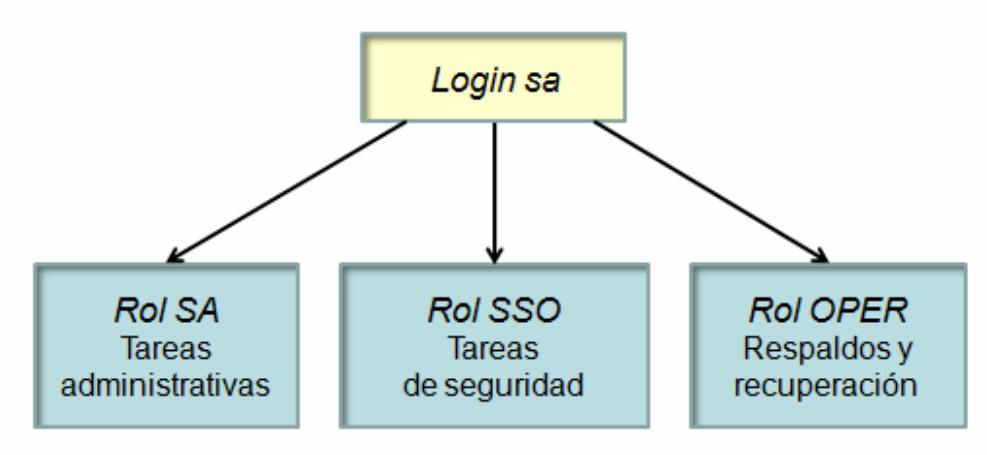


Fig. 3-4 Roles del login sa

3.5.2 Roles de sistema

Los roles de sistema asignados al sa pueden ser asignados a cualquier login, para mantener la integridad de la Base de Datos es de gran importancia asignar los roles adecuados a cada login dependiendo las actividades que le corresponda ejecutar. A continuación describo cada una de las actividades que se pueden realizar con los roles de sistema.

Las tareas que se pueden realizar con el rol Administrador de sistema **"sa_role"** son:

- Administrar el almacenamiento en disco
- Borrar y modificar logines
- Otorgar/revocar el sa_rol
- Crear Bases de Datos de usuario; otorgar propiedad sobre ellas
- Otorgar ciertos permisos a los usuarios del sistema
- Cambiar los parámetros de configuración para mejorar el rendimiento de ASE
- Respaldar las Bases de Datos
- Monitorear la recuperación de las Bases de Datos

El rol de Oficial de seguridad del sistema **"sso_rol",** tiene todos los privilegios para realizar las tareas relacionadas a la seguridad del sistema tales como:

- Crear logines y asignarles una contraseña inicial
- Cambiar la contraseña de cualquier cuenta
- Otorgar y revocar los roles de sistema sso_rol y oper_rol
- Crear, otorgar y revocar roles de usuario
- Configurar el periodo de expiración de las contraseñas

Los usuarios asignados al Rol de Operador **"oper_rol"** pueden respaldar y cargar Bases de Datos en el servidor ASE. El rol de operador permite a un usuario utilizar los comandos dump database, dump transaction, load database y load transaction para respaldar y recuperar todas las Bases de Datos en un servidor sin tener que ser el dueño de cada una de ellas. Estas operaciones pueden ser realizadas en solo una Base de Datos por el dueño de la Base de Datos o el administrador de sistema.

3.5.3 Dueños de objetos

Adicionalmente a los roles de sistema, existen dos tipos de dueños de objetos para restringir el acceso a la Base de Datos

El dueño de una Base de Datos **"dbo"(database owner)** es quien la crea, el dueño de una Base de Datos también puede ser un usuario al que se le transfirió la propiedad de la Base de Datos.
Un dbo puede ejecutar el procedimiento almacenado sp_adduser para permitir a cualquier usuario acceder a su Base de Datos. El dbo también puede utilizar el comando grant para dar permiso a los usuarios para crear objetos y ejecutar comandos dentro de su Base de Datos.
Los dueños de Base de Datos no reciben automáticamente privilegios sobre los objetos creados por otros usuarios dentro de su Base de Datos. Sin embargo utilizando una combinación de los comandos setuser y grant los dbo pueden adquirir privilegios de cualquier objeto en la Base de Datos.

Un dueño de objetos de Base de Datos **"dboo" (data base object owner)** es un usuario quien crea objetos dentro de una Base de Datos. Un objeto de Base de Datos puede ser una tabla, índice, vista, regla, procedimiento almacenado, defaults, llaves, etc. Antes de que un usuario pueda crear un objeto, el dbo debe otórgale los permisos para crear objetos. El dueño de los objetos crea un objeto utilizando la sentencia apropiada, y entonces otorga permisos sobre el mismo a otros usuarios.

El creador de los objetos es automáticamente asignado con todos los permisos en ese objeto.

3.5.4 Login

Para crea logines dentro de un servidor ASE, se utiliza el procedimiento almacenado sp_addlogin, el cual almacena información del login en la tabla syslogins de la Base de Datos master.

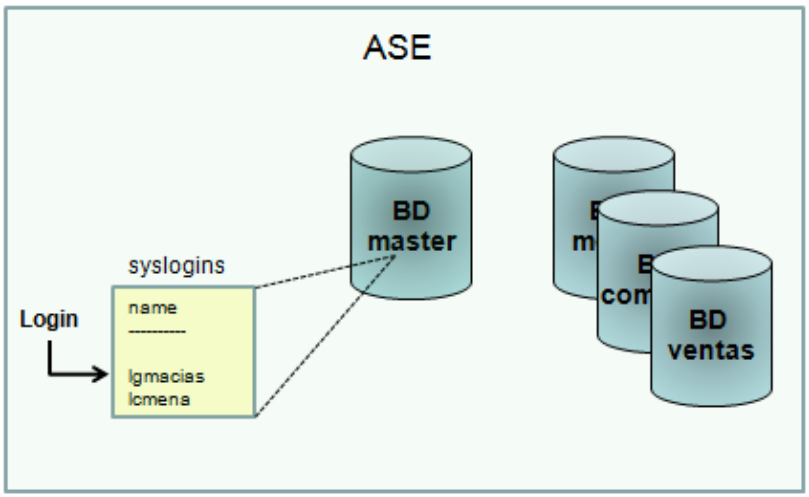


Fig. 3-5 Creación de login

Con el rol sso se pueden agregar logines usando **sp_addlogin**

Sintaxis	Ejemplo
sp_addlogin <nombre_login>, <contraseña>, <bd_predeterminada>, <leguaje_predeterminado>, <"nombre_completo">	sp_addlogin lgmacias, inicio10, ventas, spanish, "Gabriel Macias Duran"

Tabla 3-11 Sintaxis creación de login

Se debe especificar una Base de Datos predeterminada (si no se especifica, *master* es la predeterminada, lo que no es recomendable).

Modificación de Login

La información de un login puede ser modificada ya sea por el sa o por el mismo login. Se utiliza sp_modifylogin para modificar la siguiente información de un login:

- El nombre completo
- El lenguaje predeterminado
- La Base de Datos predeterminada

Sintaxis	Ejemplo
sp_modifylogin <nombre_login>, <*columna*>, <*valor*>	sp_modifylogin lgmacias, defdb, nomina sp_modifylogin lgmacias, deflanguage, english sp_modifylogin lgmacias, fullname, "Gabriel Macias"

Tabla 3-12 Sintaxis modificación de login

La contraseña de un login puede ser modificada ya sea por el sso o por el mismo login utilizando el procedimiento sp_password.

Sintaxis		Ejemplo
Ejecutado por el login	sp_password <*contraseña_actual*> , <*nueva_contraseña*>	sp_password inicio10, von377
Ejecutado por el SSO	sp_password <*contraseña_de_sso*>, <*nueva_contraseña*>,<*nombre_login*>	sp_password ssopass, inicio, lgmacias

Tabla 3-12 Sintaxis modificación de contraseña

Eliminación o bloqueo de Login

Si un login no debe seguir siendo utilizado, (ya sea por que el empleado ya no trabaja en la empresa o bien sus funciones cambiaron y no requieren de ese acceso) se puede elegir eliminarlo o bloquearlo. Bloquear un login es más seguro que eliminarlo, esto porque bloquear un login mantiene el suid para que no pueda ser reutilizado.
ASE puede reutilizar el suid de un login eliminado cuando crea un nuevo login. Esto puede comprometer el acceso a los objetos de Base de Datos que fueron autorizados al anterior login.

Un login no se puede eliminar cuando:

- El login está dentro de cualquier Base de Datos
- El login es el único sa o sso dentro del sistema

Con los roles sa o sso se puede utilizar sp_locklogin para desbloquear o bloquear logines.

Sintaxis		Ejemplo
Bloquear login	sp_locklogin <nombre_login>, lock	sp_locklogin lgmacias, lock
Desbloquear login	sp_locklogin <nombre_login>, unlock	sp_locklogin lgmacias, unlock

Tabla 3-13 Sintaxis bloqueo/desbloqueo de login

Con el rol sa se puede eliminar un login con el comando sp_droplogin

Sintaxis	Ejemplo
sp_droplogin *<nombre_login>*	sp_droplogin *lgmacias*

Tabla 3-14 Sintaxis eliminación de login

3.5.5 Usuarios de Bases de Datos

Para acceder a una Base de Datos, el usuario debe estar dado de alta en esa Base de Datos.
Los usuarios son almacenados en la tabla *sysusers* de cada una de las Base de Datos en donde son creados.

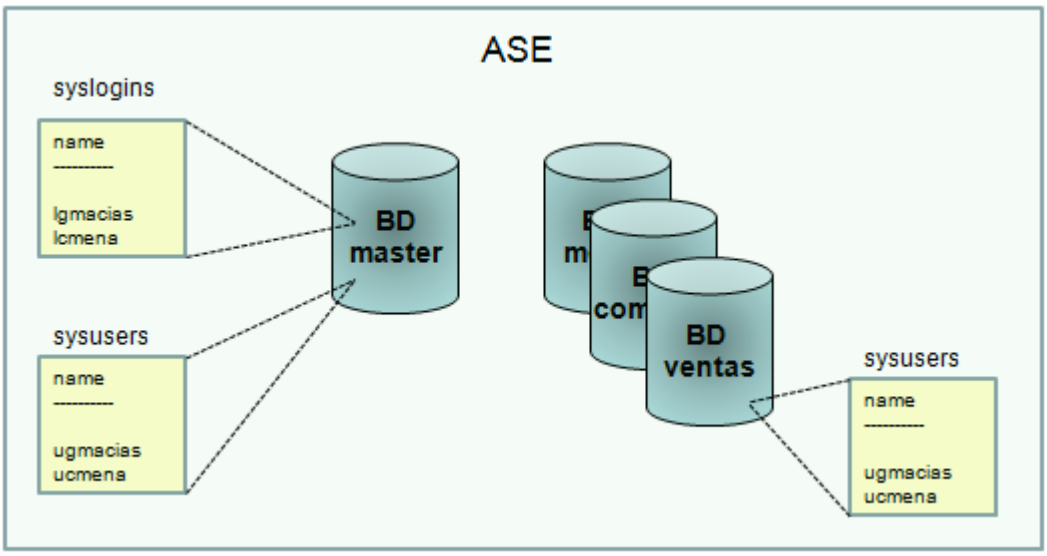


Fig. 3-6 Agregando usuarios a una Base de Datos

El sa o dbo utilizan sp_adduser para agregar usuarios a una Base de Datos. Con este comando también se puede asignar el usuario a un grupo.

Sintaxis	Ejemplo
sp_adduser *<nombre_login>*, *<nombre_usuario_bd>*, *<nombre_grupo>*	sp_adduser lgmacias, ugmacias, contabilidad

Tabla 3-15 Sintaxis agregando usuarios a bd

Eliminación de usuarios de las Bases de Datos

El sa o dbo pueden utilizar sp_dropuser para eliminar el acceso a un usuario a la Base de Datos en donde se está ejecutando el procedimiento sp_dropuser.

Sintaxis	Ejemplo
sp_dropuser *<nombre_usuario_bd>*	sp_dropuser ugmacias

Tabla 3-16 Sintaxis eliminación de usuarios

3.5.6 Permisos en los objetos de Bases de Datos

Existen dos tipos de permisos para objetos, los permisos de acceso a los objetos y los permisos de creación de objetos, dentro de estos últimos encontramos el permiso para crear Bases de Datos “create database” este permiso solo puede ser asignado o revocado por el sa.

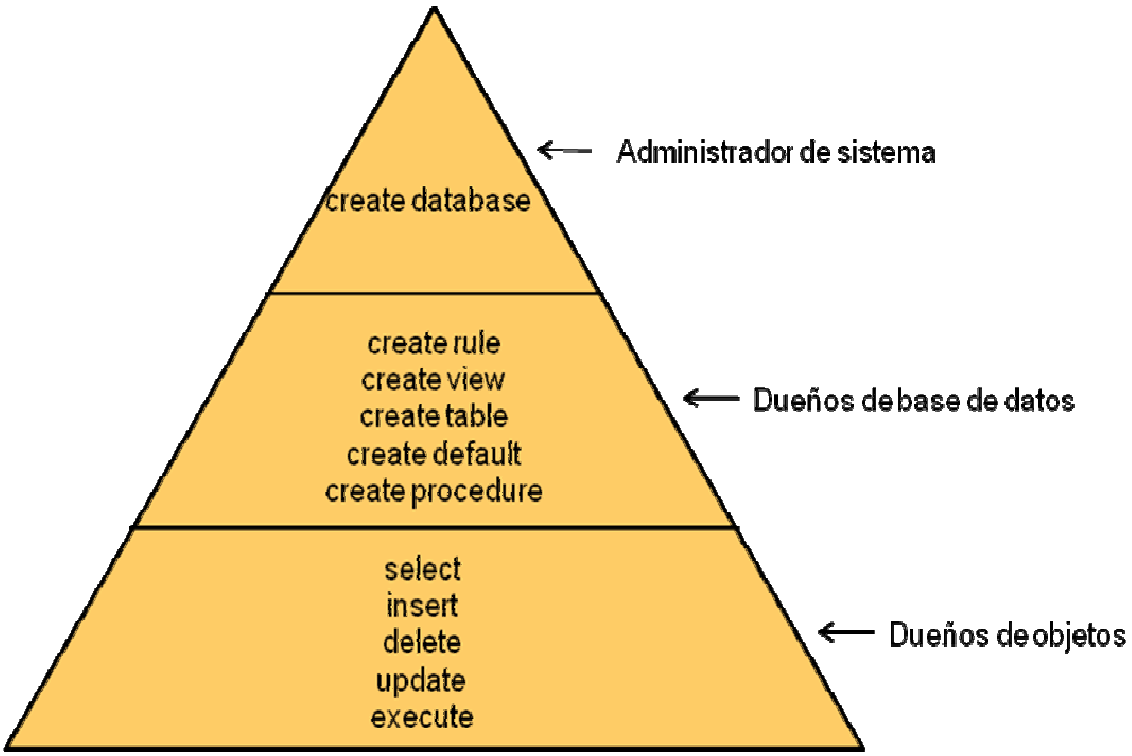


Fig. 3-7 Permisos en los objetos de Base de Datos

Permisos de acceso a los objetos

Los permisos de acceso a los objetos regulan el uso de ciertos comandos que acceden a ciertos objetos de Base de Datos. Por ejemplo, para explícitamente otorgar permiso para utilizar el comando select en la tabla autores. Los permisos de acceso a los objetos son otorgados y revocados por el dboo y el sa.
La siguiente tabla muestra la relación de los permisos de acceso y los objetos para los que aplican.

Permiso	Objeto
select	Tabla, vista, columna
update	Tabla, vista, columna
insert	Tabla, vista
delete	Tabla, vista
references	Tabla, columna
execute	Procedimiento almacenado

Tabla 3-17 Permisos de acceso a los objetos

Para otorgar permisos en objetos específicos se utiliza el comando grant.

Sintaxis		Ejemplo
Permisos en tablas para login(es)	grant <permiso(s)> on <nombre_tabla(s)> to <nombre_login(es)>	grant insert on templeados to lgmacias
Permisos en tablas para rol(es)	grant <permiso(s)> on <nombre_tabla(s)> to <nombre_rol(es)>	grant delete on templeados to gerente_ventas
Permisos en tablas para grupo(s)	grant <permiso(s)> on <nombre_tabla(s)> to <nombre_grupo(s)>	grant update on templeados to vendedores
Permisos en vistas para login(es)	grant <permiso(s)> on <nombre_vista(s)> to <nombre_login(es)>	grant select on vlibros to lgmacias
Permisos en vistas para rol(es)	grant <permiso(s)> on <nombre_vista(s)> to <nombre_rol(es)>	grant insert on vlibros to gerente_ventas
Permisos en vistas para grupo(s)	grant <permiso(s)> on <nombre_vista(s)> to <nombre_grupo(s)>	grant insert on vlibros to vendedores
Permisos en procedimientos almacenados para login(es)	grant <permiso(s)> on <nombre_sp(s)> to <nombre_login(es)>	grant execute on sp_helpdb to lgmacias
Permisos en procedimientos almacenados para rol(es)	grant <permiso(s)> on <nombre_sp(s)> to <nombre_login(es)>	grant execute on sp_helpdb to gerente_ventas
Permisos en procedimientos almacenados para grupo(s)	grant <permiso(s)> on <nombre_sp(s)> to <nombre_grupo(s)>	grant execute on sp_helpdb to vendedores

Tabla 3-18 Sintaxis otorgando permisos de acceso

Para revocar permisos sobre objetos específicos se utiliza el comando revoke

Sintaxis		Ejemplo
Permisos en tablas para login(es)	revoke <permiso(s)> on <nombre_tabla(s)> from <nombre_login(es)>	revoke insert on templeados from lgmacias
Permisos en tablas para rol(es)	revoke <permiso(s)> on <nombre_tabla(s)> from <nombre_rol(es)>	revoket delete on templeados from gerente_ventas
Permisos en tablas para grupo(s)	revoke <permiso(s)> on <nombre_tabla(s)> from <nombre_grupo(s)>	revoke update on templeados from vendedores
Permisos en vistas para login(es)	revoke <permiso(s)> on <nombre_vista(s)> from <nombre_login(es)>	revoke select on vlibros from lgmacias
Permisos en vistas para rol(es)	revoke <permiso(s)> on <nombre_vista(s)> from <nombre_rol(es)>	revoke insert on vlibros from gerente_ventas
Permisos en vistas para grupo(s)	revoke <permiso(s)> on <nombre_vista(s)> from <nombre_grupo(s)>	revoke insert on vlibros from vendedores
Permisos en procedimientos almacenados para login(es)	revoke <permiso(s)> on <nombre_sp(s)> from <nombre_login(es)>	revoke execute on sp_helpdb from lgmacias
Permisos en procedimientos almacenados para rol(es)	revoke <permiso(s)> on <nombre_sp(s)> from <nombre_login(es)>	revoke execute on sp_helpdb from gerente_ventas
Permisos en procedimientos almacenados para grupo(s)	revoke <permiso(s)> on <nombre_sp(s)> from <nombre_grupo(s)>	revoke execute on sp_helpdb from vendedores

Tabla 3-19 Sintaxis revocando permisos de acceso

Permisos de creación de objetos

Los permisos de creación de objetos regulan el uso de comandos que crean objetos. Estos permisos pueden ser otorgados solo por el sa o el dbo. Los comandos de creación de objetos son:

- create database
- create default
- create procedure
- create rule
- create table
- create view

Para otorgar privilegios en la creación de objetos se utiliza el comando grant.

Sintaxis		Ejemplo
Privilegios para login(es)	grant <permiso(s)> to <nombre_login(es)>	grant create rule to lgmacias
Privilegios para rol(es)	grant <permiso(s)> to <nombre_rol(es)>	grant create table to gerente_ventas
Privilegios para grupo(s)	grant <permiso(s)> to <nombre_grupo(s)>	grant create view to vendedores

Tabla 3-20 Sintaxis otorgando permisos de creación de objetos

Para revocar permisos sobre la creación de objetos se utiliza el comando revoke

Sintaxis		Ejemplo
Privilegios desde login(es)	revoke <permiso(s)> from <nombre_login(es)>	revoke create rule from lgmacias
Privilegios desde rol(es)	revoke <permiso(s)> from <nombre_rol(es)>	revoke create table from gerente_ventas
Privilegios desde grupo(s)	revoke <permiso(s)> from <nombre_grupo(s)>	revoke create view from vendedores

Tabla 3-21 Sintaxis revocando permisos de creación de objetos

3.5.7 Grupos

Lo grupos proporcionan un camino conveniente para otorgar y revocar permisos a más de un usuario en una simple sentencia. Los grupos permiten asignar un nombre colectivo a un grupo de usuarios. Son especialmente útiles si sé administra un gran número de usuarios dentro de una Base de Datos especifica.
Todos los usuarios son miembros del grupo “public” y pueden también ser miembros de otro grupo. Los usuarios permanecen en el grupo “public”, aun cuando están dentro de otro grupo.

Un sa o dbo pueden crear un grupo en cualquier momento utilizando sp_addgroup

Sintaxis	Ejemplo
sp_addgroup <nombre_grupo>	sp_addgroup gerentes

Tabla 3-22 Sintaxis creación de grupo

El sa puede asignar logines a los grupos con sp_changegroup

Sintaxis	Ejemplo
sp_changegroup <nombre_grupo>, <nombre_login(es)>	sp_changegroup gerentes, lcmena

Tabla 3-23 Sintaxis modificación de grupo

Para eliminar un grupo se utiliza sp_dropgroup. No se puede eliminar un grupo que tiene miembros.

Sintaxis	Ejemplo
sp_dropgroup *<nombre_grupo>*	sp_dropgroup gerentes

Tabla 3-24 Sintaxis eliminación de grupo

3.5.8 Roles

Si los roles predeterminados de ASE no cumplen con los requerimientos de control de acceso para los diferentes puesto dentro de una organización, es posible crear roles de usuario con los permisos necesarios para realizar las tareas asociadas a cada puesto.
Los roles de usuarios se almacenan en la Base de Datos *master*, y están asignados a los logines.

Antes de implementar roles de usuario se debe definir:

- Los roles que se requieren crear
- Las responsabilidades de cada rol (permisos)
- La posición jerárquica de cada rol
- Cuales roles serán mutuamente exclusivos
- Definir nomenclatura

Una vez definidos los puntos anteriores el sso puede iniciar con la creación de roles utilizando el comando “create role”.

Sintaxis	Ejemplo
create role *<nombre_rol>*	create role gerente_ventas

Tabla 3-25 Sintaxis creación de rol

Después de que un rol ha sido creado, se deben otorgar los permisos de acceso definidos para este rol.

Sintaxis	Ejemplo
grant <permiso(s) on <nombre_tabla> to <nombre_rol>	grant delete on templeados to gerente_ventas
grant <permiso(s) on <nombre_vista> to <nombre_rol>	grant insert on vlibros to gerente_ventas
grant <permiso(s) on <nombre_sp> to <nombre_rol>	grant execute on sp_helpdb to gerente_ventas

Tabla 3-26 Sintaxis otorgando permisos al rol

Para eliminar un rol se utiliza el siguiente comando:

Sintaxis	Ejemplo
drop role <nombre_rol> with override	drop role gerente_ventas with override drop role gerente_ventas

Tabla 3-27 Sintaxis eliminación de rol

La opción “with override” revoca todos los permisos de acceso otorgados al rol en cada una de las Bases de Datos. Si el rol tiene privilegios de acceso asignados, primero se deben revocar todos los privilegios antes de que pueda ser eliminado el rol. La opción “with override” asegura que ASE automáticamente elimina toda la información de privilegios. Si se utiliza la opción “with override” no es necesario eliminar los usuarios asignados antes de borrar un role.

Para revocar permisos de los roles:

Sintaxis	Ejemplo
revoke <permiso(s)> from <nombre_rol>	revoket delete on templeados from gerente_ventas

Tabla 3-28 Sintaxis revocando permisos al rol

Los roles se pueden asignar a los logines mediante el siguiente comando:

Sintaxis	Ejemplo
grant role <nombre_rol(es)> to <nombre_login(es)>	grant role gerente_ventas to lcmena

Tabla 3-27 Sintaxis asignando roles a logines

Para revocar roles a los logines se utiliza:

Sintaxis	Ejemplo
revoke role <nombre_rol(es)> from <nombre_login(es)>	revoke role gerente_ventas from lcmena

Tabla 3-28 Sintaxis revocando roles a logines

3.6 Respaldos y Recuperación

Como lo hemos visto anteriormente la información tiene un gran valor para las organizaciones, por lo cual es indispensable asegurarla ante todo tipo de amenaza que ponga en riesgo las características que le dan valor. Para lograr esto el administrador de Base de Datos debe respaldar las Bases de Datos regularmente.

ASE integra un servidor de respaldos para poder realizar las actividades de respaldo y recuperación de las Bases de Datos, se recomienda utilizar este servidor y sus herramientas y no las herramientas proporcionadas por el sistema operativo ya que el uso de las herramientas propias de ASE garantiza una total recuperación de los datos en caso de daño o perdida de información a diferencia de las herramientas proporcionadas por el sistema operativo que corrompen los datos.
Después de una falla o perdida de información de una Base de Datos, esta puede ser recuperada, si y solo si, se tienen respaldos regulares de la Base de Datos y su log de transacciones.

Una recuperación completa depende de el uso regular de los comandos **dump database** y **dump transaction** para respaldar la Base de Datos y los comandos **load database** y **load transaction** para recuperarla.

3.6.1 ¿Qué es un servidor de Respaldos ASE?

Es el encargado de realizar los respaldos y recuperación de las Bases de Datos y su log.
El servidor de respaldos puede respaldar la información en dispositivos, los cuales son llamados dispositivos de respaldo "**dump device"** y pueden ser al igual que un dispositivo de Base de Datos un espacio en disco duro o bien citas magnéticas.

Algunas de las características que del servidor de respaldos ASE son:

- Dinámico: Se pueden realizar mientras los usuarios están utilizando la Base de Datos.
- Remoto: Puede respaldar Bases de Datos locales así como Bases de Datos de otros servidores remotos ASE.
- Automático: Se pueden programar respaldos automáticos.
- Multi-Base de Datos: Se puede realizar el respaldo de varias Bases de Datos en mismo dispositivo de respaldos.
- Multi-dispositivo: Se puede realizar el respaldo de una Base de Datos en varios dispositivos de respaldo.

El servidor de respaldos ASE respeta la arquitectura cliente-servidor, en donde el servidor de Base de Datos ASE es cliente del servidor de respaldos ASE. La siguiente figura ilustra mejor esta relación:

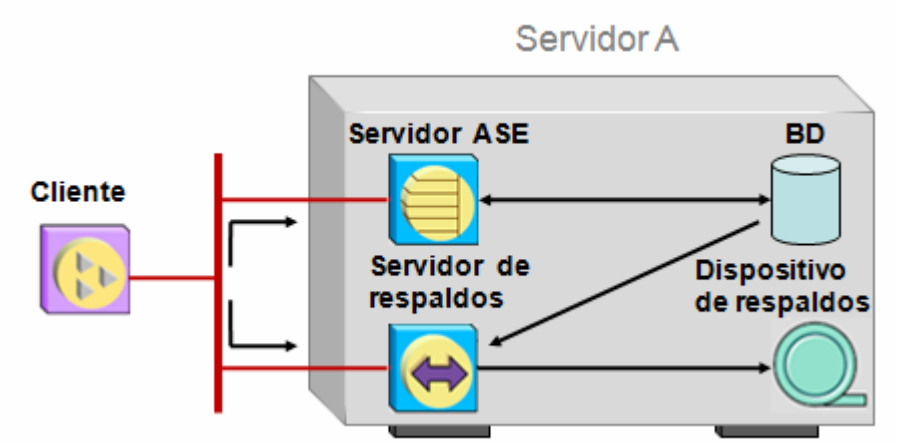


Fig. 3-8 Servidor de respaldos ASE local

Como anteriormente mencione una de las características del servidor de respaldos ASE, es que puede realizar respaldos de forma remota, para que esto sea posible debe existir comunicación entre el servidor de respaldos local (en donde se encuentra la Base de Datos a respaldar) y el servidor de respaldos remoto.

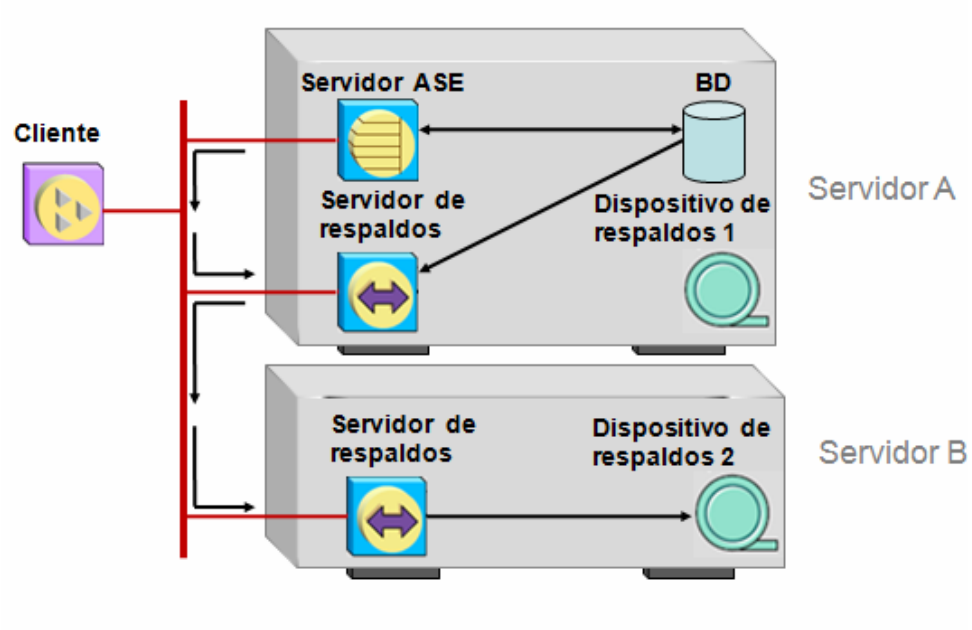


Fig. 3-9 Servidor de respaldos ASE remoto

Para poder utilizar un servidor de respaldos ASE, ya sea local o remoto es necesario realizar una configuración previa. El procedimiento de configuración se encuentra en las guías de instalación.

¿Pero qué se debe respaldar?
Existen dos tipos de respaldos que deben ejecutarse regularmente, el respaldo de Base de Datos y el respaldo del log de transacciones..

3.6.2 Respaldo de Base de Datos

Con el este tipo de respaldos se realiza un respaldo de toda la información contenida en la Base de Datos y su log de transacciones. Se utiliza el comando **dump database** para realizar este tipo de respaldos. El comando **dump database** permite realizar respaldos dinámicos. Los usuarios pueden continuar haciendo modificaciones a las Base de Datos mientras se está ejecutando el respaldo. Esto lo hace conveniente para respaldar las Bases de Datos regularmente ya que no detiene los procesos de operación diaria de una organización.
El comando **dump database** es ejecutado en tres fases. Un mensaje de progreso informa cuando cada fase ha sido concluida. Cuando un respaldo finaliza, este refleja todos los cambios hechos durante su ejecución, excepto aquellos iniciados durante la tercera fase.

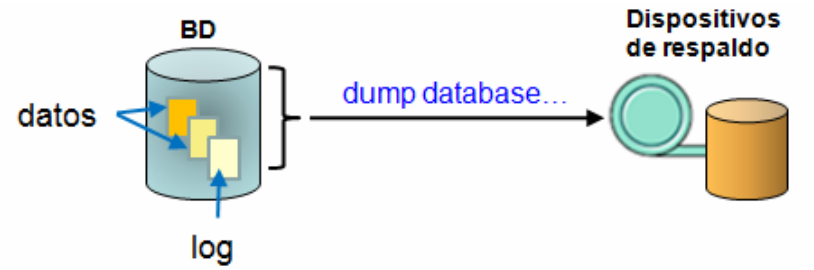


Fig. 3-10 Respaldo de Bases de Datos

Se puede ejecutar local o remotamente la sintaxis es la siguiente:

Sintaxis		Ejemplo
Local	dump database <*nombre_db*> *to* <nombre_dispositivo_respaldo>	dump database ventas to disp_resp1
Remoto	dump <*nombre_db*> *to* <nombre_dispositivo_respaldo> at <*nombre_servidor_remoto*>	dump database ventas to disp_resp1 at SRESPA

Tabla 3-29 Sintaxis Respaldos

Respaldo Multi-Base de Datos

ASE permite respaldar varias Bases de Datos en un mismo respaldo y cinta. Para poder realizar este tipo de respaldo se deben utilizar como dispositivos únicamente cintas magnéticas.

Sintaxis	Ejemplo
dump database <nombre_bd-1> to <nombre_dispositivo_respaldo> with init dump database <nombre_bd-2> to <nombre_dispositivo_respaldo> dump database <nombre_bd-3> to <nombre_dispositivo_respaldo> with unload	dump database ventas to disp_resp1 with init dump database nomina to disp_resp1 dump database compras to disp_resp1 with unload

Tabla 3-30 Sintaxis Respaldo multi-Base de Datos

Para respaldar multiples Bases de Datos en una misma cinta se debe considerar:

- Utilizar la opción init para la primera Base de Datos. Esta sobrescribe cualquier respaldo existente y almacena el primer respaldo en el inicio de la cinta.
- Utilizar la opción default noinit y nounload) para las Bases de Datos subsecuentes: Esta opción almacena las Bases de Datos una después de la otra en la cinta.
- Utilizar la opción unload para la última Base de Datos en la cinta: Esta opción rebobina y desmonta la cinta después de reslpaldar la última Base de Datos.

La figura 3-11 ilustra como respaldar tres Bases de Datos en solo una cinta.

Fig. 3-11 Respaldo multi-Base de Datos

Respaldo multi-dispositivo

Se puede respaldar una Base de Datos en diferentes dispositivos utilizando la opción *stripe on*

Sintaxis	Ejemplo
dump database <nombre_bd> to “<nombre_dispositivo_respaldo1>” stripe on “<nombre_dispositivo_respaldo2>” stripe on “<nombre_duspositivo_respaldo3>”	dump database ventas to disp_resp1 stripe on disp_resp2 stripe on disp_resp3"

Tabla 3-31 Sintaxis Respaldo multi-dispositivo

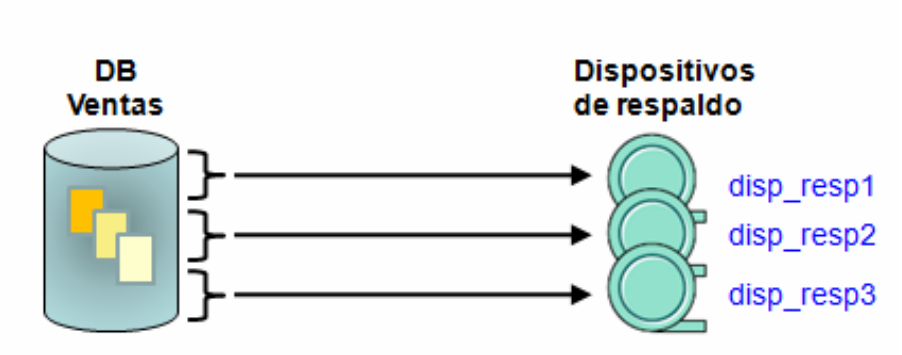


Fig. 3-12 Respaldo multi-dispositivo

3.6.3 Respaldo del log de transacciones

Para realizar respaldos del log de transacciones se utiliza el comando **dump transaction** (o su abreviación dump tran).
El comando **dump transaction** respalda el log de transacciones, respaldando todas las modificaciones hechas a la Base de Datos a partir del último respaldo del log de transacciones.

Después de que el comando dump transaction ha respaldado el log, este puede truncar (eliminar) las porciones inactivas del log (transacciones almacenadas en disco), dependiendo las opciones utilizadas.

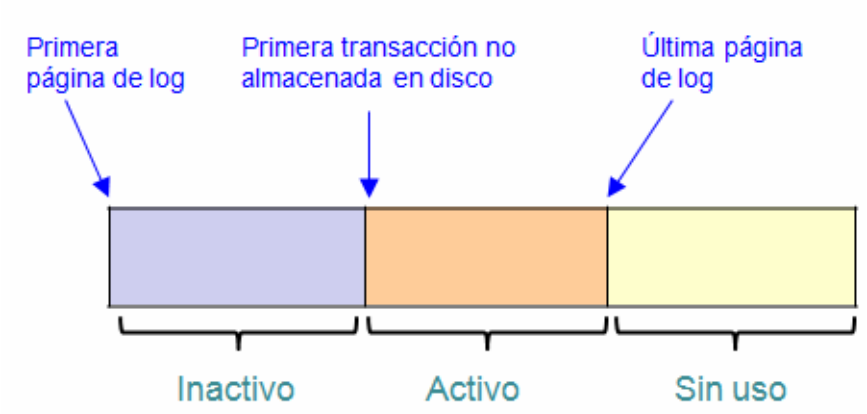


Fig. 3-13 Clasificación del Log de transacciones

El comando dump transaction toma menor tiempo y espacio que un respaldo completo de la Base de Datos hecho con dump database, y este es usualmente ejecutado más veces. Los usuarios pueden continuar haciendo cambios a las Base de Datos mientras se ejecuta este tipo de respaldo.

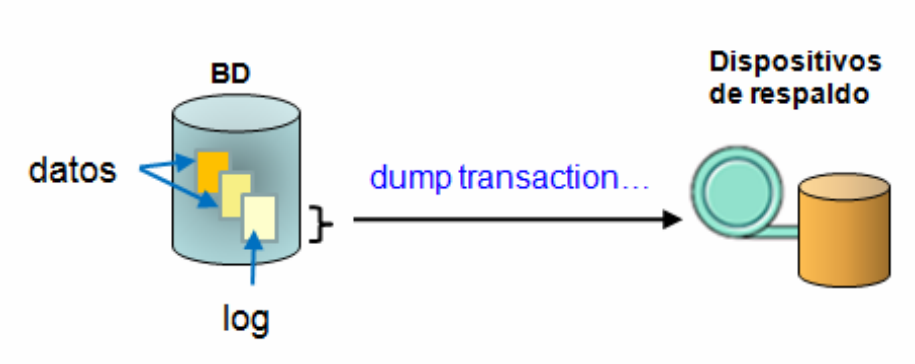


Fig. 3-14 Respaldo del log

Sintaxis	Ejemplo
dump transaction <*nombre_db*> to <nombre_dispositivo_respaldo>	dump transaction ventas to disp_logresp1

Tabla 3-32 Sintaxis respaldo del log

El log de transacciones es añadido cada vez que se modifica la Base de Datos; y puede llegar a llenarse rápidamente. Si el log se llena, y no es truncado, todos los trabajos de modificación se detienen (los usuarios no pueden realizar ninguna consulta ni modificación a la Base de Datos). Por lo tanto el log debe ser truncado para que esto no suceda.

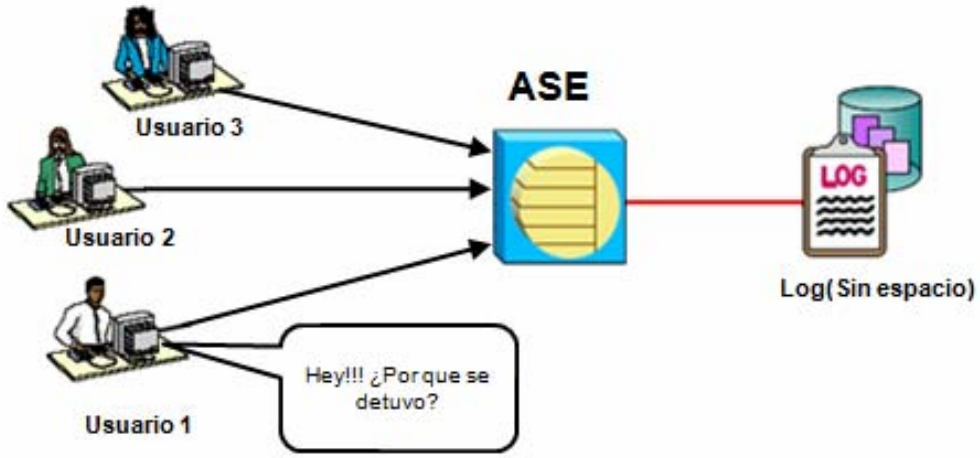


Fig. 3-15 Log sin espacio

Dependiendo el tipo de respaldo de log que se requiera, es posible utilizar las opciones truncate_only, no_log y no_truncate en el comando dump transaction.

Sintaxis		Ejemplo
Trunca el log pero no lo respalda	dump transaction <nombre_bd> to <nombre_dispositivo_respaldo> with truncate_only	dump transaction ventas to disp_logresp1 with truncate_only
Trunca el log pero no lo respalda	dump transaction <nombre_bd> with no_log	dump transaction ventas with no_log
Respalda el log pero no lo trunca	dump transaction <nombre_bd> to <nombre_dispositivo_respaldo> with no_truncate	dump transaction ventas to disp_logresp1 with no_truncate

Tabla 3-33 Sintaxis opciones de Respaldo de log

3.6.4 Recuperación

Para recuperar una Base de Datos que se encuentra dañada se utiliza el comando **load database.** Con este comando se carga el respaldo creado con **dump database**.
Pueden existir varios escenarios al momento de recuperar una Base de Datos, entre los más comunes encontramos:

Cuando se daño completamente el servidor de Base de Datos y es necesario recuperar la Base de Datos en un nuevo servidor ASE. Para este escenario se debe crear una nueva Base de Datos con las mismas características (nombre, tamaño, dueño, etc.) de la Base de Datos original (respaldada). Para realizarlo es necesario utilizar la opción *for load* en el comando **create database.** No se puede cargar un respaldo que fue hecho en una plataforma diferente a la del nuevo servidor.

Cuando se daño únicamente el contenido de la Base de Datos, se puede cargar el respaldo en ese mismo servidor y Base de Datos. Al cargar un respaldo en una Base de Datos existente se sobrescriben todos los datos contenidos en ella.

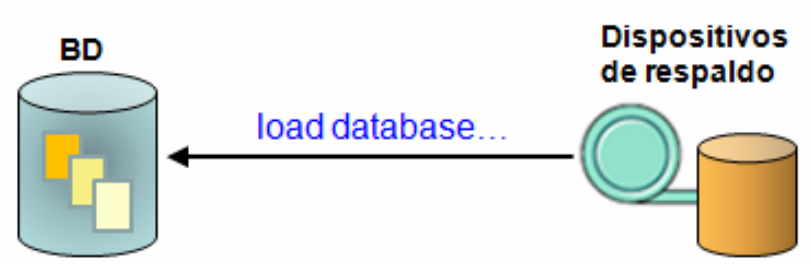


Fig. 3-16 Recuperación de Base de Datos

Sintaxis	Ejemplo
load database <*nombre_bd*> from <nombre_dispositivo_respaldo>	load database ventas from disp_resp1

Tabla 3-33 Sintaxis recuperación de la Base de Datos

Recuperando los cambios después dump database

Después de que la Base de Datos ha sido recuperada con el comando **load database**, se puede utilizar el comando **load transaction** (o su abreviación load tran) para cargar cada respaldo del log de transacciones en el orden en que fueron hechos. Este proceso reconstruye la Base de Datos re-ejecutando los cambios almacenados en el log de transacciones.

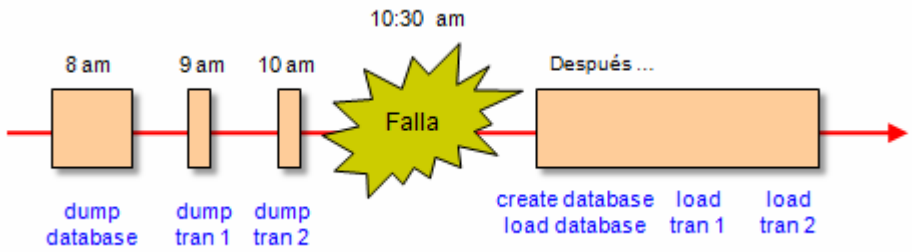


Fig. 3-17 Escenario de Recuperación

Sintaxis	Ejemplo
load transaction <nombre_bd> from <nombre_dispositivo_respaldo>	load transaction ventas from disp_logresp1

Tabla 3-34 Sintaxis Recuperación del log

Tópicos Avanzados de Bases de Datos

El uso de las Bases de Datos se ha popularizado, siendo estas utilizadas no solo para administrar las Bases de Datos Transaccionales, sino que también para soportar la implementación de herramientas especializadas que son utilizadas por la alta dirección y que sirven como base para la toma de decisiones dentro de una organización.
La Minería de Datos y el Data Warehouse, son ejemplo de este tipo de herramientas, siendo este ultima de interés para el desarrollo de este capítulo.

4.1 Data Warehouse

Un Data Warehouse es un almacén de datos que puede verse como una colección de datos integrados, variables en el tiempo, no volátiles y orientados a un tema, y que son utilizados principalmente en las organizaciones para la toma de decisiones.

4.1.1 Data Warehouse y Bases de Datos Transaccionales

Típicamente la administración de un Data Warehouse está separada de la Base de Datos Transaccional de una organización, existen varias razones para hacer esto:

Un Data Warehouse soporta Procesamiento Analítico en Línea u OnLine Analytical Processing (OLAP), los requerimientos de funcionalidad y rendimiento son un poco diferentes a las aplicaciones que utilizan el Procesamiento Transaccional en Línea u OnLine Transaction Processing (OLTP) y que tradicionalmente son soportadas por las Bases de Datos Transaccionales.

El tamaño de las Bases de Datos Transaccionales tiende a ser cientos de megabytes o gigabytes. La consistencia y la facilidad de recuperación de una Base de Datos Transaccional son críticas, y el factor clave de rendimiento es minimizar el tiempo de ejecución de las transacciones. Consecuentemente las Bases de Datos Transaccionales están diseñadas para reflejar los requerimientos funcionales de las aplicaciones.

Los Data Warehouses, en contraste, son orientados para soportar la toma de decisiones. Los datos históricos, resumidos y consolidados son más importantes que los registros individuales detallados. Desde que los Data Warehouses contienen datos consolidados, quizás de varias Bases de datos transaccionales, bajo periodos de tiempo potencialmente largos, estos tienden a ser de magnitud más grande que las Bases de Datos transaccionales; los Data Warehouses son proyectados a tener cientos de gigabytes a terabytes de tamaño.

La carga de trabajo son consultas intensivas mayormente adecuadas a las necesidades de la empresa, consultas complejas que pueden acceder a millones de registros y realizar gran cantidad de lecturas, joins y funciones de agregado. La tasa de transferencia de las consultas y los tiempos de respuesta son más importantes que el rendimiento de transacciones.

Para facilitar el análisis complejo y visualización, los datos en un Data Warehouse son típicamente modelados multidimensionalmente. Por ejemplo, en un Data Warehouse de ventas, la fecha de venta, la región de venta, el vendedor, y el producto pueden ser algunas de las dimensiones de interés. A veces estas dimensiones son jerárquicas; la fecha de venta puede ser organizada como una jerarquía de día-mes-cuatrimestre-año, los productos organizados como una jerarquía pueden ser producto-categoría-industria.

Dado que las Bases de Datos transaccionales son finamente ajustadas para soportar las cargas de trabajo de OLTP, intentando ejecutar consultas complejas del tipo OLAP contra las Bases de Datos transaccionales podría resultar en un rendimiento inaceptable. Además, para soportar las decisiones requiere datos que pueden estar perdidos en la Base de Datos operacional; por ejemplo, entender tendencias o hacer predicciones requiere datos históricos, mientras que las Bases de Datos transaccionales almacenan solo datos actuales.

Para soportar decisiones usualmente requiere consolidar datos desde varias fuentes de información: estas pueden incluir fuentes externas tales como el mercado de valores, o varias Bases de Datos transaccionales adicionales.

Finalmente, soportar los modelos de datos multidimensionales y operaciones típicas de OLAP requiere una organización de datos especial, métodos de acceso, y métodos de implementación, que no son proporcionados generalmente por los RDBMS comerciales orientados para OLTP. Estas son todas las razones por las que los Data Warehouse son implementados de forma separada de las Bases de Datos transaccionales

4.1.2 ROLAP y MOLAP

Los Data Warehouses pueden ser implementados en RDBMS estándar o en RDBMS extendidos llamados servidores relacionales OLAP (ROLAP). Estos servidores asumen que los datos son almacenados en Bases de Datos relacionales, ellos soportan extensiones del lenguaje SQL para un acceso especial y métodos para implementar eficientemente el modelo de datos multidimencional y sus operaciones.

En contraste, los servidores multidimensionales OLAP (MOLAP) son servidores que directamente almacenan datos multidimencionales en estructuras de datos especiales e implementan las operaciones OLAP bajo estas estructuras de datos especiales.

4.1.3 Data Mart

Construir un Data Warehouse empresarial es un proceso largo y complejo, requiere un extenso modelado de negocio, y puede tomar muchos años para lograrlo. En su lugar algunas organizaciones utilizan los Data Marts, los cuales son subgrupos departamentales enfocados a temas seleccionados (por ejemplo un Data Mart de mercadotecnia puede incluir información ventas, clientes y productos).

Estos Data Marts habilitan una rápida implementación, gracias a que no requieren un consenso de la organización completa.

4.1.4 Arquitectura del Data Warehouse

La siguiente figura muestra la arquitectura típica de un Data Warehouse.

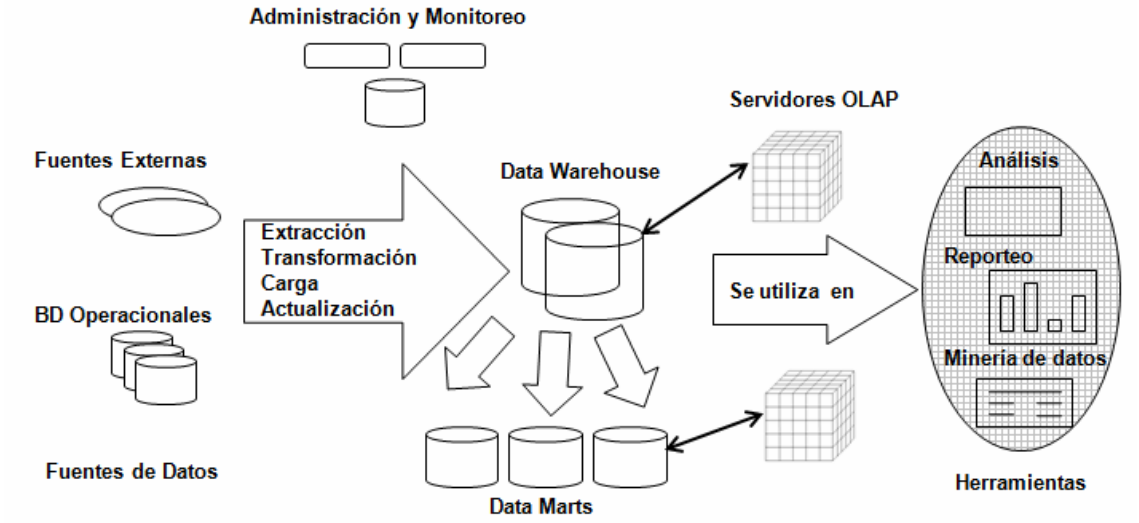


Figura 4-1 Arquitectura del Data Warehouse

Esta arquitectura incluye las fuentes de datos ya sean Bases de datos transaccionales o fuentes externas además de las herramientas para:

- Extraer datos desde varias Bases de Datos transaccionales y fuentes externas
- Limpiar, transformar e integrar estos datos
- Cargar datos dentro del Data Warehouse
- Periódicamente actualizar el Data Warehouse para reflejar las actualizaciones en las fuentes y para depurar datos del Data Warehouse

Adicionalmente al Data Warehouse principal, pueden existir varios Data Marts departamentales.

Los datos en el Data Warehouse y Data Marts son almacenados y administrados por uno o más servidores Data Warehouse, los cuales presentan vistas multidimensionales de los datos para una variedad de herramientas front end (interface de usuario final): herramientas de consulta, reporteo, análisis, y herramientas para la minería de datos.

Finalmente, existe un repositorio para almacenar y administrar los metadatos, y las herramientas para monitorear y administrar los sistemas de Data Warehouse.

4.1.5 Utilidades y herramientas Back End

Los sistemas Data Warehouse utilizan una variedad de herramientas para la extracción y limpieza de datos, y utilidades para cargar y actualizar los Data Warehouses.

Limpieza de datos

Como anteriormente describí un Data Warehouse es utilizado para la toma de decisiones, por lo cual es de vital importancia que los datos contenidos en el sean correctos. Sin embargo, debido a que grandes volúmenes de datos son involucrados desde diferentes fuentes, existe una alta probabilidad de errores y anomalías en los mismos.

Cuando la limpieza de datos llega a ser necesaria es porque la longitud de los campos es inconsistente, existen descripciones inconsistentes, asignación de valores inconsistentes y registros perdidos.

Existen tres clases de herramientas relacionadas a la limpieza de datos:

- Las herramientas de migración de datos permiten una simple transformación de reglas (por ejemplo, remplazar la cadena “genero” por la cadena “sexo”). Warehouse Manager de Prism es un ejemplo de este tipo de herramienta

- Las herramientas de depuración de datos. Explotan técnicas de análisis sintáctico y correspondencia confusa para lograr limpiar desde varias fuentes. herramientas tales como Integrity and Trillum fall están incluidas en esta categoría

- Las herramientas de auditoría hacen posible descubrir reglas y relaciones (señales de violación de reglas) por una inspección de datos. Por ejemplo, tales herramientas pueden descubrir un patrón sospechoso basado en análisis estadístico (por ejemplo, que un cierto vendedor de autos nunca a recibido una queja)

Carga

Después de la extracción, limpieza y transformación, los datos deben ser cargados en el Data Warehouse. Típicamente, las utilidades para carga en fondo son utilizadas para este propósito. Adicionalmente para poblar el Data Warehouse, una utilidad de carga debe permitir al administrador de sistema: monitorear el estado, cancelar, suspender y reanudar la carga.

Las utilidades de carga para un Data Warehouse tienen que ocuparse de volúmenes de datos mucho más grandes que las Bases de Datos transaccionales. Existe solo una pequeña ventana de tiempo (usualmente en la noche) cuando el Data Warehouse puede ser puesto fuera de línea para actualizarlo. Cargas secuenciales pueden tomar un largo tiempo, por ejemplo cargar un terabyte de datos puede tardar semanas hasta meses.

Por lo cual muchas utilidades comerciales (como RedBrick Table Management Utility) utilizan cargas incrementales durante la actualización para reducir el volumen de datos que tienen que ser incorporados dentro de los Data Warehouse. Solo los registros actualizados son insertados. Sin embargo, el proceso de carga es más pesado de administrar. Las cargas incrementales pueden tener conflicto con las consultas en curso, ya que estas son tratadas como una secuencia de transacciones más pequeñas (las cuales son grabadas periódicamente, por ejemplo, después de cada 1000 registros o algunos segundos), pero esta secuencia de transacciones tiene que ser coordinada para asegurar la consistencia de los datos.

Actualización

Actualizar un Data Warehouse consiste en propagar las actualizaciones de las fuentes de datos a las actualizaciones correspondientes en la Base de Datos y los datos derivados almacenados en un Data Warehouse. Existen dos cuestiones para considerar: ¿cuándo actualizar? y ¿cómo actualizar?

Usualmente, el Data Warehouse es actualizado periódicamente (por ejemplo, diario o semanalmente). Las políticas de actualización son establecidas por el administrador del Data Warehouse, dependiendo de las necesidades de los usuarios y el uso del Data Warehouse, y puede diferir dependiendo las fuentes.

Las técnicas de actualización también pueden depender de las características de la fuente y capacidades del servidor de Bases de Datos.

Muchos sistemas de Base de Datos proporcionan replicación de servidores que soportar técnicas incrementales para propagar las actualizaciones desde las Bases de Datos primarias a una o más replicas. Tales servidores de replicación pueden ser utilizados para actualizar incrementalmente un Data Warehouse cuando la fuente está cambiando.

4.1.6 Metodología de implementación de un Data Warehouse

Diseñar e implementar un Data Warehouse es un proceso complejo, consiste de las siguientes actividades:

- Definir la arquitectura, hacer la planeación de capacidad, y seleccionar los servidores de almacenamiento, servidores OLAP y de Base de Datos, y herramientas
- Integrar los servidores, almacenamiento y herramientas cliente
- Diseñar el esquema y vistas del Data Warehouse
- Definir la organización física del Data Warehouse, ubicación de los datos, particiones, y métodos de acceso
- Conectar las fuentes (Bases de Datos transaccionales y externas)
- Diseñar e implementar scripts para extracción de datos, limpieza, transformación, carga y actualización
- Poblar el repositorio con la definición del esquema y vistas, scripts y otros metadatos
- Diseño e implementación de aplicaciones de usuario final
- Uso de las aplicaciones y Data Warehouse en un ambiente productivo

4.1.7 Modelo de datos Multidimensional

La vista multidimensional de los datos en un Data Warehouse es un modelo conceptual popular que influye en las herramientas front-end, diseño de Bases de Datos, y las consultas para servidores OLAP. En un modelo de datos multidimensional, existen un grupo de medidas numéricas que son los objetos de análisis. Ejemplos de tales medidas son ventas, presupuesto, inventario, ingresos, etc. Cada una de estas medidas numéricas depende de un grupo de dimensiones, tales como proporcionar el contexto para la medida.

Por ejemplo, las dimensiones asociadas con una cantidad de venta pueden ser la ciudad, el nombre del producto y la fecha cuando la venta fue realizada. Las dimensiones juntas son asumidas a únicamente determinar la medida. De este modo, los datos multidimensionales ven una medida como un valor en el espacio multidimensional de dimensiones. Cada dimensión es descrita por un grupo de atributos.

Y la dimensión del producto puede consistir de cuatro atributos: la categoría, la industria del producto, año de su introducción, y el promedio margen de ganancias. Por ejemplo, el refresco X pertenece a la categoría de bebidas y la industria de alimentos, fue introducida en 1996, y puede tener un promedio de ganancia del 80%. Los atributos de una dimensión pueden estar relacionados vía una jerarquía de relaciones. En el ejemplo anterior, el producto es relacionado a los atributos categoría e industria a través de tal relación jerárquica.

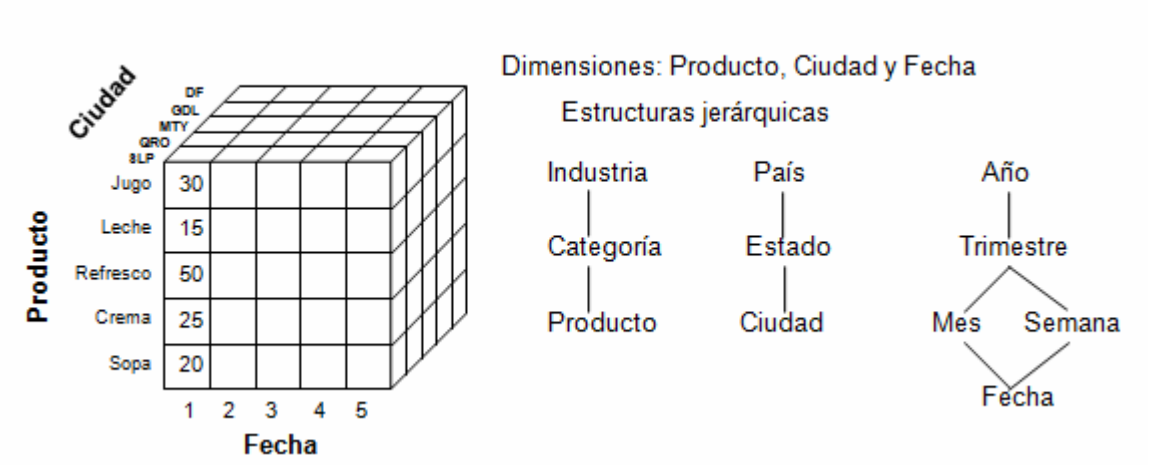


Figura 4-2 Modelo de datos Multidimensional

4.1.8 Metodología de diseño

El modelo de datos multidimensional descrito anteriormente es implementado directamente por los servidores MOLAP. Sin embargo, cuando un servidor relacional ROLAP es utilizado, el modelo multidimensional y sus operaciones tienen que ser asignadas dentro de relaciones en las consultas SQL.

Muchos Data Warehouse utilizan el esquema de estrella para representar el modelo de datos multidimensional el cual es formado por una tabla de hechos (Fact table) y una tabla para cada dimensión. Cada tupla en la tabla Fact consiste de un puntero (llave foránea) a cada una de las dimensiones que proporcionan sus coordinadas multidimensionales, y almacena las medidas numéricas para estas coordenadas. Cada tabla de dimensión consiste de columnas que corresponden a los atributos de la dimensión.

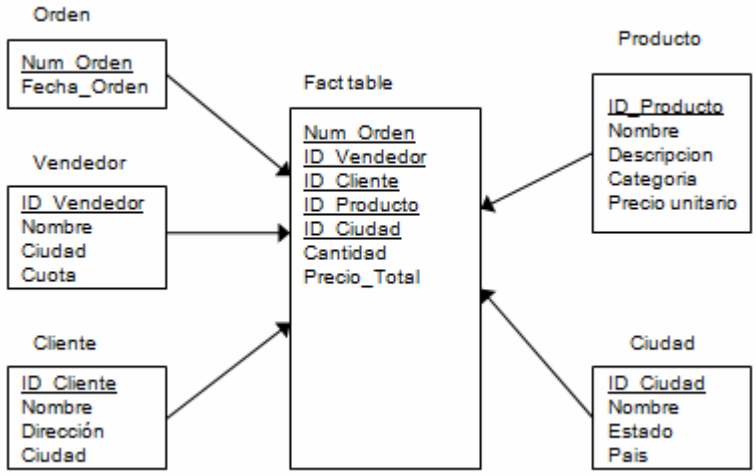


Figura 4-3 Esquema de Estrella

Conclusiones

El trabajo desarrollado en esta tesina ha constituido en presentar un panorama general en cuanto a la administración de las Bases de Datos tratando de proporcionar las herramientas básicas necesarias para una adecuada administración.

Las organizaciones se ven beneficiadas al implementar Bases de Datos dentro de sus procesos, ya que permiten optimizar el tiempo invertido en la administración de información, además de asegurar las características que le dan valor.

Los modelos y métodos matemáticos son de gran utilidad para desarrollar esquemas de trabajo y representaciones de la realidad, como es el caso de las Reglas de Codd, las cuales al ser producto de un riguroso análisis matemático son sumamente solidas y confiables.

Como administrador de Bases de Datos es indispensable conocer el lenguaje SQL, ya que es un lenguaje estándar, y por lo tanto la mayoría de los RDBMS lo implementan. El Lenguaje SQ es de gran valor tanto para los administradores de Bases de Datos como para los desarrolladores de software debido a que proporciona los elementos necesarios para realizar consultas simples que involucran una sola tabla, o bien consultas complejas que involucran a dos ó más tablas.

ASE es un RDBMS ya que se alinea a la reglas de Cood y hace posible administrar de manera eficiente las Bases de Datos.

Es cierto que el implementar un Data Warehouse puede resultar una tarea complicada y costosa, pero los beneficios que las organizaciones pueden obtener son notables ya que reducen en gran cantidad el tiempo que se invierte en generar los reportes que sirven para la toma de decisiones.

Podemos concluir que el uso de las Bases de Datos facilita la administración de la información, y para que esto sea posible y explotar al máximo todos los beneficios que ofrecen las Bases de Datos, es necesario contar con personal capacitado y especializado en el RDBMS que se esté utilizando; Por lo cual es indispensable actualizarse constantemente en cuanto al tema de Bases de Datos ya que evoluciona tan rápidamente como la tecnología.

Bibliografía

Ullman, Jeffrey y Widom, Jennifer, *Introducción a los Sistemas de Bases de Datos*. Editorial Prentice Hall, 1991.

Schiller, Bradley R, *Fundamentos de Bases de Datos*. Editorial McGraw-Hill, 2001.

Adoración de Miguel y Mario Piattini, *Bases de Datos Relacionales*. Editorial RAMA, 1999.

Piattini Velthuis, Mario Gerardo, *Tecnología y Diseño de Bases de Datos*. Editorial Alfaomega – RAMA, 2007.

Rozic, Sergio Ezequiel, *Bases de Datos y su Aplicación con SQL*. Mp Ediciones Sa, 2004.

Hansen, Gary W y Hansen, James, *Diseño y Administración de Bases de Datos*, Editorial Prentice Hall,1996.

Wiley, **John***, Building the Data Warehouse*. Editorial Inmon, 1992.

Wiley, **John** *The Data Warehouse Toolkit*. Editorial Kimball, R.,1996.

Fuentes de Información

Wikipedia. *Reglas de Cood*. http://es.wikipedia.org/wiki/RDBMS Abril 2009.

Instituto Tecnológico de Colima, Departamento de Sistemas y Computación. *Tutorial de Fundamentos de Bases de Datos.*
http://labredes.itcolima.edu.mx/fundamentosbd/sd_u2_1.htm Enero 2007

SyBooks Online. *System Administration Guide: Adaptive Server Enterprise 15.0*
http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.dc35130_0400/html/SySearch_rb/title.htm Octubre 2005.

Surajit Chaudhuri & Umeshwar Dayal, *An Overview of Data Warehousing and OLAP Technology. http://research.microsoft.com/apps/pubs/?id=76058 Marzo 1997.*