**Unidad 1**

**Base de Datos: Colección de datos interrelacionados almacenados en conjunto, sin redundancias perjudiciales o innecesarias,** su finalidad es **servir a una aplicación o más**, de la mejor manera posible; los datos se almacenan de forma que resulten independientes de los programas que los usan; se emplean métodos bien determinados para incluir nuevos datos y para modificar o extraer los datos almacenados

**Dato: son los hechos conocidos que se pueden grabar y que tienen un significado implícito**.

**Una base de datos representa algún aspecto del mundo rea**l, lo que en ocasiones se denomina **minimundo o universo de discurso**. Los cambios introducidos en el minimundo se reflejan en la base de datos.

**Un sistema de administración de datos DBMS:** Es una colección de programas que permite a los usuarios crear y mantener una base de datos. El DBMS es un sistema de software de propósito general que facilita los procesos de **definición**, **construcción**, **manipulación**, y **compartición** de base de datos entre varios usuarios y aplicaciones.

* **Definir** una base de datos implica especificar los tipos de datos , estructuras y restricciones de los datos que se almacenaran en la base de datos. La definición o información descriptiva de una base de datos también se almacena en esta última en forma de catálogo o diccionario de la base de datos, llamada **metadato**.
* **Construcción** de la base de datos es el proceso consistente en almacenar los datos en algún medio de almacenamiento controlado por el DBMS.
* **Manipulación** de una base de datos incluye funciones como la consulta de la base de datos para recuperar datos específicos, actualizar la base de datos para reglejar los cambios introducidos en el minimundo y generar informes a partir de los datos
* **Compartir**, una base de datos permite que varios usuario y programas accedan a la base de datos de forma simultánea.

Una **aplicación** accede a la base de datos enviando consultas o solicitudes de datos al DBMS.

Una **consulta** normalmente provoca la recuperación de algunos datos.

Una **transacción** puede provocar la lectura o la escritura de algunos datos de forma simultánea.

**Protección:** incluye la protección del sistema contra el funcionamiento defectuoso del hardware o el software, y la protección de la seguridad contra el acceso no autorizado o malintencionado.

**Mantenimiento:** Una gran base de datos típica puede tener un ciclo de vida de muchos años, por lo que el DBMS debe ser capaz de mantener el sistema de base de datos permitiendo que el sistema evolucione según cambias los requisitos con el tiempo.

**Un sistema de base de datos es la combinación de Hardware (Servidores, etc.), Software(DBMS), Usuarios(DBA, usuarios finales), Datos(tablas, relaciones).**

**Características de la metodología de bases de datos**

En la metodología de bases de datos se mantiene un único almacén de datos, que se define una sola vez y a la que acceden varios usuarios.

Las principales características de la metodología de bases de datos frente a la metodología del procesamiento de archivos son las siguientes:

1. **Naturaleza auto descriptiva de un sistema de base de datos.**

**Es una característica fundamental el que el sistema de base de datos no solo contiene la propia base de datos sino una completa definición de la estructura de la base de datos y sus restricciones. Esta definición se almacena en el catalogo DBMS**, que contiene información como la estructura de cada archivo, el tipo y el formato de almacenamiento de cada elemento de datos y distintas restricciones de los datos**. La información almacenada en un catálogo se denomina metadatos y describe la estructura de la base de datos principal.**

El software DBMS y los usuarios de la base de datos utilizan el catalogo cuando necesitan información sobre la estructura de la base de datos.

1. **Aislamiento entre programas y datos, abstracción de los datos.**

Los programas que acceden a un DBMS no necesitan cambios en la estructura de programa, en la mayoría de los casos. La estructura de los archivos de datos se almacena en el catalogo DBMS, independientemente de los programas de acceso, esto se denomina  **independencia de programa/datos.**

En un sistema orientado a objetos, los usuarios pueden definir operaciones sobre los datos como parte de las definiciones de la base de datos. Una Operación se puede especificar de dos formas: la interfaz de una operación, incluye el nombre de la operación y los tipos de datos de sus argumentos y la implementación de la operación se especifica separadamente y puede modificarse sin que la interfaz se vea afectada. Las aplicaciones de usuario pueden operar sobre los datos invocando estas operaciones por sus nombres y argumentos, independientemente de cómo estén implementadas las operaciones. **Esto se denomina independencia programa operación**. Las características que permite la independencia programa-datos y la independencia programa-operación se denomina **abstracción de datos**.

Un DBMS proporciona a los usuarios una representación conceptual de los datos que no incluye muchos de los detalles de cómo están almacenados los datos o de cómo están implementadas las operaciones. **Un modelo de datos es un tipo de abstracción de datos se utiliza para proporcionar esa representación conceptual**. El modelo de datos utiliza conceptos lógicos, como los conceptos de almacenamientos. Por ello, el modelo de datos oculta los detalles del almacenamiento y de la implementación que no resultan interesantes a la mayoría de los usuarios de base de datos.

El proceso de abstracción no solo incluye la estructura de datos sino también las operaciones sobre los datos, proporcionando una abstracción de las actividades del minimundo normalmente entendidas por los usuarios.

1. **Soporte para varias vistas de Datos.**

Una vista puede ser un subconjunto de la base de datos o puede contener datos virtuales derivados de los archivos de la base de datos pero que no están explícitamente almacenados.

1. **Compartición de datos y procesamiento de transacciones multiusuarios.**

**Un DBMS multiusuario, debe permitir que varios usuarios puedan acceder a la base de datos al mismo tiempo**. Esto es esencial si los datos destinados a varias aplicaciones serán integrados y mantenidos en una solo a base de datos. El DBMS debe incluir software de control de concurrencia para que esos varios usuarios que intentan actualizar los mismos datos, lo hagan de un modo controlado para que el resultado de la actualización sea correcto. Estos tipos de aplicaciones se denominan aplicaciones de procesamiento de transacciones en línea OLTP.

**Una transacción es un programa en ejecución o un proceso que incluye uno o más accesos a la base de datos, como la lectura o la actualización de los registros de la misma**. El DBMS debe implementar varias propiedades de transacción, como la de **aislamiento** que garantiza que parezca que cada transacción se ejecuta de forma aislada de otras transacciones y la de **atomicidad** que garantiza que se ejecuten o todas o ninguna de las operaciones de bases de datos de una transacción.

**Actores**

1. **Administradores de Base de Datos:** El administrador de los recursos es denominado DBA, el DBA es el responsable del acceso autorizado a la base de datos, de la coordinación y monitorización de uso, y de adquirir los recursos software y hardware necesarios. El DBA también es responsable de problemas como la brechas de seguridad o de unos tiempos de respuesta pobres.
2. **Diseñadores de Base de Datos:** Son los que identifican que datos serán los que se almacenaran en la base de datos. Y de elegir las estructuras apropiadas para representar y almacenar esos datos. Debe comunicarse con los principales usuarios de la base de datos para conocer sus requisitos.
3. **Usuarios Finales:**

* **Usuarios finales casuales**: acceden ocasionalmente, pueden necesitar información diferente en cada momento, usan sofisticado lenguaje de consulta, son administradores de nivel medio.
* **Usuarios finales principiantes o paramétricos**: su labor principal giran en torno a la consulta y actualización constantes de la base de datos, utilizando tipos de consultas y actualizaciones estándar que se han programado y probado.(transacciones enlatadas)
* **Usuarios finales sofisticados**: se encuentran los ingenieros, los científicos, los analistas comerciales y los otros muchos que están familiarizados con el DBMS a fin de implementar sus aplicaciones.
* **Usuarios finales independientes**: mantienen bases de datos personales utilizando paquetes de programas confeccionados que proporcionan unas interfaces fáciles de usar.

1. **Analistas de sistemas y programadores de aplicaciones:** Los analistas de sistemas determinan los requisitos de los usuarios finales, especialmente de los usuarios finales principiantes y paramétricos, así como las especificaciones de desarrollo para las transacciones en latadas que satisfacen esos requisitos. Los programadores de aplicaciones implementan esas especificaciones.

Ventajas de la Tecnología de DBMS

* **Control de redundancia**: la redundancia resultante de almacenar los mismos datos varias veces conduce a serios problemas, como actualizaciones constantes, lo que lleva a mucho esfuerzo y tiempo, o llegar a un grado de inconsistencia cuando no se han realizado las tareas de actualización adecuadamente.
* **Restricción de acceso no autorizado**: un DBMS debe proporcionar seguridad y un subsistema de autorización, que el DBA utiliza para crear cada cuenta y especificar las restricciones de las mismas. Después el DBMS debe implementar automáticamente esas restricciones.
* **Almacenamiento persistente para los objetos del programa**: las bases de datos se pueden utilizar para proporcionar almacenamiento persistente a los objetos de programa y las estructuras de datos, siendo estos los orientados a objetos.
* **Suministro de estructuras de almacenamiento para un procesamiento eficaz de las consultas**: los sistemas de base de datos deben proporcionar capacidades para ejecutar eficazmente consultas y actualizaciones. Para ello archivos auxiliares denominados índices, que están basados casi siempre en el árbol de estructuras de datos o en las estructuras de datos dispersas, convenientemente modificados para la búsqueda en disco. El modulo de procesamiento de optimización de consultas del DBMS es el responsable de elegir un plan eficaz de ejecución de consultas para cada consulta basándose en la estructuras de almacenamiento existentes.
* **Copia de seguridad y recuperación**: un DBMS debe ofrecer la posibilidad de recuperarse ante fallos del hardware o del software. El subsistema de copia de seguridad y recuperación del DBMS es el responsable de la recuperación.
* **Suministro de varias interfaces de usuario:** un DBMS debe proporcionar varias interfaces para diferentes usuarios: Interfaz de consulta, interfaz de programación, formulario y código de comandos, interfaces por menús.
* **Representación de relaciones complejas entre los datos:** un DBMS debe ser capaz de representar las relaciones complejas entre los datos, definir las nuevas relaciones que surgen, y recuperar y actualizar fácil y eficazmente los datos relaciones.
* **Implementación de las restricciones de integridad:** restricciones de integridad mas simple es especificar un tipo de dato por cada elemento de datos, otro tipo especifica la unicidad en los valores del elemento de datos. Otras restricciones pueden ser las reglas de negocio.
* **Inferencia y acciones usando reglas**: algunos sistemas ofrecen la posibilidad de definir reglas de deducción para inferir información nueva a partir de los hechos guardados en la base de datos. Estos sistemas se denominan sistemas de base de datos deductivos. Los triggers y procedimientos almacenados.
* **Implicaciones adicionales:**
  + - **Potencial para implementar estándares:** la metodología de base de datos permite al DBA definir e implementar estándares entre los usuarios de la base de datos en una empresa grande.
    - **Tempo de desarrollo de aplicaciones reducido:** se necesita muy poco tiempo para desarrollar una aplicación nueva.
    - **Flexibilidad:** los DBMS modernos permiten ciertos tipos de cambios evolutivos en la estructura de la base de datos.
    - **Disponibilidad de información actualizada**
    - **Economías de escala.**

**Conceptos y arquitectura de los sistemas de bases de datos**

**Modelo de datos, esquemas e instancias**

**La abstracción de datos** se refiere generalmente a la supresión de detalles de la organización y el almacenamiento de datos y a la relevancia de las características fundamentales para un conocimiento mejorado de los datos.

**Un modelo de datos** proporciona los medios necesarios para conseguir esa abstracción.

**Estructura de una base de datos** se refiere a los tipos de datos, relaciones y restricciones que deben mantenerse para los datos. La mayoría de los modelos de datos incluye un conjunto de operaciones básicas para especificar las recuperaciones y actualizaciones en la base de datos. Además de las operaciones básicas, se incluyen conceptos en el modelo de datos para especificar aspecto dinámico o de comportamiento de una aplicación de base de datos (en modelo de base de datos relacional son los procedimientos almacenados).

**Modelo de Datos 🡪 Colección de Conceptos 🡪 describe una estructura de Base de Datos**

**Categoría de Modelo de Datos**

1. **Los modelos de datos conceptuales**, los modelos de alto nivel o conceptuales ofrecen conceptos muy cercanos a como muchos usuarios perciben los datos, llevan los conceptos de **entidades**, **atributos** y **relaciones**.
2. **Los modelos de datos representativos o de implementaciones**, ofrecen conceptos que los usuarios finales pueden entender pero que no están demasiado alejados de cómo se organizan los datos; son los más utilizados en los DBMS comerciales y tradicionales. **Representan los datos mediante estructuras de registro** y por tanto se los conoce a veces como modelos de datos basados en registros.
3. **Los modelos de datos físicos** ofrecen conceptos que describen los detalles de cómo se almacenan los datos en el computador. Los conceptos ofrecidos por los modelos de datos de bajo nivel están pensados principalmente para los especialistas en computadoras, no para usuarios finales normales, describen como se almacenan los datos en el computador en forma de archivos, representado la información como formatos de **registro**, ordenación de registro y rutas de acceso.

**Esquema, instancias y estado de la base de datos**

En cualquier modelo de datos es importante distinguir entre la **descripción** de la base de datos y la misma base de datos. **La descripción de una base de datos se denomina esquema de base de datos (el DBMS lo almacena en el catálogo)**, que se especifica durante la fase de diseño y no se espera que cambie con frecuencia. La mayoría de los modelos de datos tienen ciertas convenciones para la visualización de los esquemas a modo de diagramas. Un esquema visualizado se denomina  **diagrama del esquema.**

Los datos de una base de datos en un momento concreto se denominan **estado de la base de datos** o **snapshot**, o **conjunto actual de ocurrencias o instancias de la base de datos**. En un estado dado de la base de datos, cada estructura de esquema tiene su propio conjunto actual de instancias. (**El DBMS debe garantizar que cada estado de la base de datos sea un estado válido**). Cuando se define una nueva base de datos, solo se especifica el esquema al DBMS, el estado de la base de datos es el *estado vacio*, el estado inicial se da cuando ésta se rellna por primera vez. El esquema recibe el nombre de intención, y el estado de la base de datos extensión del esquema.

**Arquitectura de tres esquemas e independencia de los Datos**

El objetivo de la arquitectura de 3 esquemas es separar las aplicaciones de usuarios y las bases de datos físicas.

1. **Nivel Interno, tiene un esquema interno**, que describe la estructura de almacenamiento físico de la base de datos. **Utiliza un modelo de datos físico** y describe todos los detalles de almacenamiento de datos y las rutas de acceso a la base de datos.
2. **Nivel Conceptual**, **tiene un esquema conceptual**, describe la estructura de toda la base de datos para una comunidad de usuarios. El esquema conceptual oculta los detalles de la estructuras de almacenamiento físico y se concentran en describir las entidades, los tipos de datos, las relaciones, las operaciones de los usuarios y las restricciones. El esquema conceptual **se describe con un modelo de datos representativo** cuando se implementa un sistema de bases de datos. Este esquema conceptual de implementación se basa a menudo ene un diseño de esquema conceptual en el modelo de datos de alto nivel.
3. **Nivel de vista o externo**, tiene **una cierta cantidad de esquemas externos** o vistas de usuario, describe la parte de la base de datos en la que un grupo de usuarios en particular está interesado y le oculta el resto de la base de datos. Como en el caso anterior, cada esquema externo se implementa normalmente mediante un modelo de datos representativo, posiblemente basado en un diseño de esquema externo de un modelo de datos de alto nivel.

Los 3 esquemas son solo descripciones de datos, los datos que existen en realidad están en el nivel físico. Cada grupo de usuarios solo se refiere a su propio esquema externo. El DBMS debe transformar una solicitud especificada en el esquema externo en una solicitud contra el esquema conceptual y esta a su vez en una solicitud contra el esquema interno. Estos procesos de conversión de solicitudes se denominan **mapeados**.

**Independencia de los Datos**

**La independencia de los datos es la capacidad de cambiar el esquema de un nivel de un sistema de bases de datos sin tener que cambiar el esquema en el siguiente nivel más alto**.

1. **Independencia lógica de datos**: es la capacidad de cambiar el esquema conceptual sin tener que cambiar los esquemas externos o los programas de aplicación. Eso puede ser para expandir la base de datos, cambiar restricciones, o para reducir la base de datos. Solo es necesario cambiar la definición de la vista y los mapeados en un DBMS que soporta independencia lógica de datos.
2. **Independencia física de los datos**: es la capacidad de cambiar el esquema interno sin que haya que cambiar el esquema conceptual. Por lo tanto, tampoco es necesario cambiar esquemas externos. Puede que haya que realizar cambios en el esquema interno porque algunos archivos físicos fueran reorganizados de cara a mejorar el rendimiento de las recuperaciones o las actualizaciones. Si en la base de datos permanecen los mismos datos que antes, no hay necesidad de cambiar el esquema conceptual. Esto existe en la mayoría de las base de datos y de los entornos de archivo en los que al usuario se le ocultan la ubicación exacta de los datos en el disco, los detalles de hardware de la codificación del almacenamiento, la colocación, la compresión, la división, la fusión de registros, etc.

**Lenguajes e interfaces de bases de datos**

**Lenguajes DBMS:**

Una vez realizado el diseño, se debe continuar con los esquemas conceptual e interno y el mapeado entre los dos. En DBMS donde no hay una separación estricta de niveles, el DBA utiliza el DDL, lenguaje de definición de datos, para definir los dos esquemas. El DBMS deberá tener un compilador DDL para almacenar las descripciones en el catalogo.

En DBMS donde hay una clara separación de niveles, se utiliza solo DDL para el esquema conceptual, y para esquema interno se utiliza SDL, lenguaje de definición de almacenamiento. El mapeado se puede especificar en cualquiera de los dos lenguajes. Para conseguir una arquitectura de 3 niveles, se necesita un tercer lenguaje para definir las vistas, DVL, lenguaje de definición de vistas, especifica las vistas del usuario y el mapeado al esquema conceptual.

DML, lenguaje de manipulación de datos, es utilizado para realizar las operaciones básicas sobre los datos almacenados.

Hay dos tipos principales de DML, el DML de alto nivel o no procedimental para especificar de forma concisa las operaciones complejas con las bases de datos mediante introducción interactiva desde el monitor o terminal. Un DML de bajo nivel debe incrustarse en un lenguaje de programación de propósito general, normalmente recupera registros individuales y los procesa por separado.

**Entorno de un sistema de bases de datos**

**Módulos componentes de un DBMS**

Las bases de datos y el catalogo del DBMS normalmente se almacenan en disco. El acceso al disco esta principalmente controlado por el sistema operativo. Un modelo administrador de los datos almacenados de alto nivel del DBMS controla el acceso a la información del DBMS almacenada en el disco, sea parte de la base de datos o del catálogo.

El personal DBA utiliza sentencias DDL y otros comandos privilegiados, el compilador DDL procesa las definiciones de esquema, especificadas en el DDL, y almacena las descripciones de los esquemas en el catalogo (nombre y tamaños de los archivos, información de mapeado entre esquemas y las restricciones) del DBMS.

Los usuarios casuales interactúan con la interfaz de consulta interactiva. Un compilador de consultas analiza estas consultas sintácticamente para garantizar la corrección de las operaciones de los modelos. El optimizador de consultas se ocupa de la reconfiguración y la posible reordenación de las operaciones y genera un código ejecutable que lleva a cabo las operaciones necesarias para la consulta y realiza las llamadas al procesador runtime.

Los programadores de aplicaciones, escriben programas en un determinado lenguaje y son llevados a un precompilador, éste extrae los comandos DML y se los envía al compilador DML generando un código objeto, y así poder acceder a la base de datos. Los comandos DML compilados junto con el resto del programa se enlazan, formando una transacción enlatada cuyo código ejecutable incluye las llamadas al procesador de base de datos runtime.

El procesador de base de datos runtime, trabaja con el diccionario del sistema y se puede actualizar con estadísticas. Funciona con el administrador de datos almacenados, que a su vez utiliza servicios básicos del sistema operativos para ejecutar operaciones de entrada y salida de bajo nivel entre el disco y la memoria principal. Se encarga de otros aspectos de la transferencia de datos como la administración de los búferes en la memoria principal.

Procesador de BD runtime: recibe comandos DBA, consultas y transacciones enlazadas con programas y se encarga de ejecutarlas accediendo a el diccionario de datos y comunicándose con el S.O. para efectuar las operaciones de entrada / salida de información. También se encarga de la administración del buffer en la memoria de procesamiento, esta función puede o no compartirla con el S.O.

**Almacenamiento en Disco**

La colección de datos que constituye una base de datos computarizada debe almacenarse físicamente en un medio de almacenamiento. Los medios de almacenamiento  **se categorizan jerarquicamente:**

1. **Almacenamiento Principal**: medios en los cuales el CPU puede operar, como la memoria principal o la caché. Proporciona un acceso rápido a los datos, pero tiene capacidad de almacenamiento limitada.  
   En el extremo más alto se encuentra la memoria caché, que es normalmente utilizada por el CPU.  
   Le sigue la memoria DRAM, denominadas memoria principal, es de bajo costo, pero su desventaja es la volatilidad y baja velocidad comparada a la caché.
2. **Almacenamiento Secundario**: se refiere a los discos magnéticos, ópticos, y las cintas. Las unidades de disco duro se consideran como secundarias, mientras que las unidades ópticas, medios removibles y cintas se consideran como terciarias.

Los programas residen y se ejecutan en la DRAM, por lo que las bases parmente mas grandes residen en el almacenamiento secundario y las porciones de las bases de datos que se leen y se escriben en los búferes de la memoria principal según se necesitan.

**Almacenamiento de Base de Datos:** Las bases de datos almacenan grandes cantidades de datos que pueden persistir largos periodos de tiempo. La mayoría de las bases de datos se almacenan en almacenamiento secundario de discos magnéticos por las siguientes razones:

* Son demasiado grandes para entrar en la memoria principal
* Menos frecuencia de pérdida de datos en sistemas de almacenamiento secundario
* Es costo de mantenimiento por unidad de datos es de un orden de magnitud inferior.

Las cintas magnéticas se utilizan con frecuencia como medio de almacenamiento para hacer copias de seguridad de las base de datos, porque el almacenamiento en cinta es mucho menos costoso que el de disco. Pero el acceso a los datos es mucho más lento, acceso offline.

El proceso de diseño fisco de la base de datos implica elegir las técnicas de organización de datos particulares que mejor se adapten a los requisitos de la aplicación.

Las aplicaciones típicas de bases de datos solo necesitan procesar una pequeña porción de la base de datos en cada momento. Siempre que se necesita una determinada porción de datos, debe buscarse en el disco, copiarse en la memoria principal para su procesamiento y, después, reescribirse en el disco si los datos han sido cambiados. Los datos en el disco se organizan como **ficheros de registros**. Un **registro** es una colección de valores de datos que pueden interpretarse como hechos relacionados con las entidades, sus atributos y sus relaciones. Los registros deben almacenarse en disco de un modo que haga posible su localización de forma más eficaz.

**Ubicación de los registros de fichero**

**Registros y tipos de Registros**

Los datos se almacenan en forma de registros. Un registro consta de una colección de valores o elementos de datos relacionados, donde cada valor está formado por uno o más bytes y corresponde a un campo concreto del registro. Los registros normalmente describen entidades y sus atributos.

Una colección de nombres de campos y sus correspondientes tipos de datos constituyen un tipo de registro. Un tipo de dato, asociado a cada campo, especifica los tipos de valores que un campo puede tomar.

El tipo de dato de un campo normalmente es uno de los tipos de datos estándar que se utilizan en programación; numérico, cadenas de caracteres, booleanos, etc.

**Ficheros, registros de longitud fija y registros de longitud variable.**

Un fichero es una secuencia de registros. En muchos casos, todos los registros de un fichero son del mismo tipo de registro. Si cada registro del fichero tiene exactamente el mismo tamaño, es un registro de longitud fija. Si en el fichero hay registros que tienen tamaños diferentes, el fichero está compuesto por registros de longitud variable. (La longitud variable se da cuando el fichero tiene tipos de registros como ser el nombre de un empleado, es variable, o campos que son opcionales, etc.).

Cada registro tiene los mismos campos, y las longitudes de campo son fijas, por lo que el sistema puede identificar la posición del byte inicial de cada campo relativa a la posición inicial del registro. Esto facilita la localización de los valores de campo por parte de los programas. (solo registros de longitud fija)

**Bloque de Registros**

**Registro extendidos y Registros no extendidos.**

Los registros de un fichero deben asignarse a bloques de disco porque un bloque es la unidad de transferencia de datos entre el disco y la memoria. Cuando el tamaño del bloque es mayor que el tamaño del registro, un bloque puede contener varios registros, aunque algunos ficheros pueden tener registros que por su tamaño no entran en un bloque, por lo tanto, para utilizar el espacio desaprovechado se puede almacenar parte de un registro en un bloque y el resto en otro. Un puntero al final del primer bloque apunta al bloque que contiene el resto del registro, en caso de que no sean bloques consecutivos en el disco. Esta organización se denomina **extendida**. Si los registros no tienen permitido sobrepasar el tamaño del bloque, se denomina  **no extendida.**

**Asignación de bloques de ficheros en un disco**

En una asignación continua, los bloques del fichero se asignan a bloques de discos consecutivos. De este modo, la lectura del fichero entero es mucho más rápida mediante el buffer doble, pero se complica para añadir ficheros. En una asignación enlazada, cada bloque del fichero contiene un puntero al siguiente bloque del fichero. Esto hace más sencilla la expansión del fichero, pero más lenta la lectura del fichero entero. Una combinación de las dos asigna clústeres de bloque de disco consecutivos, y los clústeres se enlazan. En ocasiones, los clústeres se les denominan segmentos de ficheros o extensiones.

Una cabecera de fichero o descriptor del fichero contiene información sobre un fichero que los programas del sistema necesitan para acceder a los registros. La cabecera incluye información para determinar las direcciones de disco de los bloques del fichero y las descripciones de formato de registro.

1. **Fichero de registro desordenado**

Es el tipo de organización más sencillo y básico, según el cual los registros se guardan en el fichero en el mismo orden en que se insertan, es decir, los registros se insertan al final del fichero. Esta organización se conoce como fichero heap o pila.

La inserción de un registro nuevo es muy eficaz. El último bloque del disco del fichero se copia en el búfer se añade el registro nuevo y se reescribe el bloque de nuevo en el disco. La búsqueda de un registro, implica una búsqueda lineal, bloque a bloque por todo el fichero. Para eliminar un registro, primero debe buscar su bloque, copiar el bloque en un búfer, eliminar el registro del búfer, y por último reescribir el bloque en el disco, dejando espacio inutilizado.

La ordenación es una operación costosa si el fichero es grande, para ello se utilizan técnicas especiales de ordenación externa.

En un fichero desordenado de registro de longitud fija que utiliza bloques extendidos y la asignación contigua, es inmediato el acceso a cualquier registro por su posición en el fichero.

1. **Fichero de registro ordenado**

Los registros de fichero se pueden ordenar físicamente en el disco en función de los valores de uno de sus campos, denominado campo de ordenación, esto conduce a un fichero ordenado o secuencial. Si el campo de ordenación también es un campo clave (garantiza un valor único en cada registro), entonces el campo se denomina clave de ordenación del fichero. La ventaja de los ordenados, es la lectura es eficaz. Segundo, ventaja es encontrar el siguiente registro al actual según el orden de la clave ordenación normalmente no requiere acceder a bloques adicionales porque el siguiente registro se encuentra en el mismo bloque que el actual. Tercero, el uso de una condición de búsqueda basándose en el valor del campo clave de ordenación ofrece un acceso rápido.

La inserción o eliminación es costosa, ya que siempre deben permanecer los registros ordenados físicamente. Una manera de facilitar la inserción es reservar algo de espacio sin usar en cada bloque de cara a los registros nuevos. Otra técnica es utilizar el modo de desbordamiento.

Una opción para que la inserción sea eficaz, es dejar especio sin usar en cada bloque de cara a los registros nuevos. Otro método es crear un fichero temporal desordenado denominado fichero de desbordamiento. El fichero ordenado actual se denomina fichero principal o maestros. Los registros nuevos se insertan al final del fichero de desbordamiento. Periódicamente el fichero de desbordamiento se ordena y mezcla con el maestro durante la reorganización del fichero.

1. **Técnicas de Dispersión**.

Proporciona un acceso muy rápido a los registros bajo ciertas condiciones de búsqueda. La condición de búsqueda debe ser una condición de igualdad sobre un solo campo, denominado campo de dispersión. Este campo también es un campo clave del fichero, en cuyo caso se nomina clave de dispersión. La idea es proporcionar una función *h*, denominada función de dispersión, que aplica el valor del campo de dispersión de un registro y produce la dirección del bloque de disco en que está almacenado el registro.

**Estructuras de Indexación para los ficheros**

Las estructuras de índice proporcionan rutas de acceso, que ofrecen formas alternativas de acceder a los registros sin que se afecte la ubicación física de los registros en el disco. Es un acceso eficaz a los registros basándose en la indexación de los campos que se utilizan para construir el índice. Es posible acceder a cualquier campo del fichero para crear un índice y se pueden construir varios índices con diferentes campos en el mismo fichero.

**Tipos de Índices**

**Índices de un nivel.**

**Fichero Índice**: Normalmente se define una estructura de índice con un solo campo del fichero, que se conoce como **campo de indexación**. El índice almacena todos los valores del campo de índice, junto con una lista de punteros a todos los bloques de disco que contienen los registros con ese valor de campo. Los valores de índice están ordenados de manera que se puede hacer una búsqueda binaria. El fichero de índice es mucho mas pequeño que un fichero de datos, por lo que la búsqueda en el índice mediante una búsqueda binaria es razonablemente eficaz.

Hay varios tipos de índices ordenados:

Un índice principal o primario se especifica en el campo clave de ordenación de un fichero ordenado de registros. Se utiliza un campo clave de ordenación para ordenar físicamente los registros del fichero en el disco, y cada registro tiene un valor de ordenación único para ese cambio.  
Si ese campo de ordenación no es campo clave, es decir que varios ficheros tiene el mismo valor para el campo de ordenación, se utiliza otro tipo de índice denominado índice agrupado.  
Un fichero puede tener como máximo un campo de ordenación física, por lo que puede tener un índice principal o un índice agrupado, pero no ambos.   
Un tercer índice es el índice secundario, que se puede especificar sobre cualquier campo no ordenado del fichero. Un fichero puede tener varios índice secundarios además de su método de acceso principal.

1. **Índice Principal:** es un fichero ordenado cuyos registros son de longitud fija con dos campos. El primer campo es del mismo tipo de datos que el campo clave de ordenación del fichero de datos, el segundo campo es un puntero a un bloque del disco. En el fichero índice hay una entrada de índice por cada bloque del fichero de datos. Cada entrada del índice tiene dos valores, el valor del capo clave principal para el primer registro de un bloque, y un puntero a ese bloque. El primer registro de cada bloque del fichero de datos es el registro ancla del bloque o simplemente ancla de bloque.  
   Los índices se pueden clasificar:
   1. **Denso**: tiene una entrada de índice por cada valor de clave de búsqueda del fichero de datos.
   2. **Escaso o No Denso**: solo tiene entradas para algunos de los valores de búsqueda.

Un índice principal es un índice escaso, porque incluye una entrada por cada bloque de disco del fichero de datos y las claves de su registro ancla. El 1er registro de cada bloque  
La inserción y el borrado de registros supone un gran problema, ya que para insertar un registro se debe mover los registros y hacer un lugar para el nuevo y luego cambiar las entradas a los índices. El uso de un fichero de desbordamiento puede minimizar el problema

1. **Índice Agrupado**: si los registros están ordenados fiscalmente por un campo no clave, este campo se denomina campo agrupado. Un índice agrupado también es un fichero ordenado con dos campos, el primero es del mismo tipo que el campo agrupado del fichero de datos, y el segundo es un puntero a un bloque. En el índice agrupado hay una entrada por cada valor distinto del campo agrupado, que contiene el valor y un puntero al primer bloque del fichero de datos que tienen un registro con ese valor para su campo agrupado. Este es otro ejemplo de un índice no denso porque tiene una entrada por cada valor distinto del campo de indexación, que no es una clave por definición, y que por lo tanto, tiene valores duplicados en lugar de un valor único por cada registro del fichero.

La inserción y el borrado provocan problemas ya que los registros están ordenados físicamente. Para alivianar el problema se suele reservar un bloque entero para cada valor del campo agrupado. Todos los regiros con esos valores se agrupa en un mismo bloque.

1. **Índices Secundarios:** proporciona un medio secundario de acceso a un fichero para el que ya existe algún acceso principal. El índice secundario puede ser un campo que es una clave candidata y que tiene un valor único en cada registro, o puede ser una no clave con valores duplicados. El índice es un fichero ordenado con dos campos. El primero es del mismo tipo de datos que algún campo no ordenado del fichero de datos que es un campo de indexación. El segundo campo es un puntero de bloque o un puntero de registro. Puede haber muchos índices secundarios.

El campo clave tiene un valor distinto por cada registro, este recibe el nombre de clave secundaria. En este caso, hay una entrada de índice por cada registro del fichero de datos, que contiene el valor de la clave secundaria para el registro y un puntero al bloque en el que está almacenado el registro. Dicho índice es denso.  
Es decir, si el índice por campo no clave los valores del campo de indexación se repiten, hay 3 opciones:

* 1. Generar un índice denso con un registro por cada valor
  2. Índice con registro de longitud variable con un campo repetitivo de puntero a bloque.
  3. Índice no denso, cada registro almacena punteros a bloque de punteros y estos apuntan a registros.

Un índice secundario proporciona una ordenación lógica de los registros por el campo de indexación. Si accedemos a los registros según el orden de las entradas del índice secundario, los obtendremos ordenados por el campo de indexación.

**Índice multinivel.**

Un índice de multinivel considera el fichero índice, al que ahora nos referimos como primer nivel o base del un índice multinivel, como un fichero ordenado con un valor distinto por cada K(i). por lo tanto podemos crear un índice principal para el primer nivel, este índice al primer nivel se lo denomina segundo nivel del índice multinivel. Como el segundo nivel es in índice principal, podemos utilizar anclas de bloque de modo que el segundo nivel tenga una sola entrada por cada bloque del primer nivel es el mismo que para el índice de primer nivel porque todas las entradas del índice tiene el mismo tamaño, cada una con valor del campo y una dirección de bloque.

El esquema multinivel puede utilizarse con cualquier tipo de índice, siempre y cuando el índice de primer nivel tenga valores distintos y entradas de longitud fija.

**Índices multinivel dinámicos utilizando arboles B u B+**

Los arboles B y B+ son casos especiales de estructuras de datos en forma de árbol bien conocidas. Un árbol está formado por nodos, un nodo que no tiene hijos, se llama hoja, el nodo raíz no tiene padre.

Un subárbol de un nodo consta de ese nodo y de todos sus nodos descendientes.

Una forma de implementar un árbol es con tantos punteros en cada nodo como nodos hijos tiene ese nodo. Un nodo normalmente contiene algún tipo de información almacenada. Cundo in índice multinivel se implementa como estructura de árbol, esa información incluye los valores del campo de indexación del fichero que se utiliza para guiar la búsqueda de un determinado registro.

**Arboles de búsqueda y Arboles B**

Podemos usar un árbol de búsqueda como mecanismo para buscar registros almacenados en un fichero de disco. Los valores del árbol pueden ser los valores de uno de los campos del registro, el llamado campo de búsqueda. Cada valor clave del árbol está asociado a un puntero de registro que tiene ese valor en el fichero de datos. El árbol de búsqueda en si puede almacenarse en disco, asignando cada nodo del árbol a un bloque del disco.

**El árbol B** tiene restricciones adicionales que garantizan que el árbol siempre estará equilibrado y que el espacio desperdiciado por la eliminación nunca será excesivo. Los algoritmos para insertar y eliminar son mas complejos para poder mantener las restricciones.

1. Algoritmos más complejos, cada valor del campo de búsqueda aparece vez con su puntero
2. Restricciones adicionales para asegurar equilibrio:
3. Cada nodo interno con forma <P1,<K1,Pr1>,P2,<K2,Pr2>,…,<Kq-1,Prq-1>,Pq>
4. Dentro de cada nodo K1<K2<…<Kq-1
5. Valores del campo clave de búsqueda X del subárbol Ki-1<X<Ki
6. Cada nodo tiene a lo sumo p punteros de árbol
7. Cada nodo tiene por lo menos p/2 punteros de árbol-Raíz tiene como mínimo 2 punteros de árbol
8. Un nodo con q punteros de árbol tiene q-1 valores del campo clave de búsqueda
9. Todos los nodos hojas están en el mismo nivel y sus punteros de árbol son nulos

**En un árbol B+** los punteros a datos se almacenan solo en los nodos hoja del árbol, la estructura de los nodos hoja difiere de la de los nodos internos. Los nodos hoja tienen una entrada por cada valor del campo de indexación, junto con un puntero al registro si el campo de búsqueda es un campo clave. En el caso de un campo no clave el puntero apunta a un bloque que contiene punteros a los registros del fichero de datos, creándose un nivel extra de indirección.

1. Implementaciones comerciales usan esta estructura
2. Los punteros a datos sólo en nodos hoja
3. Los punteros de los nodos hoja pueden ser al:
4. Registro, si es campo clave
5. Bloque que contiene el registro con indirección, si no es campo clave
6. Los punteros en nodos internos son punteros de árbol a bloque
7. Los nodos hoja se encadenan para obtener acceso ordenado a registros

Los nodos hoja de un árbol B+ normalmente están enlazados para ofrecer un acceso ordenado a los registros a través del campo búsqueda. Estos son parecidos al primer nivel de un índice. Los nodos internos del árbol B+ corresponden a los demás niveles de un índice multinivel. Los punteros en los nodos internos son punteros de árbol a bloques que son nodos del árbol, mientras que los punteros de los nodos hojas son punteros de datos a registros o bloques del fichero.

**Índices ordenados por varios atributos**

Una ordenación lexicográfica funciona de forma parecida para ordenar las cadenas de caracteres. Un índice en una clave compuesta por n atributos funciona de forma parecida a cualquier índice.

**Dispersión partida**

La dispersión partida es una extensión de la dispersión externa estática que permite el acceso sobre varias claves; solo resulta adecuada para comparaciones de igualdad, las consultas de rango no están soportadas. En la dispersión partida, para una clave compuesta por n componentes, la función de dispersión está diseñada para producir un resultado con n direcciones de dispersión separadas. La dirección de cubo esta concatenación de estas n direcciones. Entonces es posible buscar por la clave de búsqueda compuesta requerida buscando los cubos adecuados que coinciden con las partes de la dirección en la que estamos interesados.

Una ventaja de la dispersión partida es que puede extenderse fácilmente a cualquier número de atributos. Las direcciones de cubo pueden diseñarse para que los bits de orden superior de las direcciones correspondan a los atributos a los que se accede con mas frecuencia.

**Fichero rejillas**

Si queremos acceder a un registro con dos claves, podemos construir una array rejilla con una escala lineal por cada uno de los atributos de búsqueda. Las escalas se hacen en cierto modo para lograr una distribución uniforme sobre ese atributo. Este método es útil para consultas de rango que se mapearían en un conjunto n de celdas correspondientes a un grupo de valores a lo largo de escalas lineales. El array rejilla permite así una partición del fichero a lo largo de las dimensiones de los atributos clave de búsqueda y proporciona un acceso mediante combinaciones de valores a lo largo de esas dimensiones. Los ficheros rejilla funcionan bien para reducir el tiempo de acceso por clave múltiple.

**Seguridad en las Bases de Datos**

**Tipos de Seguridad**

* Aspectos legales y éticos en relación con el derecho de acceso a determinada información. (leyes)
* Temas de políticas a nivel gubernamental, institucional o de empresas en relación con los tipos de información que no debería estar disponible públicamente. (créditos)
* Niveles del sistema, reforzando funciones de seguridad. (hardware, software, DBMS)
* Necesidad dentro de la organización.

**Amenazas a las bases de datos:** las amenazas a las bases de datos tienen como consecuencia la perdida o la degradación de todos o de algunos de los siguientes objetivos de seguridad comúnmente aceptados:

1. **Perdida de integridad**: tiene relación con el requisito a cumplir que la información se encuentre protegida frente a modificaciones inadecuadas.
2. **Perdida de disponibilidad:** tiene relación con que los objetos estén disponibles para un usuario humano o para un programa que tenga los derechos correspondientes.
3. **Perdida de confidencialidad:** tiene relación con la protección de los datos frente al acceso no autorizado.

El DBMS debe proporcionar técnicas que permitan a determinados usuarios o grupos de usuarios el acceso a partes concretas de una base de datos sin tener acceso al resto. Un DBMS incluye un subsistema de autorizaciones y seguridad de base de datos responsable de garantizar la seguridad de partes de una base de datos frente accesos no autorizados.

**Controles para evitar estas amenazas:**

1. **Control de acceso:** restringir el acceso al sistema de base de datos en su totalidad.
2. **Control de inferencias:** restringir el acceso a cierta información detallada.
3. **Control de flujos:** prevenir que la información fluya a usuarios no autorizados.
4. **Control de cifrado:** es una medida para proteger datos confidenciales.

Un DBMS incluye un subsistema de autorizaciones y seguridad en bases de datos responsable de garantizar la seguridad de partes de una base de datos frente a accesos no autorizados.

**DBA**

Es el responsable principal de la gestión de un sistema de base de datos. Entre las responsabilidades del DBA es la concesión y retirada de privilegios a los usuarios que necesitan utilizar el sistema.

1. **Creación de cuentas.**
2. **Concesión de privilegios**
3. **Retirada de privilegios**
4. **Asignación del nivel de seguridad.**

**Protección de acceso, cuentas de usuarios, auditorias de base de datos**

El sistema de base de datos debe seguir el rastro de todas las operaciones en la base de datos que un usuario determinado realice durante cada sesión, que comprende toda la secuencia de interacciones realizadas por este. Es importante seguir el rastro a las operaciones de actualización que se realicen sobre la base de datos, de modo que el DBA pueda determinar que usuario provocó el daño. En caso de una operación malintencionada se realiza una auditoria de base de datos, que consiste en la revisión del archivo de registro para examinar todos los accesos y operaciones realizadas sobre la base de datos durante un determinado periodo de tiempo.

**Mecanismos de Seguridad:**

1. **Discrecionales**: se utilizan para conceder permisos a usuarios, incluyendo la capacidad de acceso a determinados archivos de datos, registros o campos en un modo concreto.

**Control de acceso discrecional:** para reforzarlo, se basa en la concesión y revocación de privilegios.

**Tipos de privilegios discrecionales:**

En SQL2, el concepto de identificador de autorización se utiliza para referirse a una cuenta de usuarios.  
El DBMS debe proporcionar acceso selectivo a cada relación de la base de datos basándose en cuentas determinadas. Existen dos niveles de asignación de privilegios de uso del sistema de base de datos:

1. **El nivel de cuenta**: el DBA especifica los privilegios en particular que posee cada cuenta, independientemente de las relaciones existentes en lavase de datos. Se aplican a las funcionalidades de la propia cuenta y pueden incluir el privilegio para crear un esquema o una relación base.
2. **El nivel de relación**. En este nivel el DBA puede controlar el privilegio de acceso a cada relación o vista individual en la base de datos. Tipos de privilegio:
   * 1. Privilegio de selección
     2. Privilegio de modificación
     3. Privilegio de referencia

La concesión y revocación de privilegios sigue un modelo de autorizaciones para los privilegios discrecionales conocidos como modelo de matriz de acceso, en el que las filas de una matriz M representan los sujetos (usuarios, cuentas, programas) y las columnas representan los objetos (relaciones, registros, columnas, vistas, operaciones). Cada posición M (i,j) de la matriz representa los tipos de permisos (lectura, escritura, actualización) que posee el sujeto i sobre el objeto j.

Para controlar la concesión y revocación de privilegios de una relación, a cada relación R de una base de datos se le asigna una cuenta propietario que es la cuenta que se le utilizo cuando se creo la relación por primera vez. Al propietario se le conceden todos los privilegios.

**Revocación de privilegios:** tiene el propósito de retirar privilegios, algunos casos se determina los privilegios temporales a algunos usuarios.

1. **Obligatorios**: reforzar la seguridad a varios niveles mediante la clasificación de los datos y de los usuarios en varias clases de seguridad, para después implementar la política de seguridad adecuada a la organización. Las clases típicas son:
   * + 1. **Alto secreto (TS)**
       2. **Secreto (S)**
       3. **Confidencial (C)**
       4. **No clasificado (U)**

El modelo utilizado habitualmente para la seguridad a varios niveles, conocido como modelo Bell-LaPadula, clasifica cada sujeto y objeto en una de las clases típicas, llamadas nivel de autorización, surgen estas dos reglas:

1. A un sujeto S no se le permite el acceso de lectura un objeto O a menos que la clase (S) > = clase (O). propiedad de seguridad simple. Ningún sujeto puede leer un objeto cuya clasificación de seguridad sea más alta que el nivel de autorización del sujeto.
2. A un sujeto S no se le permite escribir un objeto O a menos que la clase (S) < = clase (O). esto se conoce como propiedad estrella. Impide que un sujeto escriba un objeto que se encuentre en una clasificación de seguridad menor que el nivel de autorización del sujeto. Por ejemplo, un sujeto con un nivel de autorización TS puede hacer una copia de un objeto de clasificación TS y después guardarlo como un nuevo objeto de clasificación U, haciendo que de este modo pase a ser visible en todo el sistema.

**Control de Acceso basado en roles**

Es una alternativa viable a los controles de acceso discrecionales y obligatorios tradicionales, garantiza que solo los usuarios autorizados tienen acceso a determinados datos o recursos. Los usuarios crean sesiones durante las cuales pueden activar un subconjunto de roles a los que pertenecen. Cada sesión se puede asignar a muchos roles, pero se corresponde con un único usuarios o sujeto.

**DBMS Distribuidos**

Un sistema de computación distribuido consiste en un número de elementos de procesamiento, no necesariamente homogéneos, interconectados mediante una red de computadoras y que cooperan para la realización de ciertas tareas asignadas.  
La viabilidad económica de este planteamiento procede de dos razones: una mayor potencia de computación, y cada elemento de procesamiento autónomo pueda ser administrado de manera independiente.  
Una DDB es una colección de múltiples bases de datos distribuidas interrelacionadas de forma lógica sobre una red de computadoras, y un DDBMS como el software encargado de administrar la base de datos distribuida mientas hace la distribución trasparente para el usuario.

**Ventajas**

1. **Administración de datos distribuidos con distintos niveles de transparencia**. Un DBMS distribuido debe ser una distribución transparente en el sentido de ocultar los detalles de dónde está físicamente ubicado cada fichero dentro del sistema
   1. **Transparencia de Red o de distribución**: hace referencia a la autonomía del usuario de los detalles operacionales de la red.
      1. **de localización**: el comando usado para realizar una tarea es independiente de la ubicación de los datos y del sistema que ejecuto dicho comando.
      2. **de denominación**: implica que una vez transferido un nombre, puede accederse a los objetos nombrados sin ambigüedad.
   2. **Transparencia de replicación:** puede replicarse información en varios lugares para mayor disponibilidad.
   3. **Transparencia de Fragmentación**
   4. **Transparencia de diseño y ejecución:** hace referencia a la libertad de saber como está diseñada la base de datos distribuida y donde ejecuta una transacción.
2. **Incremento de la fiabilidad y la disponibilidad:** la fiabilidad está definida ampliamente como la probabilidad de que un sistema esté funcionando en un momento del tiempo. La disponibilidad es la probabilidad de que el sistema esté continuamente disponible durante un intervalo de tiempo. La distribución aumenta la fiabilidad y disponibilidad del sistema, y se realiza replicación de dato, aún más.
3. **Rendimiento mejorado:** un DDBMS fragmenta la base de datos manteniendo la información lo más cerca posible del punto donde es mas necesaria.
4. **Expansión más sencilla**

**Problemas**

1. Múltiples copias de datos
2. Fallo de sitios
3. Falla de enlace
4. Problemas de comunicación
5. Estancamiento distribuido

**Funciones Adicionales de la BD distribuidas**

1. **Seguimiento de los datos**: capacidad de controlar la distribución de los datos.
2. **Procesamiento de consultas distribuidas**: posibilidad de acceder a sitios remotos y de transmitir consultas y datos a lo largo de todos esos sitios mediante una red de comunicación
3. **Administración de transacciones distribuidas**: estrategias de ejecución de consultas y transacciones
4. **Administración de datos replicados**: capacidad de decidir que copia de datos acceder y de mantener replicado
5. **Recuperación de una BD distribuida**: facultad de recuperarse de las caídas de una localización individual.
6. **Seguridad**
7. **Administración del directorio** (catálogo) distribuido

**Técnicas de Diseño**

**Fragmentación de Datos**

**Fragmentación Horizontal:** de una relación es un subconjunto de las tuplas de esa relación. Las tuplas pertenecientes al fragmento horizontal se especifican mediante una condición sobre uno o más atributos de la relación. Divide a la relación horizontalmente agrupando filas para crear subconjuntos de tuplas cada uno de ellos con un cierto significado lógico. Estos fragmentos pueden asignarse entonces a diferentes sitios del sistema distribuido. La fragmentación horizontal derivada aplica el particionamiento de una relación primaria a otras relaciones secundarias que están relacionadas con aquella a través de una clave externa. Para reconstruir una relación fragmentada horizontalmente se necesita aplicar la operación UNION.

**Fragmentación Vertical:** divide a la relación por columnas. Un fragmento vertical de una relación solo mantiene ciertos atributos de la misma. Esta fragmentación es necesario incluir la clave primaria o alguna otra clave candidata, en cada fragmento vertical de modo que nos permita reconstruir la relación completa a partir de ellos. Para reconstruir una relación fragmentada verticalmente se necesita aplicar la operación OUTER UNION o FULL OUTERN JOIN.

**Fragmentación Mixta:** utiliza los dos tipos de fragmentación.

Un esquema de fragmentación de una base de datos es un conjunto de fragmentos que incluyen todos los atributos y tuplas de esa base de datos y que satisface la condición de que es posible reconstruirla en su totalidad aplicando alguna secuencia de operaciones. Para reconstruir una relación fragmentada en forma mixta se necesita aplicar la operación UNION y OUTER UNION o OUTER JOIN en el orden apropiado.

Un esquema de ubicación describe la colocación de los fragmentos en sitios del DBBS, es un mapa que indica, por cada fragmento, el o los sitio/s en el que está almacenado. Si un fragmento se encuentra en más de un sitio, está replicado.

**Replicación y ubicación de los datos**

Es útil para mejorar la disponibilidad de los datos. En el caso más extremo es la replicación de toda la base de datos en cada sitio del sistema distribuido, siendo esta una base de datos distribuida totalmente replicada. Hacia el lado opuesto a esto, es la no replicación, o sea fragmentación.

Un punto intermedio es la replicación parcial de datos, esto es, fragmentos de la base de datos pueden replicarse mientras que otros no. El número de copias de cada fragmento puede oscilar desde una hasta el número total de sitios del sistema distribuido.

**Diferencias a nivel de Hardware entre BDD y Centralizada**

* Existen múltiples equipos llamados sitios o nodos
* Los sitios deben estar conectados por algún tipo de red

La conexión puede ser a través de:

* Red de área local
* red de área expandida

Se pueden usar diferentes topologías

**Unidad 2**

**Modelo de Datos es un conjunto de herramientas conceptuales .**

Un modelo de datos es básicamente una "descripción" de algo conocido como contenedor de datos (algo en donde se guarda la información), así como de los métodos para almacenar y recuperar información de esos contenedores. Los modelos de datos no son cosas físicas: son abstracciones que permiten la implementación de un sistema eficiente de base de datos; por lo general se refieren a [algoritmos](http://es.wikipedia.org/wiki/Algoritmo), y conceptos matemáticos.

**Modelo Lógicos Basados en Registros (**se utilizan para describir datos en los niveles conceptuales y físicos**)**

Los modelos lógicos basados en registros se utilizan para describir los datos en los modelos conceptual y físico. A diferencia de los modelos lógicos basados en objetos, se usan  para especificar la estructura lógica global de la BD y para proporcionar una descripción a nivel más alto de la implementación.

Los modelos basados en registros se llaman así porque la BD está estructurada en registros de formato fijo de varios tipos. Cada tipo de registro define un número fijo de campos, o atributos, y cada campo normalmente es de longitud fija. La estructura más rica de estas BBDD a menudo lleva a registros de longitud variable en el nivel físico.

Se clasifican:

1. **Jerárquicos**: Éstas son bases de datos que, como su nombre indica, almacenan su información en una estructura jerárquica. En este modelo los datos se organizan en una forma similar a un árbol (visto al revés), en donde un nodo padre de información puede tener varios hijos. El nodo que no tiene padres es llamado raíz, y a los nodos que no tienen hijos se los conoce como hojas. Las bases de datos jerárquicas son especialmente útiles en el caso de aplicaciones que manejan un gran volumen de información y datos muy compartidos permitiendo crear estructuras estables y de gran rendimiento. Una de las principales limitaciones de este modelo es su incapacidad de representar eficientemente la redundancia de datos.
2. **Red**: Éste es un modelo ligeramente distinto del jerárquico; su diferencia fundamental es la modificación del concepto de nodo: se permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico). Fue una gran mejora con respecto al modelo jerárquico, ya que ofrecía una solución eficiente al problema de redundancia de datos; pero, aun así, la dificultad que significa administrar la información en una base de datos de red ha significado que sea un modelo utilizado en su mayoría por programadores más que por usuarios finales.
3. **Relacional:** Éste es el modelo utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postulados sus fundamentos por [Edgar Frank Codd](http://es.wikipedia.org/wiki/Edgar_Frank_Codd), , no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos. Su idea fundamental es el uso de "relaciones". Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados "tuplas". Esto es pensando en cada relación como si fuese una tabla que está compuesta por [registros](http://es.wikipedia.org/wiki/Registro_(base_de_datos)) (las filas de una tabla), que representarían las tuplas, y [campos](http://es.wikipedia.org/wiki/Campo_(base_de_datos)) (las columnas de una tabla). En este modelo, el lugar y la forma en que se almacenen los datos no tienen relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar para un usuario esporádico de la base de datos. La información puede ser recuperada o almacenada mediante "consultas" que ofrecen una amplia flexibilidad y poder para administrar la información.

**Modelo Lógicos Basados en Objetos: se utilizan para describir datos en los niveles conceptuales y externos.**

Los modelos lógicos basados en objetos se usan para describir datos en el nivel conceptual y el externo. Se caracterizan porque proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos. Los modelos más conocidos son el modelo entidad-relación y el orientado a objetos.

**Entidad Relación:** se basan en la percepción del mundo real que consiste en una colección de objetos llamados entidades y sus relaciones, una entidad es un objeto distinguible de otro por medio de un conjunto específico de atributos. Atributos clave: minimalidad y unicidad

**Orientado a Objeto:** Este modelo, bastante reciente, y propio de los [modelos informáticos orientados a objetos](http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos), trata de almacenar en la base de datos los objetos completos (estado y comportamiento).

Una base de datos orientada a objetos es una base de datos que incorpora todos los conceptos importantes del paradigma de objetos:

* [**Encapsulación**](http://es.wikipedia.org/wiki/Encapsulaci%C3%B3n_(programaci%C3%B3n_orientada_a_objetos)) - Propiedad que permite ocultar la información al resto de los objetos, impidiendo así accesos incorrectos o conflictos.
* [**Herencia**](http://es.wikipedia.org/wiki/Herencia_(programaci%C3%B3n_orientada_a_objetos)) - Propiedad a través de la cual los objetos heredan comportamiento dentro de una jerarquía de clases.
* [**Polimorfismo**](http://es.wikipedia.org/wiki/Polimorfismo_(inform%C3%A1tica)) - Propiedad de una operación mediante la cual puede ser aplicada a distintos tipos de objetos.

En bases de datos orientadas a objetos, los usuarios pueden definir operaciones sobre los datos como parte de la definición de la base de datos. Una operación (llamada función) se especifica en dos partes. La interfaz (o signatura) de una operación incluye el nombre de la operación y los tipos de datos de sus argumentos (o parámetros). La implementación (o método) de la operación se especifica separadamente y puede modificarse sin afectar la interfaz. Los programas de aplicación de los usuarios pueden operar sobre los datos invocando a dichas operaciones a través de sus nombres y argumentos, sea cual sea la forma en la que se han implementado. Esto podría denominarse independencia entre programas y operaciones.

**Modelo Relacional**

**Características o Propiedades de las relaciones:**

1. Las tuplas no están ordenadas -conjunto matemático-

2. Los atributos en las tuplas no están ordenados -conjunto matemático-

3. Los valores en los atributos deben ser atómicos –relaciones normalizadas-

4. No hay tuplas repetidas –hechos diferentes del mundo real-

Este modelo utiliza el concepto de una relación matemática, como su bloque de construcción básico, y tiene su base teórica en la teoría de conjuntos y la lógica del predicado de primer orden.

Representa la base de datos como una colección de relaciones, cada una de estas relaciones se parece a una tabla de valores o, de forma algo más extensa, aun fichero plano de registros.

Cuando una relación está pensada como una tabla de calores, cada fila representa una colección de valores relacionados. Cada fila de la tabla representa un hecho que se corresponde con una relación o entidad real. El nombre de la tabla y de las columnas se utiliza para ayudar a interpretar el significado de cada uno de los valores de las filas. Una fila recibe el nombre de tupla, una columna es un atributo y el nombre de la tabla una relación.

**Dominio**: un dominio D es un conjunto de valores atómicos. Por atómico se entiende que cada valor de un dominio es indivisible en lo que al modelo relacional se refiere.

Un dominio cuenta con un nombre, tipo de dato y un formato. También puede facilitarse información adicional para interpretación de sus valores.

Un esquema relación o diseño de relación R (A1, A2, …, An)está constituido por el nombre de la relación R y una lista de atributos A1, A2, …, An. Cada atributo Ai es el nombre de un papel jugado por algún dominio D en el esquema de relación R. se dice que D es el dominio de los atributos Ai, Dom (Ai). un esquema de relación se utiliza para describir una relación, se dice que R es el nombre de la misma. El grado de una relación es el número de atributos de la misma.

Una relación r del esquema R (A1, A2, …, An), r(Ai), es un conjunto de n-tuplas r = { t1, t2, …, tn }. Cada tupla es una lista ordenada de n valores, t = < v1, v2, …, vn >. Es decir, una relación r(R) es una relación matemática de grado n en los dominios de dom(An) que es un subconjunto del producto cartesiano que definen a R.

El producto cartesiano especifica todas las posibles combinaciones de valores de los dominios subyacentes.

El número de atributos en la relación se llama grado de la relación. El número de tuplas en la relación se llama cardinalidad de la relación. La cardinalidad de una relación cambia con el tiempo, el grado no.

**Restricciones de Dominio**

**Restricciones de clave y restricciones de Valores Null**

Una relación está definida como un conjunto de tuplas. Todos los elementos de un conjunto son distintos, todas las tuplas en una relación también deben serlo. Esto significa que dos tuplas no pueden tener la misma combinación de valores para todos sus atributos, unicidad.

Cualquier conjunto de atributos SK recibe el nombre de superclave del esquema de relación R. Una superclave específica una restricción de exclusividad por la que dos tuplas distintas en cualquier estado r de R puede tener el mismo valor para SK. Una clave no tiene redundancia, una clave K de un esquema de relación R es una superclave de R con la propiedad adicional que eliminando cualquier atributo A de K deja un conjunto de atributos K’ que ya no es una superclave de R. por lo tanto una clave satisface dos restricciones:

1. Dos tuplas diferentes en cualquier estado de la relación no pueden tener valores idénticos para todos los atributos de la clave. **unicidad**
2. Es una superclave mínima, de la cual no podemos eliminar ningún atributo y seguiremos teniendo almacenada la restricción de exclusividad de la condición 1. **Minimalidad**.

**Integridad de entidad, integridad referencial y foreign keys**

Las restricciones de integridad de entidad declaran que el valor de ninguna clave principal puede ser Null. Esto se debe a que dicha clave de emplea para identificar tuplas individuales en una relación. Si se permitiera este valor, significaría que no se podrían identificar ciertas tuplas.

Las restricciones de integridad referencial están especificadas entre dos relaciones y se utilizan para mantener la consistencia entre las tuplas de dos relaciones. Las restricciones de integridad referencial suelen originarse a partir de las relaciones entre las entidades representadas por los esquemas de relación.

Foreign keys, un valor de clave ajena o foránea representa una referencia a la tupla donde se encuentra el valor correspondiente a la clave primaria. La regla de integridad referencia determina que una base de datos no debe contener valores de referencia sin concordancia.

Reglas de actualización (compensación) para DELETE/UPDATE

Casos: Null, Restringir, Por Defecto, Cascada.

**Restricciones (constraints) en valores:**

1. Implícitas o basadas en el modelo –características o propiedades de las relaciones
2. Explícitas o basadas en el esquema –DDL-a. Valores atómicos. Condiciones de Clave:
   * 1. Unicidad: no puede haber 2 tuplas con el mismo valor en todos los atributos
     2. Minimalidad: debe ser el menor conjunto de atributos (superclave-clave)
3. Reglas de negocio –triggers, aserciones, aplicaciones
4. Explícitas o basadas en el esquema –DDL-

**Algebra Relacional**

El conjunto de operaciones básicas del modelo relacional es el álgebra relacional, el cual permite al usuario especificar las peticiones fundamentales de recuperación. El resultado de una recuperación es una nueva relación, la cual puede estar constituida por una o más relaciones. Proporciona un fundamento formal para las operaciones del modelo relacional. Se utiliza como base para la implementación y optimización de consultas en los RDBMS. Los cálculos relacionales ofrecen una notación declarativa de alto nivel para especificar las consultas relacionales. Una expresión de cálculo relacional crea una nueva relación, la cual esta especificada en términos de variables que engloban filas de las relaciones almacenadas en la base de datos o columnas de las relaciones almacenadas.

**Operaciones Unarias**

**Selección**

Se emplea para seleccionar un subconjunto de tuplas de una relación que satisface una condición de selección. La selección es unaria, se aplica a una sola relación. Además no pueden implicar a más de una tupla. El grado de la relación resultante es el mismo que el de R.

**σDNI=4(EMPLEADO)**

**Proyección**

Selecciona ciertas columnas de la tabla y descarta las otras. El resultado de esta operación puede visualizarse como una partición vertical de la relación en otras dos. Una que contiene las columnas necesarias, y otra las descartadas. El número de tuplas de una relación resultante de una operación de proyección es siempre menor o igual que el de las contenidas en R.

**Πsexo,sueldo(EMPLEADO)**

**Renombrar**

Es un operador unario. Renombra tanto la relación, atributos o ambos.

TEMP\_EMP 🡨 **Πsexo,sueldo(EMPLEADO)**

**Operaciones de Conjunto**

Cuando se refieren a las bases de datos relacionales, las relaciones sobre las que se aplican estas tres operaciones deben tener el mismo tipo de tuplas. Esta condición se denomina compatibilidad de unión.

1. **Unión**: el resultado de esta operación es una relación que incluye a todas las tuplas que están tanto en R como en S, o en ambas.
2. **Intersección**: el resultado de esta operación es una relación que incluye todas las tuplas que están en R y en S
3. **Diferencia de conjuntos**: el resultado de esta operación, es una relación que incluye todas las tuplas que están en R pero no en S.

**Producto Cartesiano**

Se trata de una operación binaria. Produce un nuevo elemento combinando cada tupla de una relación con los de la otra. Es útil cuando va seguida por una selección que correlacione los valores de los atributos procedentes de las relaciones componentes.

**Operaciones Binarias**

**Concatenación**

Se emplea para combinar tuplas relacionadas de dos relaciones en una sola. Esta operación es muy importante para cualquier base de datas relacional que cuente con más de una relación, ya que nos permite procesar relaciones entre relaciones.

* Concatenación EQUIJOIN: es aquella que supone el uso de conexión solo por comparaciones de igualdad.
* Concatenación NATURAL: elimina los atributos superfluos de una comparación. Equipara todos los pares de atributos que tengan el mismo nombre en las dos relaciones.
* Concatenación Externa: es una operación que puede usarse cuando se desea mantener en el resultado todas las tuplas independientemente si tienen correspondecias o no con la otra realcion.
  1. Externa Izquierda, se mantiene todas las tuplas de la primera relación.
  2. Externa Derecha, se mantienen todas las tuplas de la segunda relación.
  3. Externa Completa, se mantienen todas las tuplas de las dos relaciones.

**División**

**R(Z) / S(X) donde X está incluido en Z. permite Y = Z – X, por lo tanto Z = X U Y**

La división es útil para cierto tipo de consultas que a veces se realizan aplicaciones de base de datos. El resultado de una división es una relación T(Y) que incluye una tupla t si las tuplas tr aparecen en R con tr[Y] = t, y con tr[X] = ts para cada tupla ts en S. esto es para que una tupla t aparezca en el resultado T de la división, los valores de aquella deben aparecer en R en combinación con cada tupla en S.

**Unidad 3**

**Introducción**

El procedimiento de normalización consiste en la aplicación de una serie de comprobaciones de las relaciones para cumplir con unos requisitos cada vez más restrictivos y descomponer las relaciones cuando sea necesario.

**Dependencia Funcional.**

Una dependencia funcional es una restricción que se establece entre dos conjuntos de atributos de la base de datos.

Una dependencia funcional, denotada por X🡪Y, entre dos conjuntos de atributos X e Y que son subconjuntos de R, especifica una restricción en las posibles tuplas que pueden formar un estado de la relación r en R. La restricción dice que dos tuplas t1 y t2 en r cumplen que t1[X] = t2[X], deben cumplir que t1[Y] = t2[Y].

Esto es, que los valores del componente Y de una tupla de r dependen de, o están determinados por, los valores del componente X; alternativamente, los valores del componente X de una tupla únicamente, funcionalmente, determinan los valores del componente Y. Decimos que también que existe una dependencia funcional de X hacia Y, o que Y es funcionalmente dependiente de X. Por lo tanto, X determina el funcionamiento de Y si para toda instancia de r del esquema de relación R, no es posible que r tenga dos tuplas que coincidan en los atributos de X y no lo hagan en los atributos de Y.

* Si una restricción de R indica que no puede haber más de una tupla con un valor X concreto en cualquier instancia de relación r(R), es decir, que X es una clave candidata de R, se cumple que X🡪Y para cualquier subconjunto de atributos de Y de R.
* Si X🡪Y en R, esto no supone que Y🡪X en R.

Es una propiedad de la semántica o del significado de los atributos.

El conjunto de todas las dependencias que influyen F, junto con las dependencias que pueden inferirse de F, reciben el nombre de clausuras de F.

**Formas normales basadas en claves principales**

**Normalización de las Relaciones**

El proceso de normalización hace pasar un esquema de relación por una serie de comprobaciones para certificar que satisface una determinada forma normal.

La normalización de los datos puede considerarse como un proceso de análisis de un esquema de relación, basado en sus DF y sus claves principales, para obtener las propiedades deseables de minimizar la redundancia y minimizar las anomalías de inserción, borrado y actualización.

**Claves:** si un esquema de relación tiene más de una clave, cada una de ellas se denomina clave candidata. Una de ellas se elige arbitrariamente como calve principal, mientras que el resto son claves secundarias.

**Propiedades de las Claves Candidatas:**

1. Unicidad: En un momento dado, no existen dos tuplas en la relación con el mismo valor de clave candidata.
2. Minimalidad: Si la clave candidata es compuesta, no es posible destruir ningún elemento de esta clave, sin destruir su unicidad.

Un atributo del esquema de relación de R recibe el nombre de atributo primo de R si es miembro de algunas de las claves candidatas de R. un atributo es no primo si no es miembro de ninguna clave candidata.

**Primera Forma Normal**

La primera forma normal prohíbe tener un conjunto de valores, una tupla de valores o una combinación de ambos como valor de un atributo para una tupla individual. Prohíbe las relaciones dentro de las relaciones o las relaciones como valores de atributo dentro de las tuplas. Los únicos valores de atributo permitidos por la 1ra forma normal son los atómicos.

**Segunda Forma Normal**

Está basada en el concepto de dependencia funcional total. Una dependencia funcional es total si X🡪Y es total si la eliminación de cualquier atributo A de X implica que la dependencia funcional deje de ser válida.

Un esquema de relación de R está en segunda funcional si todo atributo no primo de A en R es completa y funcionalmente dependiente de la clave principal R.

**Tercera Forma Normal**

Se basa en el concepto de dependencia transitiva. Una dependencia funcional X🡪Y en un esquema de relación R es una dependencia transitiva si existe un conjunto de atributos Z que ni es clave candidata ni un subconjunto de ninguna clave de R, y se cumple tanto X🡪Z como Z🡪Y.

Un esquema está en 3ra forma normal si satisface la 2da forma normal y ningún atributo no primo de R es transitivamente dependiente en la clave principal.

**Forma Normal Boyce-Codd**

Forma Normal de Boyce-Codd (o FNBC) : Es una versión ligeramente más fuerte de la (3FN). Requiere que no existan dependencias funcionales no triviales de los atributos que no sean un conjunto de la clave candidata.

Todos los determinantes de la tabla son clave candidata.

Definición Formal: Una tabla está en FNBC si y solo si está en 3FN y cada dependencia funcional no trivial tiene una clave candidata como [determinante](http://es.wikipedia.org/w/index.php?title=Determinante_Funcional&action=edit&redlink=1).

**Cuarta Forma Normal**

La cuarta forma normal (4NF) La definición de la 4NF se basa en el concepto de dependencia multivalor. La dependencia multivalor es una donde la existencia de dos o más relaciones independientes muchos a muchos causa redundancia; y esa redundancia es la que es removida por la cuarta forma normal.

Definición:    Una relación R está en 4FN con respecto a un conjunto D de  dependencias funcionales y de valores múltiples sí, para todas las dependencias de valores múltiples en D de la forma X->->Y, donde X<=R y Y<=R, se cumple por lo menos una de estas condiciones:

\* X->->Y es una dependencia de valores múltiples trivial.  
\* X es una superllave de la relación R.

**Quinta Forma Normal**

Quinta forma normal (5NF): Es un nivel de normalización de bases de datos designado para reducir la redundancia en las bases de datos relacionales que guardan hechos multivalores aislando relaciones múltiples relacionadas.

Definición: Una tabla se dice que está en 5NF sí y sólo si está en 4NF y cada dependencia de unión (join) en ella es implicada por las claves candidatas.

Características: La tabla puede dividirse en subtablas que no pueden reconstruirse.

**Unidad 4**

**Introducción SQL**

SQL es una herramienta para organizar, gestionar y recuperar datos almacenados en una base de datos informática. Es un lenguaje informático que se utiliza para interactuar con una base de datos. Funciona con un tipo específico de base de datos, llamado base de datos relacional.  
SQL se utiliza para controlar todas las funciones que suministra un DBMS a sus usuarios:

1. **Definición de datos**, permite que un usuario defina la estructura y la organización de los datos almacenados, así como las relaciones existentes entre ellos.
2. **Recuperación de datos**, permite a un usuario o a un programa recuperar y utilizar los datos almacenados en una base de datos.
3. **Manipulación de datos**, permite a un usuario o a un programa actualizar la base de datos añadiendo datos nuevos, borrando los viejos y modificando los almacenados.
4. **Control de acceso**, puede ser utilizado para restringir la capacidad de un usuario para recuperar, añadir y modificar datos, protegiendo los datos almacenados contra accesos no autorizados.
5. **Compartición de información**, coordina la compartición de datos entre usuarios recurrentes.
6. **Integridad de datos**, define restricciones de integridad en la base de datos, protegiéndola de alteraciones debidas a actualizaciones inconsistentes o fallos del sistema.

**SQL es**

1. **Es un lenguaje interactivo de consultas**, los usuarios escriben órdenes SQL e un programa SQL interactivo para recuperar datos y presentarlos en pantalla
2. **Es un lenguaje de programación de bases de datos**, los programadores introducen ordenes SQL en sus programas para acceder a los datos de las base de datos.
3. **Es su lenguaje de administración de base de datos**, el administrador responsable de la gestión de la base de datos utiliza SQL para definir la estructura de la base de datos y el control de acceso a datos almacenados.
4. **Es un leguaje cliente/servidor**, los programas de las computadores personales utilizan SQL para comunicarse con los servidores débase de datos.
5. **Es un lenguaje de bases de datos distribuidas**, los sistemas de gestión de bases de datos distribuidas utilizan SQL para ayudar a distribuir los datos a través de muchos sistemas informáticos conectados.
6. **Es** **un lenguaje pasarela de base de datos**, en una red informática de donde se mezclan diferentes productos DBMS, SQL se usa a menudo como una pasarela que permite un determinado producto DBMS se comunique con otros diferentes.

**Características y Ventajas**

1. **Independencia de los proveedores** SQL se ofrece en todos los principales proveedores de DBMS
2. **Portabilidad entre sistemas informáticos**
3. **Estándares SQL, el ANSI y el ISO, publicaron el estándar en 1986.** El ANSI y el ISO publicaron un estándar oficial de SQL en 1986. SQL es también un estándar federal americano de procesamientos de información.
4. **Aprobación de IBM**, fue originado por investigadores de IBM, y desde entonces se ha convertido en un producto estratégico de IBM basado en su DB2
5. **ODBC y Microsoft**, el estándar patrocinado por Microsoft es el acceso a las base de datos abiertas, una facilidad basada en SQL.
6. **Base Relacional,** es un lenguaje de bases de datos relacionales.
7. **Estructura de alto nivel parecida al inglés.**
8. **Consultas interactivas ad hoc,** proporciona a los usuarios ad hoc a los datos almacenados. Los datos son más accesibles y pueden ser utilizados para tomar mejores decisiones y más informadas.
9. **Acceso a la base de datos mediante programas**
10. **Vistas múltiples de datos**
11. **Lenguaje completo de la base de datos,** SQL proporciona un lenguaje completo y consistente para crear una base de datos.
12. **Definición dinámica de datos,** usando SQL una base de datos puede ser modificada y/o ampliada dinámicamente.
13. **Arquitectura cliente/servidor.**

**Arquitecturas**

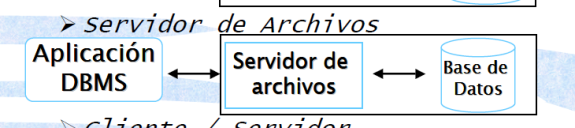
**Arquitectura Centralizada**

Se basa en un computador central, en el cual está alojado la base de datos y el programa de aplicación. Tanto el DBMS como los datos físicos residen un mismo lugar, junto con el programa de aplicaciones que acepta entradas desde el terminal de usuario. El DBMS accede a la base de datos para extraer los registros almacenados recuperando los datos necesarios, y el programa de aplicación muestra el resultado en pantalla. El sistema es multiusuario por lo que cuando acceden varios usuarios, cae el rendimiento



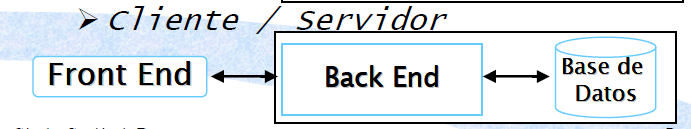
**Arquitectura Servidora de Archivos**

Se basa en un computador que almacena todos los archivos compartidos, y cada usuario tiene una PC independiente con una copia del DBMS, cada uno realiza una consulta, el DBMS pide al software de red los bloques de datos que necesita. Esta arquitectura proporciona un rendimiento excelente para consultas típicas, ya que cada usuario dispone de la potencia completa de una computadora personal ejecutando su propia copia del DBMS



**Arquitectura Cliente/Servidor**

Cada usuario tiene una PC con el software de aplicación unido a una LAN. Cuando se realiza una consulta la petición SQL viaja hasta el servidor DBMS. Las funciones del DBMS están divididas en dos partes. Los front-ends de la base de datos, tales como herramientas de consulta interactiva, generadores de informe y programas de aplicación, se ejecutan en la PC. El motor de soporte, back-end de la base de datos que almacena y gestiona los datos se ejecuta en el servidor.



**El mito de la portabilidad:**

* **Interfaz de programa**, el estándar original especifica una técnica abstracta para utilizar SQL desde dentro de un programa de aplicaciones escrito en un lenguaje de programación. SQL2 especifica una interfaz SQL embebida para leguajes de programación populares.
* **SQL Dinámico**, SQL1 no incluye características requeridas para desarrollar frontales de base de datos de propositico general, tales como herramientas de consulta y generadores de informes. SQL2 lo incluye.
* **Estructura de base de datos**, SQL1 especifica el lenguaje SQL utilizado cuando una base de datos particular ha sido abierto y esta lista para procesar. Los detalles de nombrado de la base de datos y de cómo se establece la conexión inicial varían enormemente y no son portables. El estándar SQL2 crea más uniformidad, pero no puede enmascarar completamente estos detalles.
* **Códigos de Error**, el SQL1 no específica códigos de error a devolver cuando SQL detecta un error, SQL2 proporciona códigos de error estándar.
* **Tipos de Datos**, SQL1 define un conjunto mínimo de tipos de datos, pero omite algunas de los tipos mas populares. El estándar SQL2 afronta estos.
* **Diferencias semánticas**, debido a que el estándar especifica ciertos detalles como definidos por el fabricante, es posible ejecutar la misma consulta con dos implementaciones SQL diferentes y producir dos conjuntos diferentes de resultados.
* **Secuencias de cotejo**, SQL1 no define la secuencia de cotejo (ordenación) de los caracteres almacenados en la base de datos. SQL2 incluye.
* **Tablas de Sistema**, el estándar SQL1 no dice nada acerca de las tablas de sistema que proporcionan información referente a la estructura de la propia base de datos. El SQL2 están estandarizadas.
* **SQL Interactivo**, el estándar solamente especifica el SQL de programa utilizado por un programa de aplicación y no el SQL interactivo.

**Conceptos Básicos**

1. **Sentencias**: El lenguaje consta de unas treinta sentencias, cada una demanda una acción específica por parte del DBMS. Toda sentencia comienza con un verbo, describe lo que la sentencia hace. La sentencia continúa con una o más clausulas, cada una puede especificar los datos sobre los que debe actuar la sentencia, o proporcionar más detalles acerca de lo que la sentencia se supone que hace, siendo cada una opcional o necesaria. **Verbo, cláusulas, mayúsc./minúsc.**
2. **Nombres**: los objetos se identifican asignándoles u nombre. Estos nombres son utilizados en las sentencias para identificar el objeto de la base de datos sobre la que la sentencia debe actuar. Los nombres deben contener de 1 a 18 caracteres. Deben ser breves y cualificados.

* Nombre de las tablas
* Nombre de Columnas
* Nombre de usuarios
* Nombre de Procedimientos Almacenados, y Referencias

1. **Expresiones:** Las expresiones se utilizan para calcular valores que se recuperan de una base de datos y para calcular valores utilizados en las búsquedas en la base de datos (), +, -, \*,/
2. **Funciones Internas:** Estas funciones proporcionan con frecuencia facilidades de conversión de tipo de datos, o para volver a darle formato a los datos.

* Numéricas
* Carácter (valores numéricos o carácter)
* Fecha
* Conversión
* Funciones de grupo

1. **Valor Null:** Es un dato faltante, desconocido o no ser aplicable. Es un indicador que dice a SQL y al usuario que el dato falta o no es aplicable.
2. **Tipos de datos estándar**

* Tipos de datos estándar: carácter longitud Fija, enteros, decimales, coma flotante
* Tipos de Datos Extendidos: cadena de caracteres de longitud variable, moneda, fecha y hora, booleanos, flujo de bytes no convencionales.

1. **Constantes**

* Constantes numéricas 🡪 3E-2,0.5,-9
* Constantes de cadena 🡪 ’JUAN’
* Constantes de fecha y hora 🡪 18-JAN-91
* Constantes simbólicas 🡪 GETDATE(),CURRENT\_DATE,SYSDATE

**Consultas Simples**

**Sentencia SELECT:** Esta sentencia recupera datos de una base de datos y los devuelve en forma de resultados de consulta.

**SELECT {lista de columnas}**

**FROM tabla**

*WHERE condiciones*

*GROUP BY {lista de columnas}*

*HAVING condiciones*

*ORDER BY {lista de columnas}*

1. **SELECT** lista los datos a recuperar, siendo estos columnas, o columnas a calcular, puede ir un nombre de una columna, una constante, una expresión SQL.
2. **FROM** lista las tablas que contienen los datos a recuperar por la consulta
3. **WHERE** condiciona que incluya ciertas filas de datos en los resultados de las consultas.
4. **GROUP BY** especifica una consulta resumen, agrupa todas las filas de datos de la base de datos, una consulta resumen agrupa todas las filas similares y luego produce una fila resumen de resultados para cada grupo.
5. **HAVING** condiciona a que incluya ciertos grupos producidos por la clausula GROUP BY en los resultados de la consulta, al igual que WHERE condiciona la búsqueda para ciertos grupos.
6. **ORDER BY** ordena los resultados de la consulta en la base de datos de una o más columnas.

**Resultados de Consultas**

El resultado de una consulta es siempre una tabla de datos, semejantes a las tablas de la base de datos. Los resultados tienen siempre el mismo formato tabular de fila/columna que las tablas efectivas de la base de datos. Los resultados formaran una tabla con varias columnas y varias filas.

SELECT apellido,nombres,nrodoc

FROM empleados

* **Consultas Sencillas**, son las que solicitan columnas de datos de una única tabla en la base de datos
* **Columnas calculadas**, son aquellas cuyos valores se calculan a partir de los valores de los datos almacenados.

SELECT descrip, (prevent - precosto)

FROM artículos

**Selección de todas las columnas**

SELECT \*

FROM empleados

**Filas Duplicadas,** si no se incluye la clave primaria en la clausula SELECT, puede producir filas duplicadas. SQL genera un conjunto completo del resultado y elimina luego las filas duplicadas exactos de alguna otra formar los resultados finales.

SELECT DISTINCT apellido, nombres

FROM empleados

**Selección de Filas con WHERE**

Esta clausula se utiliza para especificar las filas que se desean recuperar. Si la condición de búsqueda es TRUE, la fila se incluye en los resultados de la consulta. Si el valor de la condición de búsqueda tiene un valor NULL la fila se excluye del resultado.

SELECT apellido, nombres, dirección

FROM empleados

WHERE nrodoc = 20780963

**Condiciones de Búsqueda**

1. **Test de comparación**, compara el valor de una expresión con el valor de otra. (<=, >=, =, <, >,<> ) calcula y compara los valores de dos expresiones por cada fila de datos. Si el campo es NULL, el test da FALSE.
2. **Test de rango**, examina si el valor de una expresión cae dentro de un rango especificado de valores. BETWEEN, SQL proporciona una forma diferente de condición de búsqueda, el test de rango comprueba si un valor de dato se encuentra entre dos valores especificados. Implica el uso de tres expresiones SQL, la 1ra define el valor a comprobar, las expresiones segunda y tercera definen los extremos superiores e inferiores del rango a comprobar.  
   Si la expresión del test produce un valor NULL, o si ambas expresiones definitorias del rango producen valores NULL, el test devuelve NULL.  
   Si la expresión que define el extremo inferior del rango produce un valor NULL, el test devuelve false si el valor de test es superior al límite superior.  
   Si la expresión que define el extremo superior del rango produce un valor NULL, el test devuelve FALSE si el valor de test es menor que el límite inferior y NULL en caso contrario.
3. **Test de pertenencia a conjunto**, comprueba si el valor de una expresión se corresponde con un de un conjunto de valores. Si la expresión del test devuelve un valor NULL el test IN devuelve NULL. X IN (A,B,C) es equivalente a X = A or X = B or X = C
4. **Test de correspondencia con patrón LIKE**¸ comprueba si el valor de una columna que contiene datos de cadena de caracteres se corresponde a un patrón especificado.   
   Caracteres comodines, signo de porcentaje %, se corresponde con cualquier secuencia de cero a mas caracteres. El comodín subrayado \_ se corresponde con cualquier carácter simple.  
   Carácter escape \*, el estándar especifica como una manera de comparar literalmente caracteres que son comodines, para ello utiliza el signo $, agregando al final del patrón ESCAPE ‘ $’, determinando que el signo comodín precedido del $ sea tomado en forma literal.  
   El test LIKE debe aplicarse a una columna con un tipo de datos cadena. Si el valor del dato en la columna es NULL, el test LIKE devuelve NULL
5. **Test de valor nulo**, comprueba si una columna tiene el valor NULL, el resultado de una búsqueda puede ser TRUE o FALSE.

**Condiciones de Búsqueda AND, OR, NOT:** Se utilizan estas condiciones para formar búsquedas más complejas.

**Ordenación de los Resultados de una Clausula ORDER BY**

Las filas de los resultados de una consulta no están dispuestas en ningún orden particular. Esta clausula consta de las palabras claves ORDER BY, seguidas de una lista de especificaciones de ordenación separadas por comas.

**Regla de procesamiento de consultas de tabla única.**

Los resultados producidos por una sentencia SELECT, se especifica aplicando cada una de sus clausulas, una por una. La clausula FROM se aplica en primer lugar, luego se aplica la clausula WHERE, posteriormente se aplica la clausula SELECT, finalmente se aplica la clausula ORDER BY para ordenar los resultados.

1. Comenzar con la tabla designada en FROM
2. Si hay WHERE aplicar la condición
3. Armar la lista de selección
4. Si hay SELECT DISTINCT, eliminar duplicados
5. Si hay cláusula ORDER BY, aplicarla

**Combinación de Resultados**

La operación UNION produce una única tabla de resultados que combina las filas de la primera consulta con las filas de los resultados de la segunda consulta. En ambas consultas no se puede usar ORDER BY, sino que puede ser utilizado después de la segunda sentencia SELECT.

Restricciones:

1. Amabas tablas deben contener el mismo número de columnas
2. El tipo de dato de cada columna en la primera tabla debe ser el mismo que el tipo de datos de la columna correspondiente en la segunda tabla
3. Ninguna de las dos tablas puede estar ordenadas con la clausula ORDER BY. Sin embargo, los resultados combinados pueden ser ordenados, según se describe en la sección siguiente.
4. El estándar solamente permite nombres de columna o una especificación de todas las columnas en la lista de selección, y prohíbe las expresiones en la lista de selección.

**Unión y Filas Duplicadas**

Por omisión la unión elimina las filas duplicadas como parte de su procesamiento. Si se desea duplicar las filas, se debe incorporar la palabra ALL. Si se sabe que la combinación de las consultas no produce filas duplicadas, debe agregarse el ALL, ya que aumenta el tiempo de la consulta al no ordenarle al motor que revise la duplicación de las filas.

**Composiciones Simples**

El proceso de formar parejas de filas haciendo coincidir los contenidos de las columnas relacionadas se denomina componer JOINING de las tablas, la tabla resultante se denomina composición entre las dos tablas.

**Consulta Padre/Hijo:** Las consultas más comunes implican a dos tablas que tienen una relación natural padre/hijo.

Las claves ajenas y las claves primarias crean relaciones padre/hijo en una base de datos. La tabla que contiene la clave ajena es el hijo en la relación, la tabla con la clave primaria es el padre. Para realizar la consulta padre/hijo, debe especificarse la comparación de las claves en la condición de búsqueda.

* Las composiciones que asocian claves primarias a claves ajenas siempre crean relaciones padre/hijo de uno a muchos
* Otras composiciones pueden generar relaciones uno a muchos si la columna de emparejamiento en al menos una de las tablas tiene valores únicos para todas las filas de la tabla.
* En general, las composiciones sobre las columnas de emparejamiento arbitrarias generan relaciones de muchos a muchos.

**Consideraciones SQL para consultas multitablas:**

Los nombres de columnas cualificados son necesarios a veces en consultas multitabla para eliminar referencias de columna ambiguas.

Las selecciones de todas las columnas tiene un significado especial para las consultas multitablas, ej. Ventas.\*

Las autocomposiciones pueden ser utilizadas para crear una consulta multitabla que relaciona una tabla consigo misma.

SELECT e.apelemp,e.nomemp,jefe.apelemp

FROM empleados e, empleados jefe

WHERE e.nrojefe \* = jefe.nroemp

SELECT e.apelemp,e.nomemp,jefe.apelemp

FROM empleados e LEFT JOIN empleados jefe ON e.nrojefe = jefe.nroemp

Los alias de tablas pueden ser utilizados en la clausula FROM para simplificar nombre de columna cualificados y permitir referencias de columna no ambiguas en autocomposiciones.

**La clausula FROM tiene dos funciones importantes:**

1. Identifica todas las tablas que contribuyen con datos a los resultados de la consulta. Las columnas referenciadas en la sentencia SELECT deben provenir de una de las tablas designadas en la clausula FROM.
2. Determinar la marca que se utiliza para identificar la tabla en la referencia de columna cualificada dentro de la sentencia SELECT. Si se especifica un alias, este pasa a ser la marca de tabla; en caso contrario se utiliza como marca el nombre de la tabla, tal como aparece en la clausula FROM.

**Consideraciones:**

* Cualificar atributos utilizados y que figuran en ambas tablas (por lo menos)
* Emparejar clave primaria con clave ajena en el WHERE
* Posibilidad de simplificación, utilizando alias de tablas
* Efecto del \*, sólo y cualificado.
* No aparecen las filas sin emparejar
* El efecto de NULL en clave foránea

**Regla de procesamiento: Consultas multitabla**

1. Formar producto cartesiano.
2. Aplicar cláusula WHERE, si está.
3. Extraer columnas seleccionadas.
4. Eliminar filas duplicadas si está la cláusula DISTINCT.
5. Ordenar si está especificado ORDER BY.

**Consultas Sumarias**

SQL permite resumir datos de la base de datos mediante un conjunto de funciones de columna. Una función de columna SQL acepta una columna entera de datos como argumento y produce un único dato que resume la columna.

1. **SUM**: calcula total de la columna  
   calcula la suma de una columna de valores de datos. Deben tener un tipo numérico.

Suma y como argumento valor numérico y resultado del mismo tipo de dato**. Ignora valores NULL.**

1. **AVG**: calcula promedio  
   Promedio y argumento valor numérico y el resultado de otro tipo por la división. **Ignora valores NULL.**
2. **COUNT**: cuenta cantidad de valores  
   COUNT (n): Cantidad de valores en columna. Argumento de cualquier tipo. Resultado un entero. **Ignora valores NULL.**

COUNT(\*): Cantidad de filas, incluyendo duplicadas y nulas

1. **MAX**: obtiene el máximo valor  
   MAX(n): Valor máximo de n. Argumento de cualquier tipo de dato y resultado del mismo tipo. **Ignora valores NULL**.
2. **MIN**: obtiene el mínimo valor  
   MIN(n): Valor mínimo de n. Argumento cualquier tipo de dato y resultado del mismo **tipo. Ignora valores NULL.**Cadena de Caracteres, depende del conjunto de caracteres que está siendo utilizado.
3. COUNT (DISTINCT n) :Cantidad de filas con valores no repetidos.

**TODA LA COLUMNA CON NULL:**  
 SUM, AVG, MIN y MAX devuelven **NULL**.  
 **COUNT devuelve 0**.  
SI NO HAY DATOS EN LA COLUMNA, ESDECIR, LA COLMNA ESTA VACIA, LAS FUNCIONES SUM, AVG, MIN, MAX DEVUELVEN CERO

**Regla de Procesamiento**

1. Si la sentencia es una UNION de sentencias SELECT, aplicar los pasos 2 hasta el 5 a cada una de las sentencias para generar sus resultados individuales.
2. Formar el producto de las tablas indicadas en la clausula FROM. Si la clausula FROM desina una sola tabla, el producto es esa tabla.
3. Si hay una clausula WHERE, aplicar su condición de búsqueda a cada fila de la tabla producto, reteniendo aquellas filas para las cuales la condición de búsqueda es TRUE.
4. Para cada fila restante, calcular el valor de cada elemento en la lista de selección para producir una única fila de resultados. Para una referencia de columna simple, utilizar el valor de la columna en la fila actual. Para una función de columna, utiliza como argumento el conjunto entero de filas.
5. Si se especifica SELECT DISTINCT, eliminar las filas duplicadas de los resultados que se hubieran producido.
6. Si la sentencia es una UNION de sentencias SELECT, mezclar los resultados de consulta para las sentencias individuales en una única tabla de resultados. Eliminar las filas duplicadas a menos que se haya especificado UNION ALL.
7. Si hay una clausula ORDER BY, ordenar los resultados de la consulta según se haya especificado.

**Consultas Agrupadas GROUP BY**

Se denomina consulta agrupada ya que agrupa los datos de las tablas fuentes y produce una única fila resumen por cada grupo de filas. Las columnas indicadas en la clausula se denominan columnas de agrupación de la consulta, ya que son las que determinan como se dividen las filas en grupos.

Hay una estrecha relación entre las funciones de columna SQL y la clausula GROUP BY, ya que las funciones de columna toman una columna de valores de datos y producen un resultado único. Cuando la clausula GROUP BY está presente, informa a SQL que debe dividir los resultados detallados en grupos y aplicar la función de columna separadamente a cada grupo, produciendo un único resultado por cada grupo.

**Restricciones de consultas agrupadas**

1. No pueden agruparse basándose en el valor de una expresión calculada
2. Todos los elementos de la lista de selección deben tener un único valor por cada grupo de filas, un elemento de selección en una consulta agrupada puede ser:
   1. Una **constante**
   2. Una **función de columna**, que produce un único valor que resume las filas del grupo
   3. Una **columna de agrupación**, que por definición tiene el mismo valor en todas las filas del grupo
   4. Una expresión que afecte a combinaciones de las anteriores
3. Una consulta agrupada incluirá siempre una columna de agrupación y una función de columna en su lista de selección. Sin la función columna en el SELECT la consulta puede hacerse sin el GROUP BY y usando el SELECT DISTINCT
4. Contrariamente, si no se incluye una columna de agrupación en los resultados de la consulta, no será posible decir que fila de los resultados de la consulta proviene de que grupo.

**Valores NULL en la consulta de agrupación**

El valor NULL presenta un problema, en la clausula WHERE cuando se comparan dos valores NULL diferentes, el resultado es NULL, los valores NULL no se consideran iguales. El estándar considera que dos valores NULL son iguales a efectos del GROUP BY. Si dos filas tienen NULL en las mismas columnas de agrupación y valores idénticos en las columnas de agrupación no NULL, se agrupan dentro del mismo grupo de filas.

**Condiciones de Búsqueda de grupos HAVING**

Al igual que la clausula WHERE puede ser utilizada para seleccionar y rechazar filas individuales que participan en una consulta, la clausula HAVING puede ser utilizada para seleccionar y rechazar grupos de filas. El formato de la cláusula HAVING seguida de una condición de búsqueda. Por lo tanto condiciona una la búsqueda para los grupos.

**Regla de procesamiento con GROUP BY y HAVING**

1. Si hay UNION aplicar del 2 al 7 para cada SELECT implicado.
2. Formar el producto de las tablas indicadas en el FROM.
3. Si existe una cláusula WHERE aplicar la condición de búsqueda a cada fila de la tabla producto.
4. Si existe una cláusula GROUP BY disponer las filas resultantes a modo de grupo de filas.
5. Si existe una cláusula HAVING aplicar la condición de búsqueda a cada grupo de filas.
6. Calcular los valores indicados en la lista de selección de la cláusula SELECT.
7. Si existe la cláusula DISTINCT eliminar filas duplicadas.
8. Si es una UNION de consultas, unir y eliminar filas duplicadas si no es UNION ALL.
9. Si existe la cláusula ORDER BY ordenar el resultado.

**Restricciones en condiciones de búsqueda por grupos.**

Un elemento que parezca dentro de la condición de búsqueda en la clausula HAVING puede ser:

1. Una constante
2. Una función columna, que produzca un único valor resumen de las filas del grupo
3. Una columna de agrupación, que por definición tiene el mismo valor para todas las filas del grupo
4. Una expresión que afecte a combinaciones de las anteriores

Las expresiones de comparación deben ser calculables para un grupo de filas.

**Valores NULL y el HAVING**

Si la condición de búsqueda es NULL, el grupo de filas se descarta y no contribuye una fila resumen a los resultados de la consulta.

**HAVING sin GROUP BY**

La clausula HAVING se utiliza casi siempre juntamente con la clausula GROUP BY, pero la sintaxis de la sentencia SELECT no lo precisa. Si una clausula HAVING aparece sin una clausula GROUP BY, SQL considera el conjunto entero de resultados detallados como un único grupo. En otras palabras, las funciones de columna de la clausula HAVING se aplican a un solo y único grupo para determinar si el grupo está incluido o excluido de los resultados, y ese grupo está formado por todas las filas.

**Subconsultas**

Una subconsulta es una consulta que aparece dentro de la clausula WHERE o HAVING de otra sentencia SQL. Las subconsultas proporcionan un modo eficaz y natural de gestionar peticiones de consultas que se expresan en términos de los resultados de otras consultas. En la clausula WHERE, ayudan a seleccionar las filas individuales que aparecen en los resultados de la consulta. En la clausula HAVING ayudan a seleccionar los grupos de filas que aparecen en los resultados de la consulta.

**Diferencias de la Subconsulta**

* Una subconsulta debe producir una única columna de datos como resultado.
* La clausula ORDER BY no puede ser especificada en una subconsulta. Los resultados de la subconsulta se utilizan internamente por parte de la consulta principal y nunca son visibles al usuario.
* Una sub consulta no puede ser la unión de varias sentencias SELECT diferentes.
* Los nombres de columna que aparecen en una subconsulta pueden referirse a columnas de tablas de la consulta principal.
* Debe estar encerrada entre ()
* Debe estar a la derecha del operador

¿Cómo se procesa la Subconsulta?

* + Primero se ejecuta la Subconsulta
  + El resultado pasa a la consulta principal

**Subconsultas en la clausula WHERE**

Esta funciona como parte del proceso de selección de filas. SQL efectúa la subconsulta repetidamente, una vez por cada fila examina en la clausula WHERE.

**Referencia Externa**

Dentro del cuerpo de una subconsulta, con frecuencia es necesario referirse al valor de una columna en la fila actual de la consulta principal

**Condiciones de Búsqueda en Subconsultas**

1. **Test de Comparación**, compara el valor de una expresión con el valor producido por una subconsulta, y devuelve un resultado TRUE si la comparación es cierta, si la subconsulta no produce filas o produce un valor NULL, el test de comparación devuelve NULL. El valor producido por la subconsulta debe ser único. Debe producir una única fila de resultados. El test de comparación solo se utiliza del lado derecho del operador.

Listar los empleados que estén en el mismo departamento que ‘Biri’.

SELECT last\_name, title

FROM s\_emp

WHERE dept\_id = (SELECT dept\_id FROM s\_emp

WHERE last\_name=‘Biri’);

SQL ofrece seis modos diferentes de comparar expresiones: =, <, >, <=, >=, <>

Mostrar el apellido, cargo y sueldo de todos los empleados que ganen menos que el promedio.

SELECT last\_name, title, salary

FROM s\_emp

WHERE salary < (SELECT AVG(salary) FROM s\_emp);

1. **Test de pertenencia a conjunto,** Compara un único valor de datos con una columna de valores producida por una subconsulta. La subconsulta produce una columna de valores y la clausula WHERE de la consulta principal comprueba si un valor de la consulta principal coincide con uno de los valores de la columna.

Ejemplo: nombres de empleados cuyo departamento pertenezca a la región 2

SELECT last\_name, first\_name, dept\_id

FROM s\_emp

WHERE dept\_id in (SELECT id FROM s\_dept

WHERE region\_id=2);

1. **Test de Existencia,** verifica si una subconsulta produce alguna fila de resultados. El EXISTS no utiliza los resultados de la subconsulta. Siempre contiene una referencia externa que enlaza la subconsulta a la fila que actualmente está siendo examinada por la consulta principal.

Ejemplo: lista de departamentos en los cuales existan empleados que ganen más de $2500

SELECT name, address

FROM s\_dept

WHERE exists (SELECT dept\_id FROM s\_emp

WHERE dept\_id = s\_dept.id AND salary >2500);

1. **El Test cuantificado ANY,** **ALL**, ambos test comparan un valor de dato con la columna de valores producidos por una subconsulta.
   * 1. **ANY,** se utiliza conjuntamente con los operadores (=, <, >, <=, >=, <>) para comparar un único valor de test con una columna de valores producidos por una subconsulta. Para efectuar el test, SQL utiliza el operador de comparación especificado para comparar el valor de test con cada valor de datos de la columna, uno cada vez. Para entenderlo:  
        WHERE X < ANY (SELECT Y …)  
        se lee 🡪 en donde, para algún Y, X es menor que Y

Ejemplo: listar los empleados que han aceptado una orden que supere a su sueldo

Leyéndose bien 🡪 selecciona los empleados donde, para alguna orden aceptada por el empleado, el precio de la orden supere el sueldo del empleado.

SELECT last\_name, first\_name, dept\_id

FROM s\_emp

WHERE salary <ANY (SELECT total FROM s\_ord

WHERE id = sales\_rep\_id);

El estándar especifica las reglas para describir resultados del test ANY cuando el valor de test se compara con la columna de resultados de la subconsulta:

* Si la subconsulta produce una columna vacía de resultados, el test ANY devuelve FALSE
* Si el test de comparación es TRUE para al menos uno de los valores de datos en la columna, entonces la condición de búsqueda ANY devuelve TRUE
* Si el test de comparación es FALSE para todos los valores de datos de la columna, entonces la condición de búsqueda ANY devuelve FALSE.
* Si el test de comparación no es true para ningún valor de datos en la columna, pero es NULL para uno o más valores, entonces la condición del ANY devuelve NULL.
  + 1. **ALL,** se utiliza conjuntamente con los operadores (=, <, >, <=, >=, <>) para comparar un único valor de test con una columna de valores producidos por una subconsulta. Para comprenderlo fácilmente, se especifica:

WHERE X < ALL (SELECT Y …)  
donde, para todo Y, X es menor que Y

Ejemplo: listar los empleados para los cuales, el total de cada una de sus ventas siempre ha sido superior a su sueldo

Se lee 🡪 seleccionar los empleados en donde, para todas las ventas realizadas por el empleado, el salario sea menor que el total de la ventas.

SELECT last\_name, first\_name, dept\_id

FROM s\_emp

WHERE salary <ALL (SELECT total FROM s\_ord

WHERE id = sales\_rep\_id);

El estándar especifica las reglas para describir resultados del test ALL cuando el valor de test se compara con la columna de resultados de la subconsulta:

* Si la subconsulta procesa una columna vacía de resultados, el test ALL devuelve TRUE.
* Si el test de comparación es TRUE para todos y cada uno de los valores de dato en la columna, entonces la condición de búsqueda ALL devuelve TRUE.
* Si el test de comparación es FALSE para algún valor de dato columna en la columna, entonces la condición de búsqueda ALL devuelve FALSE.
* Si el test de comparación no es FALSE para ningún valor de datos en la columna, pero es NULL para uno o más valores, entonces la condición de búsqueda ALL devuelve NULL. En esta situación no se puede concluir si existen valor producido por la subconsulta para el cual el test de comparación sea cierto.

**Subconsultas Anidadas**

Del mismo modo, se pude escribir subconsultas dentro de la subconsultas, el estándar SQL no especifica un máximo de niveles de anidación, pero en la práctica una consulta consume mucho más tiempo cuando se incrementa el número de niveles. La consulta también resulta más difícil de leer, comprender y mantener cuando contiene más de uno o dos niveles de subconsultas.

**Subconsultas correlacionadas.**

Conceptualmente SQL efectúa una subconsulta una y otra vez, una vez por cada fila de la consulta principal. Para muchas subconsultas, la subconsulta produce los mismos resultados para cada fila o grupo de filas.

Una subconsulta que contiene una referencia externa se denomina subconsulta correlacionada, ya que sus resultados están correlacionados con cada fila individual de la consulta principal, llamándose a la referencia, referencia correlacionada.

Una subconsulta puede contener una referencia externa a una tabla en la clausula FROM de cualquier consulta que contenga la subconsulta, no importa lo profundamente que las subconsultas estén animadas. Un nombre de una columna en una subconsulta de cuarto nivel, por ejemplo, puede referirse a una de las tablas indicadas en la cláusula FROM de la consulta principal, o a una tabla de cualquiera de las subconsultas que contienen la subconsulta en la cual el nombre de la columna aparece. Con independencia de nivel de anidación, una referencia externa siempre toma el valor de la columna en la fila actual de la tabla que está siendo examinada.

Puesto que una subconsulta puede contener referencias externas, hay incluso más posibilidades de que aparezcan nombres de columnas ambiguas en una subconsulta que en una consulta principal. Cuando un nombre de una columna sin cualificar aparece dentro de una subconsulta SQL debe determinar si se refiere a una tabla de la clausula FROM propia de la subconsulta o de la clausula FROM de una consulta que contiene a la subconsulta. SQL siempre interpreta una referencia a una columna en una subconsulta utilizando la clausula FROM más cercana posible.

**Subconsulta en la clausula HAVING**

Cuando una subconsulta aparece en esta clausula, funciona como parte de la selección de grupos de filas efectuada por la clausula HAVING.

También se puede utilizar una subconsulta correlacionada en la clausula HAVING, pero, la subconsulta se evalúa una vez por cada grupo de filas, todas las referencias externas de la subconsulta correlacionada deben tener un solo valor por cada grupo de filas. Esto es, que la referencia externa debe ser o bien una referencia a una columna de agrupación de la consulta externa o bien estar contenida dentro de una función de columna, esto sería que el valor de la función de columna para el grupo de filas que está siendo examinado se calcula como parte del procesamiento de la subconsulta.

**Reglas de procesamiento**

1. Comenzar con la tabla designada en FROM
2. Si hay WHERE aplicar la condición, si hay subconsulta aplicarla a cada fila
3. Si existe GROUP BY, hacer grupos de filas de acuerdo a las columnas de agrupación
4. Si hay una cláusula HAVING aplicarla a cada grupo. Si contiene una Subconsulta aplicarla para cada grupo
5. Armar la lista de selección
6. Si hay SELECT DISTINCT, eliminar duplicados
7. Si hay cláusula ORDER BY, aplicarla

**Actualizaciones de bases de datos**

1. **Añadir Datos**

Típicamente una nueva fila de datos se añade a una base de datos relacional cuando una nueva entidad representada por una fila “aparece en el mundo exterior”.

La unidad más pequeña que puede añadirse a una base de datos relacional es una fila. Un DBMS basado en SQL proporciona 3 maneras de añadir nuevas filas de datos a una base de datos:

1. Sentencia INSERT de una fila, añade una única nueva fila de datos a una tabla.
2. Sentencia Insert multifila, extrae filas de datos de otra parte de la base de datos y las añade a una tabla.
3. Utilidad de carga masiva, extrae datos a una tabla desde un archivo externo a la base de datos. Se utiliza para cagar datos inicialmente.

**Sentencia INSERT:** Añade una fila a una tabla. La clausula INTO específica la tabla que recibe la nueva fila y la clausula VALUE específica los valores de los datos que contendrá la nueva fila.

**Inserción de Valores NULL**

Cuando SQL inserta una nueva fila de datos en una tabla, automáticamente asigna un valor NULL a cualquier columna cuyo nombre falte en la lista de columnas de la sentencia INSERT.

Inserción de todas las columnas. SQL permite omitir la lista de columnas de la sentencia INSERT. Cuando se omite la lista de columnas, SQL genera automáticamente una lista formada por todas las columnas de la tabla, en secuencia de izquierda a derecha. Esta es la misma secuencia de columnas generadas por SQL cuando se utiliza una consulta Select \*.

**Sentencia INSERT Multifila**

La segunda forma de la sentencia INSERT añade múltiples filas de datos a su tabla destino. En esta forma de la sentencia, los valores de datos para las nuevas filas no son especificados explícitamente dentro del texto de la sentencia.

INSERT INTO TABLA (VAL1, VAL2, VAL3, ….)

SELECT VAL1, VAL2, VAL3, ….

FROM TABLA1

WHERE condición.

**Restricciones del estándar**

1. La consulta no puede contener una clausula ORDER BY. No es útil.
2. El resultado de la consulta debe contener el mismo número de columnas que hay en la lista de columnas de la sentencia INSERT y los tipos de los datos deben ser compatibles, o sea el mismo dominio.
3. La consulta no puede ser una UNION de varias sentencias SELECT diferentes.
4. La tabla destino de la sentencia INSERT no puede aparecer en la clausula FROM de la consulta o de ninguna subconsulta que esta contenga.
5. **Supresión de Datos**

Una fila de datos se suprime de una base de datos cuando la entidad representada por la fila desaparece del mundo real.

**Sentencia DELETE**

DELETE FROM

WHERE condición.

Esta sentencia elimina filas seleccionadas de datos de una única tabla. La clausula FROM específica la tabla destino que contiene las filas. La clausula WHERE especifica que filas de la tabla van a ser suprimidas. La condición de búsqueda que pueden especificarse en la clausula WHERE son las mismas que hay disponibles en la clausula WHERE de la sentencia SELECT.

DELETE busca a través de la tabla las filas a suprimir, a veces se le denomina sentencia DELETE buscada.

**Supresión de todas las filas.**

La clausula WHERE en la sentencia DELETE es opcional, si esta se omite, DELETE elimina todas las filas de la tabla.

**DELETE con Subconsulta**

Las subconsultas pueden jugar un papel importante en la sentencia DELETE, ya que permiten suprimir filas en base a información contenida en otras tablas. Las referencias externas aparecerán con frecuencia en las subconsultas de una sentencia DELETE, ya que implementaran la composición entre la/s tabla/s de la subconsulta y la tabla destino de la sentencia DELETE. La única restricción al uso de subconsultas en una sentencia DELETE es que la tabla destino no puede aparecer en la clausula FROM de la subconsulta. El estándar SQL2 suprime esta restricción especificando que la sentencia DELETE trate a la subconsulta como si aplicara a la tabla destino completa, antes de que ninguna fila haya sido suprimida.

1. **Modificación de la Base de Datos**

**Sentencia UPDATE**

Esta sentencia modifica los valores de una o más columnas en las filas seleccionadas de una tabla única. La tabla destino a actualizar se indica en la sentencia y es necesario disponer de permiso para actualizar la tabla así como cada una de las columnas individuales que serán modificadas. La clausula WHERE selecciona las filas de la tabla a modificar. La clausula SET especifica que columnas se van a actualizar y calcula los nuevos valores.

UPDATE TABLA SET Columna = modificación, …

**WHERE columna condición.**

La clausula SET es una lista de asignaciones separadas por comas. Cada una identifica una columna destino a actualizar y especifica como calcular el nuevo valor para la columna destino. Cada columna destino debería aparecer solamente una vez en la lista; no debería haber dos asignaciones para la misma columna destino. El estándar exige nombres sin cualificar para las columnas destino, pero algunas implementaciones SQL permiten nombres de columna cualificados.

**Actualización de todas las Filas**

La clausula WHERE es opcional. Si se omite la clausula, efectúa una actualización masiva en toda la tabla.

**UPDATE con subconsulta**

Al igual que DELETE, las subconsultas pueden jugar un papel importante en la sentencia. Ya que permite seleccionar las filas a actualizar en base a información contenida en otras tablas.

Al igual que en DELETE, pueden anidarse a cualquier nivel y pueden contener referencias externas a la tabla destino de la sentencia. Ídem con el estándar SQL1 y luego del estándar SQL2

**Integridad de Datos**

Para preservar la consistencia y la corrección de los datos almacenados, un DBMS relacional impone típicamente una o más restricciones de integridad de datos. Estas restricciones restringen los valores que pueden ser insertados en la base de datos o creados mediante una actualización de la base de datos. Varios tipos diferentes de restricciones de integridad de datos suelen encontrarse en las bases de datos relaciones.

1. **Datos requeridos**  
   la restricción de integridad de datos más simple requiere que una columna tenga un valor no NULL. Esta restricción NOT NULL se especifica como parte de la sentencia CREATE TABLE.

la sentencia INSERT, debe especificar un valor de datos no nulo para la columna. Un intento de hacerlo da lugar a un error.

* 1. la sentencia UPDATE, al actualizar un dato, debe ser por uno no nulo, o dará un error.

La incapacidad de añadir una restricción NOT NULL a una tabla existente es resultado del método que la mayoría de los productos DBMS utilizan para implementar los valores NULL internamente.

1. **Chequeo de validez**

Cada columna al crearse la tabla, tiene asignado un tipo de dato, el DBMS asegura que únicamente datos del tipo especificado sean introducidos en la columna. El procedimiento de validación comprueba los datos e indica mediante un valor de retorno si los datos son aceptables.   
El estándar SQL2 proporciona un soporte ampliado para comprobar la validez a través de una restricción de comprobación, que es una condición de búsqueda, como la de la clausula WHERE que produce un valor TRUE/FALSE. Cuando se especifica una restricción de comprobación para una columna, el DBMS automáticamente comprueba el valor de esa columna cada vez que se inserta una nueva fila o que se actualiza una fila. Una restricción de comprobación se especifica como parte de la sentencia CREATE TABLE.

1. **Integridad de entidad**la clave primaria de una tabla debe tener un valor único para cada fila de la tabla o si no la base de datos perderá su integridad como modelo del mundo exterior. Por ello la razón de la exigencia de que las claves primarias tengan valores únicos se denomina restricción de integridad entidad. El DBMS comprueba automáticamente la unicidad del valor de la clave primaria con cada sentencia INSERT y UPDATE. Un intento de insertar una fila con una clave duplicada, falla.
   1. **Otras restricciones de unidad**, a veces es apropiado exigir que una columna que no es clave primaria de una tabla contenga un valor único en cada fila. El DBMS fuerza una restricción de unicidad del mismo modo que la fuerza la restricción de clave primaria. El estándar utiliza la sentencia CREATE TABLE para especificar restricciones de unicidad en columnas o combinaciones de columnas.
   2. **Unicidad y Valores NULL**, los valores NULL presenta un problema cuando aparecen en la clave primaria de una tabla o en una columna que esta especificada en una restricción de unicidad. Por lo tanto SQL requiere que toda columna que forma parte de una clave primaria y toda columna designada en una restricción de unicidad deben ser declaradas NOT NULL.
2. **Integridad referencial**, asegura la integridad de las relaciones padre/hijo creadas mediante claves ajenas y claves primarias.
   1. **Problemas de integridad referencial:**
      1. **La inserción de una nueva fila hijo**, cuando se inserta una nueva fila en la tabla hijo, su valor clave ajena debe coincidir con uno de los valores de clave primaria en la tabla padre. Si el valor de la clave ajena no coincide con ninguna clave primaria, la inserción de la fila corromperá la base de datos, ya que habrá un hijo sin padre. Se maneja comprobando los valores de las columnas de la clave ajena antes de permitir ejecutar la sentencia INSERT. Si no coincide, se rechaza.
      2. **La actualización de la clave ajena en una fila hijo**, si la clave ajena se modifica mediante una secuencia de UPDATES, el nuevo valor debe coincidir con un valor de clave primaria en la tabla padre. En caso contrario la fila actualizada no tendrá padre. Es análogamente tratado mediante comprobación del valor de clave ajena actualizada.
      3. **La supresión de una fila padre**, si la tabla padre que tiene uno o más hijos se suprime, las filas hijo quedaran sin padre. Los valores de clave ajena en estas filas ya no se corresponderán con ningún valor de calve primaria en la tabla padre. Para resolver esto es más complejo, dependiendo de la situación se podrá:
         1. **Impedir la supresión hasta reasignar los hijos.**
         2. **Suprimir las filas hijos automáticamente.**
         3. **Seteando un valor null a todos los hijos.**
         4. **Seteando un valor por defecto.**
      4. **La actualización de la clave primaria.** Esta es una forma diferente dl problema anterior. Si la clave primaria de una fila en la tabla padre se modifica, todos los hijos actuales de esa fila quedan huérfanos, puesto que sus claves ajenas ya no corresponden con ningún valor de la clave primaria. Se controla también por prohibición, antes de que la PK pueda ser modificada, el DBMS efectúa comprobaciones para asegurarse de que no haya filas hijo que tengan valores de clave ajena correspondientes. Dependiendo de la situación:
         1. **Impedir la supresión hasta reasignar los hijos.**
         2. **Suprimir las filas hijos automáticamente.**
         3. **Seteando un valor null a todos los hijos.**
         4. **Seteando un valor por defecto.**
   2. **Reglas de Supresión y actualización.**

Por cada relación padre hijo, se puede especificar una regla de supresión asociada y una regla de actualización asociada. La regla de supresión le dice al DBMS que debe hacer cuando un usuario trate de suprimir una fila de la tabla padre.

1. La Regla RESTRICT impide suprimir una fila de la tabla padre si la fila tiene algún hijo.
2. La regla CASCADE le dice al DBMS que cuando una fila padre se suprima, todas sus filas hijo también deberían ser suprimidas automáticamente de la tabla hijo.
3. La regla de supresión SET NULL, le dice al DBMS que cuando una fila padre sea suprimida, los valores de la clave ajena en todas las filas hijo deben automáticamente pasarse a NULL
4. La regla de supresión SET DEFAULT le dice al DBMS que cuando una fila padre se suprimida, los valores de clave ajena en todas las filas hijo deben automáticamente pasarse a un valor por defecto para esa columna particular.

La regla de actualización le dice al DBMS que hacer cuando un usuario intenta actualizar el valor de una de las columnas de clave primaria en la tabla padre.

1. La Regla RESTRICT impide actualizar la clave primaria de una fila de la tabla padre si la fila tiene algún hijo.
2. La regla CASCADE le dice al DBMS que cuando el valor de una clave primaria de una fila padre se actualiza, se modifican las claves ajenas en todas sus filas hijo también deberían ser actualizadas automáticamente de la tabla hijo.
3. La regla de actualización SET NULL, le dice al DBMS que cuando el valor de la clave primaria de una fila padre sea modificada, los valores de la clave ajena en todas las filas hijo deben automáticamente pasarse a NULL
4. La regla de actualización SET DEFAULT le dice al DBMS que cuando un valor de la clave primaria de una fila padre se modificada, los valores de clave ajena en todas las filas hijo deben automáticamente pasarse a un valor por defecto para esa columna particular.
   1. **Supresiones y actualizaciones en CASCADE**

La regla RESTRICT es una regla de un nivel que afecta únicamente a la tabla padre de una relación. La Regla CASCADE, es una regla multinivel. Las reglas de supresión CASCADE deben ser especificadas con cuidado, ya que pueden provocar la supresión automática global de datos si fueran utilizadas incorrectamente. Las reglas de actualización en cascada también pueden producir actualizaciones en cascada si la clave ajena en la tabla hijo es también su clave primaria.

* 1. **Claves Ajenas y Valores NULL**a diferencia de las claves primarias, las claves ajenas de una base de datos relacional permite que contengan un valor NULL. Este valor presenta una cuestión referente a la restricción de integridad referencial creada por la relación calve primaria/clave ajena. Le corresponde el valor NULL, tanto el DB2 como en el estándar, suponen automáticamente que una clave ajena que contiene un valor NULL satisface la restricción de integridad referencial. Debería tener precaución cuando la clave ajena es compuesta, ya que el valor NULL debe ser para todos las columnas que componen la clave.
     1. **MATCH FULL,** requiere que las calves ajenas de la tabla hijo coincidan totalmente con una clave primaria de la tabla padre.
     2. **MATCH PARTIAL** permite valores null en partes de una clave ajena, mientras que los valores que no son null coinciden con las partes correspondientes de alguna clave primaria de la tabla padre.

1. **Reglas comerciales**

Estas tienen que ver con las reglas y procedimientos de una organización. Además, con frecuencia existen reglas de contabilidad que deben ser seguidas para mantener una integridad de la base, cuentas, etc. Dejar la tarea de forzar las reglas comerciales a los programas de aplicación que acceden a las bases de datos tiene varias desventajas:

1. **Duplicación de esfuerzo**
2. **Falta de consistencia**
3. **Problemas de mantenimiento**
4. **complejidad**
5. **Consistencia,** muchas transacciones del mundo real producen múltiples actualizaciones a una base de datos. El DBMS debe ser capaz de forzar este tipo de reglas de consistencias o para soportar aplicaciones que implementen tales reglas, como generar un pedido a un cliente, requiere de 1 INSERT y 2 UPDATES

**Procedimientos Almacenados**

Colección de sentencias SQL que se almacenan en la BD, en el servidor. Aceptan parámetros de entrada y pueden devolver varios valores. Usuarios pueden tener permiso de ejecutar y no derechos sobre objetos. Permiten programación modular y convocarlo desde múltiples aplicaciones

**Disparador / TRIGGERS**

Clase especial de Procedimiento Almacenado. Para cualquier evento que provoca un cambio en un contenido de una tabla, un usuario puede especificar una acción asociada que el DBMS debería efectuar. Los tres eventos que pueden disparar un disparador son un intento de INSERT, UPDATE, DELETE.

**Disparador e Integridad Referencial:** Los disparadores proporcionan un modo alternativo de implementar las restricciones de integridad referencial proporcionadas por claves ajenas y claves primarias.Los disparadores también pueden ser utilizados para proporcionar formas ampliadas de integridad referencial.

**Ventajas e Inconvenientes**

La principal ventaja de los disparadores es que las reglas comerciales pueden almacenarse en las bases de datos en las bases de datos y ser forzadas consistentemente con cada actualización de la base de datos. Reduciendo sustancialmente la complejidad de los programas de aplicación que acceden a la base de datos.

1. **Complejidad en la base de datos**: cuando las reglas se trasladan al interior de una base de datos, preparar la base de datos pasa a ser una tarea más compleja. Los usuarios que razonablemente podían esperar crear pequeñas aplicaciones propias con SQL, encontraran que la lógica de programación de los disparadores es más difícil.
2. **Reglas ocultas**: con las reglas ocultas dentro de la base de datos, programas, que parecen efectuar sencillas actualizaciones de la base de datos pueden generar una cantidad enorme de actividad sobre la base de datos. El programador ya no controla totalmente lo que sucede en la base de datos.

**Procesamiento de Transacciones**

Una transacción es una secuencia de una o más sentencias SQL que juntas forman una unidad de trabajo. Las sentencias SQL que forman la transacción suelen estar estrechamente relacionadas y efectuar acciones interdependientes. Cada sentencia de una transacción efectúa una parte de una tarea, pero todas ellas son necesarias para completar la tarea. La agrupación de las sentencias en una sola transacción indica al DBMS que la secuencia de sentencias entera debe ser ejecutada atómicamente, todas las sentencias deben completarse para que la base de datos esté en un estado consistente.

Un DBMS basado en SQL efectúa este compromiso referente a las sentencias de una transacción:

Las sentencias de una transacción se ejecutaran como una unidad atómica de trabajo en la base de datos. O todas las sentencias son ejecutadas con éxito, o ninguna de las sentencias es ejecutada

COMMIT y ROLLBACK

SQL soporta las transacciones de base de datos mediante dos sentencias de procesamiento de transacciones SQL.

La sentencia COMMIT señala el final correcto de una transacción. Informa al DBMS que la transacción esta ahora completa, todas las sentencias que forman la transacción han sido ejecutadas y la base de datos es autoconsistente.

La sentencia ROLLBACK señala el final sin éxito de una transacción. Informa al DBMS que el usuario no desea completar la transacción, en vez de ello, el DBMS debe deshacer los cambios efectuados a la base de datos durante la transacción. El DBMS restaura la base de datos a su estado antes de que la transacción comenzara.

**Modelo de Transacción ANSI/ISO**

El estándar SQL ANSI/ISO define un modelo de transacción SQL y los papeles de las sentencias COMMIT y ROLLBACK. El estándar especifica que una transacción SQL comienza automáticamente con la primera sentencia SQL ejecutada, por usuario o un por un programa. La transacción continúa con las sentencias SQL subsiguientes hasta que finaliza de uno de cuatro modos posibles

1. Sentencia COMMIT, finaliza la transacción con éxito. Una nueva transacción comienza inmediatamente después de la sentencia COMMIT.
2. Sentencia ROLLBACK, aborta la transacción deshaciendo las modificaciones que haya efectuado a la base de datos. Una nueva transacción comienza inmediatamente después de una sentencia ROLLBACK.
3. Terminación de un programa con éxito (para SQL programado) también finaliza la transacción con éxito, igual como si hubiera ejecutado una sentencia COMMIT. Puesto que el programa está finalizado, no hay ninguna nueva transacción que comenzar
4. Terminación anormal del programa (para SQL programado), también aborta la transacción, del mismo modo que si se hubiera ejecutado una sentencia ROLLBACK. Puesto que el programa está finalizado, no hay ninguna nueva transacción que comenzar.

Generalmente, un DBMS utiliza un registro de transacción (Log.) para las transacciones.

Cuando un usuario ejecuta una sentencia SQL que modifica la base de datos, el DBMS escribe automáticamente una anotación el registro de transacción mostrando dos copias de cada fila afectada por la sentencia. Una copia la fila antes y la otra la fila después. Solo después de que el DBMS realmente escribe el registro modifica la fila en disco. Si el usuario ejecuta posteriormente una sentencia COMMIT, el fin de la transacción se anota en el registro de transacción. Si el usuario ejecuta una sentencia ROLLBACK, el DBMS examina el registro para encontrar las imágenes de las filas que han sido modificadas desde que comenzó la transacción. Utilizando estas imágenes, el DBMS restaura las filas a su estado anterior, deshaciendo efectivamente todas las modificaciones a la base de datos efectuadas durante la transacción.

Si se produce un fallo del sistema, el operador típicamente recupera la base de datos ejecutando una utilidad de recuperación especial suministrada con el DBMS.

**Transacciones y procesamiento Multiusuario.**

Cuando dos o más usuarios acceden concurrentemente a una base de datos, el procesamiento de transacciones toma una nueva dimensión. Ahora el DBMS no solamente debe recurarse adecuadamente de los fallos o errores del sistema, también debe asegurarse que las acciones de los usuarios debería ser capaz de acceder a la base de datos como si tuviera acceso exclusivo a ella, sin preocuparse de las acciones del resto de los usuarios. El modelo de transacción SQL permite a un DBMS basado en SQL aislar a los usuarios unos de otros de este modo.

1. **Problema de la actualización perdida**: se da cuando dos programas leen los mismos datos y utilizan los datos como base para un cálculo y luego tratan de actualizar los datos.
2. **Problema de los datos no confirmados**
3. **Problema de los datos inconsistentes**
4. **Problema de la Inserción fantasma**

**Transacciones Concurrentes**

Durante una transacción, el usuario vera una vista completamente consistente de la base de datos. El usuario nunca contemplará modificaciones no confirmadas de otros usuarios, e incluso los cambios confirmados efectuados por otros no afectarán a los datos examinados por el usuario en mitad de una transacción.

Si dos transacciones, A y B, se están ejecutando concurrentemente, el DBMS asegura que los resultados serán los mismos en cualquier caso tanto si:

1. la transacción A se ejecutara primero, seguida de la Transacción B, como si
2. La transacción B se ejecutara primero, seguida de la transacción A.

Este concepto es conocido como serialidad de las transacciones. Efectivamente, significa que cada usuario de base de datos puede acceder a la base de datos como si no hubiera otros usuarios accediendo concurremente en ella.

**Cerramiento**

La técnica del cerramiento proporciona a una transacción acceso temporal exclusivo a una parte de una base de datos, impidiendo que otras transacciones modifiquen los datos encerrados**. El cerramiento resuelve por tanto los tres problemas de transacción concurrente. Impide que las actualizaciones perdidas, los datos no confirmados, y los datos inconsistentes puedan corromper la base de datos**. Sin embargo, el cerramiento introduce un nuevo problema, pudiendo hacer que una transacción espere mucho tiempo mientras las partes de la base de datos que desea acceder están bloqueados.

**Niveles de Cerramiento:**

1. **Nivel de las BD**, en su forma menos elaborada el DBMS puede bloquear la base entera por cada transacción. Fácil de implementar, pero solo permite un solo proceso a la vez.
2. **Nivel de Tablas**, DBMS bloquea aquellas tablas accedidas por una transacción
3. Nivel de Página, en este esquema, el DBMS bloquea los grupos individuales de datos procedentes del disco, conforme son accedidos por una transacción.
4. **Nivel de Fila**, permite que dos transacciones diferentes y concurrentes accedan a dos filas diferentes en una misma tabla, incluso si están en el mismo bloque del disco.
5. **Nivel de Datos**, el cerramiento a nivel de datos es una teoría, donde proporcionaría más paralelismo que los cerramientos a nivel de fila.

**Cierres compartidos y exclusivos**

Para aumentar el acceso concurrente a una base de datos, la mayoría de los productos DBMS comerciales utilizan un esquema de cerramiento con más de un tipo de cierra. Un esquema que utiliza cierras compartidos y exclusivos es bastante habitual.

Un **cierre compartido** se utiliza en el DBMS cuando una transacción desea leer datos de la base de datos. Otra transacción concurrente puede también adquirir un cierre compartido sobre los mismos datos, permitiendo que la otra transacción también lea los datos.

Un **cierre exclusivo** se utiliza en el DBMS cuando una transacción desea actualizar datos en la base de datos. Cuando una transacción tiene un cierre exclusivo sobre algunos datos, el resto de las transacciones no puede adquirir ningún tipo de cierre sobre los datos.

**Interbloqueos**

Para tratar los interbloqueos, un DBMS normalmente incluye una lógica que periódicamente comprueba los cierres mantenidos por varias transacciones. Cuando detecte un interbloqueo, el DBMS elige arbitrariamente una de las transacciones como perdedora del interbloqueo y le da marcha atrás. Esto libre los cierres mantenidos por la transacción perdedora, permitiendo a la otra transacción proseguir.

La probabilidad de interbloqueo puede reducirse enormemente con un planeamiento cuidadoso de las actualizaciones de la base de datos.

**Técnicas avanzadas de cerramiento:**

**Cerramiento Explicito**: su una transacción accede repetidamente a una tabla, el recargo de adquirir pequeños cierres sobre muchas partes de la tabla puede ser muy sustancial. Un programa de actualización masiva que recorre todas las filas de la tabla, cerrara la tabla entera, pieza a pieza, conforme vaya procediendo. Para este tipo de transacción, el programa debería cerrar explícitamente la tabla entera, procesar las actualizaciones y luego desbloquear la tabla. Cerrar la tabla entera tiene tres ventajas:

1. Elimina el recargo de cierre de fila a fila
2. Elimina la posibilidad de que otra transacción cierre parte de la tabla, forzando a la transacción de actualización masiva.
3. Elimina la posibilidad de que otra transacción cierre parte de la tabla e interbloquee la transacción de actualización masiva, forzándola a ser reiniciada.

Como desventaja: las otras transacciones que intenten acceder a ella deben esperar mientras la actualización está en proceso.

LOCK TABLE TABLA

* **EXCLUSIVE**: cierre exclusivo sobre la tabla entera. Ninguna otra transacción puede acceder.
* **SHARE**: adquiere un cierre compartido sobre la tabla entera. Otras transacciones pueden leer partes de la tabla, pero no pueden actualizar.

**Niveles de Aislamiento**: bajo la estricta definición de una transacción SQL ninguna acción por parte de una transacción en ejecución concurrente se permite que afecte a los datos visibles durante el curso de una transacción dada. Si un programa efectúa una consulta de base de datos durante una transacción, prosigue con otro trabajo y posteriormente efectúa la misma consulta una segunda vez, el mecanismo de transacción SQL garantiza que los datos devueltos por las dos consultas serán idénticos. Esta capacidad de devolver a recuperar fiablemente una fila durante una transacción es el nivel más alto de aislamiento que un programa puede tener frente a otros programas y usuarios. El nivel de aislamiento se denomina nivel de aislamiento de la transacción.

El aislamiento absoluto de la transacción respecto al resto de transacciones que se están ejecutando de manera concurrente es muy costoso en términos de cerramiento de la base de datos.

El nivel de aislamiento **SERIALIZABLE** es el nivel más alto proporcionado. En este nivel, el DBMS garantiza que los efectos de la ejecución concurrente de transacciones son exactamente los mismos que si se ejecutaran en secuencia.

El nivel de aislamiento **LECTURA REPETIDA** en este nivel, la transacción no puede ver actualizaciones confirmadas o no confirmadas de otras transacciones por lo que no se pueden producir los problemas de actualización perdida, datos no confirmados, ni datos modificados. Sin embargo una fila insertada en la base de datos por otra transacción concurrente puede llegar a ser visible por la transacción.

El nivel **LECTURA CONFIRMADA**, la transacción no puede ver actualizaciones no confirmadas de otras transacciones, por lo que no se puede producir el problema de actualización perdida ni el problema de datos no confirmados.

**LECTURA SIN CONFIRMAR**, es el nivel más bajo, las transacciones puede verse afectada por las actualizaciones confirmadas o no confirmadas de otras transacciones, por lo que se pueden producir los problemas de los datos no confirmados, datos modificados e inserciones fantasmas.

**Parámetros de Cerramiento:** son fijados por el DBA para mejorar las prestaciones de estos sistemas fijando manualmente parámetros de cerramiento.

1. **Tamaño de cierre:** algunos productos DBMS ofrecen la opción de cierres a nivel de tabla, página, fila, y otros. Dependiendo la aplicación específica puede ser adecuado un tamaño de cierre diferente.
2. **Numero de cierre:** un DBMS permite para cada transacción tenga un cierto número finito de cierres. El administrador de la base de datos puede frecuentemente fijar ese límite, elevándolo para permitir transacciones más complejas, o rebajándolo para estimular escalas de cierres previas.
3. **Estaca de Cierre:** con frecuencia un DBMS escalará automáticamente los cierre, reemplazando muchos cierres pequeños con un único cierre mayor.
4. **Plazo de cierre:** aun cuando la transacción no se interbloquee con otra, puede tener que esperar mucho tiempo a que otras transacciones liberen esos cierres. (time out)

**Estructura de la Base de Datos**

Los cambios de la estructura de la base de datos se realizan con un conjunto diferente de sentencias SQL, denominadas conjuntamente DDL, lenguaje de definición de datos.

Las sentencias DML pueden modificar los datos almacenados, pero no cambian su estructura.

Un lenguaje DDL puede:

* Definir y crear nuevas tablas
* Suprimir una tabla que ya no es necesaria
* Cambiar la definición de una tabla existente
* Definir una tabla virtual de datos
* Establecer controles de seguridad para una base de datos.
* Construir un índice para hacer mas rápido el acceso a la tabla
* Controlar el almacenamiento físico de los datos por parte del DBMS

**Creación de una base de datos**

Definiciones de Tablas: Es la estructura más importante de una base de datos relaciona.

CREATE TABLE NAMETABLE Define una nueva tabla en la base de datos.

**Clausuras:**

1. **Columnas**: las columnas se definen en el cuerpo de la sentencia. Las definiciones de columnas aparecen en una lista separada por comas e incluida entre paréntesis. Nombre de la columna, tipo de dato que identifica la clase de datos que la columna almacena, si la columna contiene datos requeridos se usa la clausula not null, y valores por omisión opcional para la columna.

Ejemplo:

CREATE TABLE OFICINAS  
(OFICINA INTEGER NOT NULL,  
CIUDAD VARCHAR (15) NOT NULL,  
DIR INTEGER DEFAUL T 106)

1. **Definición de clave primaria y ajena:** Se utiliza las clausulas PRIMARY KEY / FOREING KEY.

La clausula PRIMARY KEY específica la columna o columnas que forman la clave primaria. Esta columna debe ser not null.

La clausura FOREING KEY especifica una clave ajena en la tabla y la relación que crea con otra tabla de la base de datos.

1. **Restricción de unicidad:** el estándar ANSI/ISO especifica que las restricciones ee unicidad también se definen en la sentencia CREATE TABLE, utilizando la clausula UNIQUE. Muchos DBMS utilizan CREATE INDEX.
2. **Definición de almacenamiento físico:** Esta sentencia también incluye clausulas opcionales que especifican características de almacenamiento físico para la tabla. Esto los usa el DBA para optimicar el rendimiento de una base de datos de producción.

**Eliminación de una Tabla**

DROP TABLE TABLANAME 🡪 [CASCADE/RESTRICT] estas sentencias, del estándar SQL2, especifican el impacto que tiene la eliminación de una tabla sobre otros elementos de la base de datos que dependen de la tabla.

**Modificación de una definición de tabla**

**ALTER TABLE** TABLENAME:

* ADD definición de la columna
* ALTER nombre de la columna SET DEFAULT valor
* ALTER nombre de la columna DROP DEFAULT
* DROP nombre de la columna CASADE / RESTRICT
* ADD definición de la clave primaria
* ADD definición de la clave ajena
* ADD restricción de unicidad
* ADD restricción comprobación
* DROP CONSTRAINT nombre de la restricción CASCADE / RESTRICT

La sentencia es utilizada añadir una columna a la tabla existente y para modificar claves primarias o ajenas, añadiendo definiciones de claves.

**Definiciones de Restricción** (CONSTRAINS)

1. Restricción de unicidad: obliga que los datos de una columna o combinación de columnas sean únicos en cada una de las filas de la tabla.
2. Restricción de clave primaria, al igual que el de unicidad, pero también designa una columna o combinación de columnas como la clave de la fila, en el entorno de una relación padre/hija con otra tabla.
3. Restricción de clave ajena, establece que el valor de una columna o combinación de columnas coincida con el valor de una clave primaria en alguna tabla padre.

Cada una de estas restricciones se aplica a una tabla sencilla, que se especifica en la sentencia CREATE TABLE y que puede ser modificada con la sentencia ALTER TABLE.

**Restricciones de Comprobación**: es una restricción de la base de datos que limita el contenido de una tabla particular. El estándar SQL2, se especifica como una condición de búsqueda y aparece como parte de la definición de la tabla.  
**Aserciones**: una aserción es una restricción de la base de datos que restringe los contenidos de la base de datos en su conjunto. Se especifica como una condición de búsqueda. Pero a diferencia de la restricción de comprobación, la condición de búsqueda en una aserción puede restringir el contenido de múltiples tablas y los datos de las relaciones entre ellas. Una aserción se especifica como parte de la definición de lavase de datos completa, utilizando la sentencia CREATE ASSERTION de SQL2.   
**Definiciones de Dominio:** un dominio de una columna es un conjunto nominado de valores de datos que realmente funciona como un tipo de dato adicional, para utilizarlo en la definición de la base de datos. Se crea con la sentencia CREATE DOMAIN.  
**Índices:** Un índice es una estructura que proporciona un acceso rápido a las filas de una tabla en base a los valores de una o más columnas. El DBMS utiliza el índice del mismo que el de un libro. La indexación es apropiada cuando las consultas de una tabla son más frecuentes que las inserciones y las actualizaciones. El DBMS siempre estable un índice para la clave primaria de una tabla.

**Vistas**

Una vista es una tabla virtual en la base de datos cuyos contenidos están definidos por una consulta. Una vista no existe en una base de datos como un conjunto almacenado de valores, las filas y columnas de datos visibles a través de la vista son los resultados producidos por la consulta que define la vista.

Cuando el DBMS encuentra una referencia a una vista en una sentencia SQL determina la definición de la vista almacenada en la base de datos. Luego el DBMS traduce la petición que referencia a la vista a una petición equivalente con respecto a las tablas fuente de la vista y lleva a cabo la petición equivalente. De este modo el DBMS mantiene la ilusión de la vista mientras mantiene la integridad de las tablas fuentes.

**Ventajas**:

1. **Seguridad**: cada usuario puede obtener permisos para acceder ala base de datos únicamente a través de un pequeño conjunto de vistas que contienen los datos específicos que el usuario está autorizado a ver, restringiendo así el acceso del usuario a los datos almacenados.
2. **Simplicidad de consulta**: una vista puede extraer datos de varias tablas diferentes y presentarlos como una única tabla, haciendo que consultas multitabla se formulen como consultas de una sola tabla con respecto a la vista.
3. **Simplicidad estructurada**: las vistas pueden dar a un usuario una visión de la estructura dela base de datos presentando esta como un conjunto de tablas virtuales que tienen sentido para ese usuario.
4. **Aislamiento frente al cambio**. Una vista puede presentar una imagen consistente inalterada de la estructura de la base de datos, incluso si las tablas fuente subyacentes se dividen, reestructuran o cambien de nombre.
5. **Integridad de datos**. Si se accede a los datos y se introducen a trasvés de una vista, el DBMS puede comprobar automáticamente los datos para asegurarse que satisfacen restricciones de integridad especificadas.

**Desventajas:**

1. **Rendimiento**: las vistas crean la apariencia de una tabla, pero el DBMS debe traducir las consultas con respecto a la vista en consultas con respecto a las tablas fuentes subyacentes. Si la vista se define mediante una consulta multitabla compleja, entonces incluso una consulta sencilla con respecto a la vista se convierte en una composición complicada y puede tardar mucho tiempo en completarse.
2. **Restricciones de actualización**: cuando un usuario trata de actualizar filas de una vista, el DBMS debe traducir la petición a una actualización sobre las filas de la tablas fuentes subyacentes. Esto es posible para vistas sencillas, pero vistas más complejas no pueden ser actualizadas, son de solo lectura.

**Creación de una vista:**

**CREATE VIEW**  esta sentencia se utiliza para crear una vista, a la sentencia se le asigna un nombre y luego se especifica la consulta, para que la vista se cree con éxito, se debe tener los permisos necesarios para acceder a todas las tablas referenciadas en la consulta.

**Vistas Horizontales:** Es un corte horizontal en las tablas involucradas en la vista para restringir el acceso de usuarios.

**Vistas Verticales:** Restringe el acceso a usuario solo de ciertas columnas de una tabla.

**Vistas con subconjunto fila/columna**

**Vistas Agrupadas:** La consulta en la definición de vista puede incluir una clausula GROUP BY, este tipo de vista se denomina agrupada. Ya que los datos visibles a través de ella son el resultado de una consulta agrupada.

**Vistas Compuestas:** Simplifican las consultas multitabla, es una vista que extrae sus datos de dos o más tablas diferentes y presenta los resultados de la consulta como una única tabla virtual. **Actualización de Vistas**

Restricciones: SQL 1 ANSI/ISO

1. No debe especificarse DISTINCT
2. La clausula FROM debe especificar solamente una tabla actualizable
3. Cada elemento de selección debe ser una referencia de columna simple
4. La clausula WHERE no debe incluir una subconsulta, solo pueden aparecer condiciones de búsqueda simple fila a fila.
5. La consulta no debe incluir una cláusula GROUP BY o HAVING

**Eliminación de una Vista**

Para eliminar una vista se tiene la sentencia DROP VIEW  
El estándar SQL2 formalizo el soporte para la eliminación de vistas a través de la sentencia DROP VIEW, teniendo en cuenta lo que ocurre cuando un usuario intenta eliminar una vista y la definición de otra vista depende de ella, agregando la clausula CASCADE o RESTRICT

**Catalogo del Sistema – Diccionario de Datos**

Es una colección de tablas especiales en una base de datos que son propiedad, están creadas y son mantenidas por el propio DBMS. Estas tablas del sistema contienen datos que describen la estructura de la base de datos. Las tablas del catalogo de sistema se crean automáticamente al crear la base de datos. Generalmente se regoen todas juntas bajo un id usuario de sistema especial con un nombre como SYSTEM, o SYSIBM, etc.

El DBMS se refiere constantemente a los datos del catalogo de sistema cuando procesa las sentencias SQL.

En un SELECT:

* Verifica que las tablas designadas existen.
* Asegura que el usuario tiene permisos
* Comprueba si existen las columnas.
* Resuelve los nombres de las columnas
* Determina el tipo de dato de cada columna

Unos de los beneficios mas importantes del catalogo de sistema es que hace posible herramientas de consulta de fácil actualización. Permite un acceso simple y transparente a la base de datos sin tener que aprender el lenguaje SQL.  
El estándar SQL2 proporciona una especificación de un conjunto de vistas que dan un acceso estandarizado a la información que normalmente se encuentra en el catalogo del sistema. Un DBMS que cumpla el estándar SQL debe soportar estas vistas, que en su conjunto se denominan INFORMATION\_SCHEMA.

**Contenido del Catalogo**

* Tablas. El catalogo describe cada tabla de la base de datos.
* Columnas. Describe cada columna de la base de datos
* Usuarios. Describe cada usuario autorizado en la base de datos.
* Vistas. Describe cada vista definida en la base de datos.
* Privilegios. Describe cada grupo de privilegios concedidos.

**El esquema de información de SQL2**

El estándar SQL2 no especifica directamente un catalogo de sistema que deba ser soportado por las implementaciones de los DBMS. Se definió un catalogo de sistema idealizado que los fabricantes podrían diseñar partiendo de cero. Estas tablas de un catalogo idealizado se denomina esquema de definición en el estándar. El estándar no exige que un DBMS realmente soporte las tablas del catalogo de sistema o cualquier otro catalogo de sistema. Sin embargo, define una serie de vistas sobre las tablas del catalogo que identifican los objetos de la base de datos que están accesibles al usuario actual. Cualquier DBMS que cumpla con el estándar debe soportar estas vistas.