**Unidad 1: Administración de Base de Datos**

**LIBRO 1**

**Capítulo 1**

**¿Qué es un sistema de base de datos?**

Un sistema de BD es básicamente un sistema computarizado para llevar registros. Es un depósito o contenedor de una colección de archivos de datos computarizados. Los usuarios del sistema pueden realizar una variedad de operaciones sobre dichos archivos. La finalidad general de una BD es almacenar información y permitir a los usuarios recuperar y actualizar esa información con base en peticiones.

**¿Qué es una base de datos?**

Una base de datos es un conjunto de datos persistentes que es utilizado por los sistemas de aplicación de alguna empresa dada. Decimos que los datos "persisten" debido a que, una vez aceptados por el DBMS para entrar en la base de datos, sólo pueden ser removidos de la BD por alguna solicitud explícita al DBMS, no como un mero efecto lateral de algún programa que termina su ejecución.

Datos y modelos de datos

Los datos en realidad son hechos dados, a partir de los cuales es posible inferir hechos adicionales. Inferir hechos adicionales a partir de hechos dados es exactamente lo que hace el DBMS cuando responde a una consulta de un usuario.

Un *modelo de datos* es una definición lógica, independiente y abstracta de los objetos, operadores y demás que en conjunto constituyen la máquina abstracta con la que interactúan los usuarios. Los objetos nos permiten modelar la estructura de los datos. Los operadores nos permiten modelar su comportamiento.

La *implementación* de determinado modelo de datos es una realización física, en una máquina real, de los componentes de la máquina abstracta que en conjunto constituyen ese modelo.

Por lo tanto, el modelo es aquello que los usuarios tienen que conocer y la implementación es lo que los usuarios no tienen que conocer.

**¿Por qué una base de datos?**

Ventajas de un sistema de BD sobre los métodos tradicionales basados en papel para llevar un registro.

* *Compactación*: No hay necesidad de archivos en papel voluminosos.
* *Velocidad*: La máquina puede recuperar y actualizar datos más rápidamente que un humano.
* *Menos trabajo laborioso*; Se puede eliminar gran parte del trabajo de llevar los archivos a mano. Las tareas mecánicas siempre las realizan mejor las máquinas.
* *Actualidad*: En el momento que la necesitemos, tendremos a nuestra disposición información precisa y actualizada.

En el entorno multiusuario hay una ventaja adicional: El sistema de base de datos ofrece a la empresa un control centralizado de sus datos.

Administración de datos y administración de bases de datos

En la empresa habrá alguna persona identificable que tendrá responsabilidad central sobre los datos. Esa persona es el administrador de datos (DA), el cual se encarga de decidir en primer lugar qué datos deben ser almacenados en la base de datos y establecer políticas para mantener y manejar esos datos una vez almacenados (política de seguridad). El administrador de datos no es un técnico. El técnico responsable de implementar las decisiones del administrador de datos es el administrador de base de datos (DBA). Su trabajo consiste en crear la base de datos real e implementar los controles técnicos necesarios para hacer cumplir las diversas decisiones de las políticas hechas por el administrador de datos. El DBA también es responsable de asegurar que el sistema opere con el rendimiento adecuado y de proporcionar una variedad de otros servicios técnicos. Por lo regular, el DBA tendrá un equipo de programadores de sistemas y otros asistentes técnicos.

Beneficios del enfoque de base de datos

Ventajas del control centralizado.

* *Los datos pueden compartirse*: significa que las aplicaciones existentes pueden compartir la información de la base de datos y también permite desarrollar nuevas aplicaciones para operar sobre los mismos datos. En otras palabras, es posible satisfacer los requerimientos de datos de aplicaciones nuevas sin tener que agregar información a la base de datos.
* *Es posible reducir la redundancia*: En sistemas que no son de bases de datos, cada aplicación tiene sus propios archivos exclusivos. A menudo este hecho puede conducir a una redundancia considerable de los datos almacenados, con el consecuente desperdicio de espacio de almacenamiento. Sin embargo, estos dos archivos pueden integrarse y eliminar así la redundancia, en tanto el administrador de datos esté consciente de los requerimientos de datos de ambas aplicaciones.
* *Es posible evitar la inconsistencia*: Una base de datos en un estado inconsistente es capaz de proporcionar a sus usuarios información incorrecta o contradictoria.

Si no se elimina la redundancia pero se controla, entonces el DBMS puede garantizar que la BD nunca será inconsistente para el usuario, asegurando que todo cambio realizado a cualquiera de las entidades será aplicado en forma correcta.

* *Es posible brindar un manejo de transacciones*: Una transacción es una unidad de trabajo lógica, que por lo regular comprende varias operaciones de la base de datos. Por ejemplo, si el usuario declara que dos actualizaciones son parte de la misma transacción, entonces el sistema puede en efecto garantizar que se hagan ya sea ambas o ninguna de ellas, aun cuando el sistema fallara a la mitad del proceso.
* *Es posible mantener la integridad*: asegurar que los datos de la base de datos estén correctos. Si hay redundacia de datos, la inconsistencia entre dos entradas que pretenden representar el mismo "hecho" es un ejemplo de la falta de integridad. No obstante, aun cuando no exista redundancia, la base de datos podría seguir conteniendo información incorrecta. El control centralizado de la base de datos puede ayudar a evitar estos problemas (en la medida de lo posible) permitiendo que el administrador de datos defina y el DBA implemente las restricciones de integridad que serán verificadas siempre que se realice una operación de actualización.
* *Es posible hacer cumplir la seguridad*: Al tener la completa jurisdicción sobre la base de datos, el DBA puede asegurar que el único medio de acceso a la base de datos sea a través de los canales adecuados y por lo tanto puede definir las reglas o restricciones de seguridad que serán verificadas siempre que se intente acceder a datos sensibles.
* *Es posible equilibrar los requerimientos en conflicto*: Al conocer los requerimientos generales de la empresa, el DBA puede estructurar los sistemas de manera que ofrezcan un mejor servicio general. Por ejemplo, es posible elegir una representación física de los datos almacenados que proporcione un acceso rápido para las aplicaciones más importantes.
* *Es posible hacer cumplir los estándares*: Con el control central de la base de datos, el DBA puede asegurar que todos los estándares aplicables en la representación de los datos sean observados. Es conveniente estandarizar la representación de datos, la asignación de nombres y la documentación de los datos como una ayuda para compartir y entender los datos.
* *Es posible dar independencia a los datos*: es un objetivo de los sistemas de bases de datos. Es la inmunidad o robustez que tienen las aplicaciones para soportar cambios en los datos.

**Capítulo 2 - Arquitectura de los sistemas de bases de datos**

**Los tres niveles de la arquitectura**

La arquitectura ANSI/SPARC se divide en tres niveles, conocidos como interno, conceptual y externo, respectivamente.

* El nivel interno (físico) es el que está más cerca del almacenamiento físico; es decir, es el que tiene que ver con la forma en que los datos están almacenados físicamente.
* El nivel externo (lógico de usuario individual) es el más próximo a los usuarios; es decir, el que tiene que ver con la forma en que los usuarios individuales ven los datos.
* El nivel conceptual (lógico de la comunidad de usuarios) es un nivel de indirección entre los otros dos.

En otras palabras, habrá muchas vistas externas distintas, cada una consistente en una representación más o menos abstracta de alguna parte de la BD total, y habrá precisamente una vista conceptual que del mismo modo consiste en una representación abstracta de la BD en su totalidad. En forma similar, habrá precisamente una vista interna que represente a la BD tan como está almacenada físicamente.

En un sistema relacional, el nivel conceptual y externo serán relacionales ya que los datos se representarán mediante tablas, pero el nivel interno no porque los datos estarán representados mediante registros físicos del sistema.

**El nivel externo**

El nivel externo es el nivel del usuario individual. Cada usuario tiene a su disposición un lenguaje:

* Para el programador de aplicaciones, éste será ya sea un lenguaje de programación convencional o bien un lenguaje de tipo propietario que sea específico al sistema en cuestión.
* Para el usuario final, el lenguaje será ya sea un lenguaje de consulta o bien algún lenguaje de finalidad específica, tal vez controlado por formularios o por menús, confeccionado para los requerimientos de ese usuario y manejado por algún programa de aplicación en línea.

Lo importante acerca de dichos lenguajes es que incluirán un sublenguaje de datos; es decir, un subconjunto del lenguaje total que se ocupe específicamente de los objetos y operaciones de la base de datos. Se dice que el sublenguaje de datos está incrustado dentro de su lenguaje anfitrión correspondiente.

El lenguaje anfitrión es el responsable de proporcionar diversas propiedades que no son específicas de la base de datos, como las variables locales, las operaciones de cálculo, la lógica de bifurcación, etcétera. Un sublenguaje de datos específico soportado por casi todos los sistemas actuales es el lenguaje SQL. La mayoría de dichos sistemas permiten que SQL sea utilizado de manera interactiva, como un lenguaje de consulta independiente, e incrustado en otros lenguajes.

Cualquier sublenguaje de datos determinado es en realidad una combinación de por lo menos dos lenguajes subordinados: un DDL (Lenguaje de Definición de Datos), que permite la definición o declaración de objetos de una BD, y un DML (Lenguaje de Manipulación de Datos), que permite la manipulación o procesamiento de dichos objetos.

Un usuario individual se interesará generalmente sólo por alguna parte de toda la base de datos, lo que en ANSI/SPARC es llamado vista externa. Por lo tanto, una vista externa es el contenido de una base de datos que verá un usuario en particular.

**El nivel conceptual**

Es una representación de todo el contenido de la información de la BD. La vista conceptual pretende ser una vista de los datos "tal como son", en vez de tal como los usuarios están obligados a verlos debido a las limitaciones, por ejemplo, del lenguaje o el hardware en particular que pudieran utilizar.

La vista conceptual es una vista del contenido total de la base de datos, y el esquema conceptual es una definición de esa vista. Las definiciones del esquema conceptual pretenden incluir muchas características adicionales, como las restricciones de seguridad y de integridad. En la mayoría de los sistemas existentes, el esquema conceptual es en realidad algo más que una simple unión de todos los esquemas externos individuales más ciertas restricciones de seguridad y de integridad.

El esquema conceptual está escrito con otro lenguaje de definición de datos, el DDL conceptual. Si se va a lograr la independencia física de los datos, entonces las definiciones conceptuales de DDL no deben comprender en lo absoluto ninguna consideración de la representación física ni de la técnica de acceso; deben ser únicamente definiciones del contenido de la información. Si el esquema conceptual se hace verdaderamente independiente de los datos, entonces los esquemas externos, que están definidos en términos del esquema conceptual, también serán forzosamente independientes de los datos.

**El nivel Interno**

La vista interna es una representación de bajo nivel de toda la base de datos y consiste en muchas ocurrencias de cada uno de los diversos tipos de registros internos. Por lo tanto, la vista interna está todavía distante del nivel físico, ya que no tiene que ver con términos como registros físicos ni con ninguna consideración específica de los dispositivos, Sólo da por hecho un espacio de direcciones lineal infinito.

La vista interna se describe por medio del esquema interno, el cual no sólo define los diversos tipos de registros almacenados sino que especifica también qué índices existen, cómo están representados los campos almacenados, en qué secuencia están dichos registros, etcétera. El esquema interno está escrito utilizando otro lenguaje más de definición de datos: el DDL interno.

En ciertas situaciones excepcionales, a los programas de aplicación se les podría permitir operar directamente en el nivel interno en vez del nivel externo, lo que no es recomendable porque el programa sería dependiente de los datos y representa un riesgo para la seguridad y la integridad.

**El administrador de base de datos**

El DA es la persona que toma las decisiones de estrategia y política con respecto a los datos de la empresa y el DBA es la persona que proporciona el apoyo técnico necesario para implementar dichas decisiones. Por lo tanto, el DBA es el responsable del control general del sistema al nivel técnico.

Tareas del DBA:

* Definir el esquema conceptual: Es trabajo del administrador de datos decidir exactamente qué información contendrá la base de datos. Por lo regular a este proceso se le conoce como diseño lógico o conceptual de la base de datos. Una vez que el administrador decidió el contenido de la base de datos a un nivel abstracto, entonces el DBA creará el esquema conceptual correspondiente, utilizando el DLL conceptual, el cual utilizará el DBMS para responder a las peticiones de acceso.
* Definir el esquema interno: El DBA también debe decidir la forma en que van a ser representados los datos en la base de datos almacenada. A este proceso se le conoce comúnmente como diseño físico de la base de datos. Una vez realizado el diseño físico, el DBA deberá crear la definición de la estructura de almacenamiento correspondiente (es decir, el esquema interno), utilizando el DDL interno. Además, también deberá definir la transformación conceptual/interna asociada.
* Establecer un enlace con los usuarios: Es asunto del DBA enlazarse con los usuarios para asegurar que los datos necesarios estén disponibles y para escribir (o ayudar a escribir) los esquemas externos necesarios, utilizando el DDL externo aplicable. Además, también es necesario definir las transformaciones externas/conceptual correspondientes. Otros aspectos de la función de enlace con los usuarios incluyen la asesoría sobre el diseño de aplicaciones; una capacitación técnica; ayuda en la determinación y resolución de problemas; así como otros servicios profesionales similares.
* Definir las restricciones de seguridad y de integridad: las restricciones de seguridad y de integridad pueden ser vistas como parte del esquema conceptual. El DDL conceptual debe incluir facilidades para especificar dichas restricciones.
* Definir las políticas de vaciado y recarga: En el caso de que se produzca un daño en cualquier parte de la base de datos, resulta esencial poder reparar los datos afectados con el mínimo de demora y con tan poco efecto como sea posible sobre el resto del sistema. El DBA debe definir e implementar un esquema apropiado de control de daños que comprenda la descarga o vaciado periódico de la base de datos en un dispositivo de almacenamiento de respaldo y la recarga de la base de datos cuando sea necesario, a partir del vaciado más reciente.
* Supervisar el rendimiento y responder a los requerimientos cambiantes: El DBA es el responsable de organizar el sistema de tal manera que se obtenga el rendimiento ideal para la empresa y de hacer los ajustes apropiados conforme las necesidades cambien. Todo cambio al nivel interno de almacenamiento físico debe estar acompañado por el cambio correspondiente en la definición de la transformación conceptual/interna, de manera que el esquema conceptual permanezca constante.

**El sistema de administración de base de datos**

El DBMS es el software que maneja todo acceso a la base de datos. De manera conceptual, lo que sucede es lo siguiente:

1. Un usuario emite una petición de acceso, utilizando algún sublenguaje de datos específico (por lo regular SQL).

2. El DBMS intercepta esa petición y la analiza.

3. El DBMS inspecciona, en su momento, el esquema externo para ese usuario, la transformación externa/conceptual correspondiente, el esquema conceptual, la transformación conceptual/interna y la definición de la estructura de almacenamiento.

4. El DBMS ejecuta las operaciones necesarias sobre la base de datos almacenada.

Funciones del DBMS:

* Definición de datos: El DBMS debe ser capaz de aceptar definiciones de datos (esquemas y transformaciones) en la forma fuente y convertirlas a la forma objeto correspondiente. En otras palabras, el DBMS debe incluir entre sus componentes un procesador DDL, o compilador DDL, para cada uno de los diversos DDLs. El DBMS también debe entender definiciones DDL para analizar y responder a las peticiones de manipulación de datos.
* Manipulación de datos: El DBMS debe ser capaz de manejar peticiones para recuperar, actualizar o eliminar datos existentes en la base de datos o agregar nuevos datos a ésta. En otras palabras, el DBMS debe incluir un componente procesador DML o compilador DML para tratar con el DML.Las peticiones DML deben ser procesadas por el componente optimizador, cuya finalidad es determinar una forma eficiente de implementar la petición.
* Seguridad e integridad de los datos: El DBMS debe vigilar las peticiones del usuario y rechazar todo intento de violar las restricciones de seguridad y de integridad definidas por el DBA.
* Recuperación de datos y concurrencia: El DBMS o un monitor de procesamiento de transacciones (monitor PT), debe imponer ciertos controles de recuperación y concurrencia.
* Diccionario de datos: El DBMS debe proporcionar una función de diccionario de datos. El diccionario contiene "datos acerca de los datos" (metadatos); es decir, definiciones de otros objetos del sistema. En particular, todos los diversos esquemas y transformaciones y todas las diversas restricciones de seguridad y de integridad, serán almacenadas en el diccionario. Debe ser posible consultar el diccionario del mismo modo que cualquier otra base de datos, de manera que, por ejemplo, sea posible saber qué programas o usuarios se podrían ver afectados por un cambio propuesto al sistema.
* Rendimiento: El DBMS debe realizar todas las tareas antes identificadas de la manera más eficiente posible.

**Arquitectura cliente-servidor**

Un sistema de base de datos puede ser visto como un sistema que tiene una estructura de dos partes, las cuales consisten en un servidor y un conjunto de clientes.

El servidor es precisamente el propio DBMS. Soporta todas las funciones básicas del DBMS: definición de datos, manipulación de datos, seguridad e integridad de los datos, etcétera. Proporciona todo el soporte de los niveles.

Los clientes son las diversas aplicaciones que se ejecutan sobre el DBMS, tanto aplicaciones escritas por el usuario como aplicaciones integradas.

**El procesamiento distribuido**

Procesamiento distribuido significa que distintas máquinas pueden conectarse en una red de comunicaciones de tal manera que una sola tarea de procesamiento de datos pueda extenderse a varias máquinas de la red. La comunicación entre las diversas máquinas es manejada mediante algún tipo de software de administración de redes.

Un cliente y un servidor pueden ejecutarse en máquinas diferentes. Ventajas:

* Procesamiento paralelo normal: Se aplican muchas unidades de procesamiento para las tareas en conjunto, mientras que el procesamiento del servidor (base de datos) y del cliente (aplicación) se están haciendo en paralelo. De ahí que el tiempo de respuesta y la velocidad real de transporte tengan mejorías.
* La máquina servidor podría ser una máquina construida a la medida y adaptada a la función del DBMS y podría proporcionar un mejor desempeño del DBMS.
* La máquina cliente podría ser una estación de trabajo personal adaptada a las necesidades del usuario final y por lo tanto, capaz de proporcionar mejores interfaces, una alta disponibilidad, respuestas más rápidas y en general una mejor facilidad de uso para el usuario.
* Varias máquinas cliente distintas podrían ser capaces de acceder a la misma máquina servidor. Por lo tanto, una sola base de datos podría ser compartida entre varios sistemas cliente distintos.

Una sola máquina cliente podría ser capaz de acceder a varias máquinas servidor diferentes de dos formas posibles:

* Una máquina cliente dada podría ser capaz de acceder a cualquier cantidad de servidores, pero sólo uno a la vez. Es decir, cada petición individual de base de datos debe ser dirigida a un solo servidor. En un sistema así, no es posible combinar datos de dos o más servidores con una sola petición. Además, el usuario de dicho sistema debe saber qué datos contiene cada servidor en particular.
* El cliente podría ser capaz de acceder a varios servidores en forma simultánea. Es decir, una sola petición de base de datos podría combinar datos de varios servidores. En este caso, los servidores ven al cliente, desde un punto de vista lógico, como si en realidad fuera un solo servidor y el usuario no tiene que saber en qué servidor están los datos.

El soporte total a una base de datos distribuida implica que una sola aplicación debe ser capaz de operar "de manera transparente" sobre los datos que están dispersos a través de una variedad de bases de datos diferentes, las cuales son administradas por una variedad de DBMSs distintos, operan en varias máquinas distintas, son manejadas por varios sistemas operativos diferentes y están conectadas por una variedad de redes de comunicación distintas; aquí, "de manera transparente" significa que la aplicación opera desde un punto de vista lógico como si los datos fueran manejados por un solo DBMS y en una sola máquina.

**LIBRO 2**

**Visión de los Datos**

Un sistema de bases de datos es una colección de archivos interrelacionados y un conjunto de programas que permitan a los usuarios acceder y modificar estos archivos. Uno de los propósitos principales es proporcionar a los usuarios una visión abstracta de los datos.

Abstracción de datos

Para que el sistema sea útil debe recuperar los datos eficientemente. Los desarrolladores esconden la complejidad a los usuarios a través de varios niveles de abstracción para simplificar la interacción de los usuarios con el sistema:

* Nivel físico: el nivel más bajo de abstracción describe cómo se almacenan realmente los datos. Se describen en detalle las estructuras de datos complejas de bajo nivel.
* Nivel lógico: describe qué datos se almacenan en la base de datos y qué relaciones existen entre esos datos. La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples.
* Nivel de vistas: nivel más alto de abstracción, describe sólo parte de la base de datos completa. En el nivel lógico, queda algo de complejidad, debido a la variedad de información almacenada en una gran base de datos. Muchos usuarios no necesitan toda esta información, sino que necesitan acceder sólo a una parte de la base de datos. Para que su interacción con el sistema se simplifique, se define la abstracción del nivel de vistas. El sistema puede proporcionar muchas vistas para la misma base de datos.

Además de esconder detalles del nivel lógico de la base de datos, las vistas también proporcionan un mecanismo de seguridad para evitar que los usuarios accedan a ciertas partes de la base de datos.

Ejemplares y esquemas

Las bases de datos van cambiando a lo largo del tiempo conforme la información se inserta y borra. La colección de información almacenada en la base de datos en un momento particular se denomina un *ejemplar* de la base de datos. El diseño completo de la base de datos se llama el *esquema* de la base de datos. Los esquemas son raramente modificados, si es que lo son alguna vez.

**Modelos de los datos**

Bajo la estructura de la base de datos se encuentra el modelo de datos: una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Para ilustrar el concepto de un modelo de datos, describimos dos modelos de datos en este apartado: el modelo entidad-relación y el modelo relacional.

Modelo entidad-relación

El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos. Una entidad es una «cosa» u «objeto» en el mundo real que es distinguible de otros objetos. Las entidades se describen en una base de datos mediante un conjunto de atributos.

Una relación es una asociación entre varias entidades. El conjunto de todas las entidades del mismo tipo, y el conjunto de todas las relaciones del mismo tipo, se denominan respectivamente **conjunto de entidades** y **conjunto de relaciones**.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama ER,

que consta de los siguientes componentes:

* **Rectángulos**, que representan conjuntos de entidades.
* **Elipses**, que representan atributos.
* **Rombos**, que representan relaciones entre conjuntos de entidades.
* **Líneas**, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

El modelo E-R representa ciertas restricciones que los contenidos de la base de datos deben cumplir. Una restricción importante es la correspondencia de cardinalidades, que expresa el número de entidades con las que otra entidad se puede asociar a través de un conjunto de relaciones.

Modelo relacional

En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único.

El modelo relacional es un ejemplo de un modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo particular. Cada tipo de registro define un número fijo de campos, o atributos. Las columnas de la tabla corresponden a los atributos del tipo de registro.

El modelo de datos relacional es el modelo de datos más ampliamente usado, y una amplia mayoría de sistemas de bases de datos actuales se basan en el modelo relacional.

El modelo relacional se encuentra a un nivel de abstracción inferior al modelo de datos E-R. Los diseños de bases de datos a menudo se realizan en el modelo E-R, y después se traducen al modelo relacional.

Otros modelos de datos

El modelo de datos orientado a objetos es otro modelo de datos que está recibiendo una atención creciente. Se puede observar como una extensión del modelo E-R con las nociones de encapsulación, métodos (funciones) e identidad de objeto.

El modelo de datos relacional orientado a objetos combina las características del modelo de datos orientado a objetos y el modelo de datos relacional.

**Lenguajes de Bases de Datos**

Un sistema de bases de datos proporciona un **lenguaje de definición de datos (DDL)** para especificar el esquema de la base de datos y un **lenguaje de manipulación de datos (DML)** para expresar las consultas a la base de datos y las modificaciones. Forman partes de un único lenguaje de bases de datos, tal como SQL.

Lenguaje de definición de datos

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado lenguaje de definición de datos.

Lenguaje de manipulación de datos

La manipulación de datos es:

* La recuperación de información almacenada en la base de datos.
* La inserción de información nueva en la base de datos.
* El borrado de información de la base de datos.
* La modificación de información almacenada en la base de datos.

Permite a los usuarios acceder o manipular los datos organizados mediante el modelo de datos apropiado. Dos tipos:

* **DMLs procedimentales**. Requieren que el usuario especifique qué datos se necesitan y cómo obtener esos datos.
* **DMLs declarativos** (DMLs no procedimentales). Requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtener esos datos.

Una consulta es una instrucción de solicitud para recuperar información.

**Usuarios y administradores de la Base de Datos**

Usuarios de bases de datos e interfaces de usuario

Cuatro tipos diferentes de usuarios de un sistema de base de datos:

* **Usuarios normales**. Son usuarios no sofisticados que interactúan con el sistema mediante la invocación de alguno de los programas de aplicación permanentes que se ha escrito previamente. La interfaz de usuario normal para los usuarios normales es una interfaz de formularios, donde el usuario puede rellenar los campos apropiados del formulario. Los usuarios normales pueden también simplemente leer informes generados de la base de datos.
* **Programadores de aplicaciones**. Son profesionales informáticos que escriben programas de aplicación.
* **Usuarios sofisticados**. Interactúan con el sistema sin programas escritos. En su lugar, ellos forman sus consultas en un lenguaje de consulta de bases de datos. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría.
* **Usuarios especializados**. Son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas que no son adecuadas en el marco de procesamiento de datos tradicional.

Administrador de la base de datos

Una de las principales razones de usar SGBDs es tener un control centralizado tanto de los datos como de los programas que acceden a esos datos. La persona que tiene este control central sobre el sistema se llama administrador de la base de datos (ABD). Las funciones del ABD incluyen las siguientes:

* **Definición del esquema**. El ABD crea el esquema original de la base de datos escribiendo un conjunto de instrucciones de definición de datos en el LDD.
* **Definición de la estructura y del método de acceso**.
* **Modificación del esquema y de la organización física**. Los ABD realizan cambios en el esquema y en la organización física para reflejar las necesidades cambiantes de la organización, o para alterar la organización física para mejorar el rendimiento.
* **Concesión de autorización para el acceso a los datos**. La concesión de diferentes tipos de autorización permite al administrador de la base de datos determinar a qué partes de la base de datos puede acceder cada usuario.
* **Mantenimiento rutinario**.

**Estructura de un Sistema de Bases de Datos**

Un sistema de bases de datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. Los componentes funcionales de un sistema de bases de datos se pueden dividir a grandes rasgos en los componentes *gestor de almacenamiento* y *procesador de consultas*.

El gestor de almacenamiento es importante porque las bases de datos requieren normalmente una gran cantidad de espacio de almacenamiento.

El procesador de consultas es importante porque ayuda al sistema de bases de datos a simplificar y facilitar el acceso a los datos.

Gestor de almacenamiento

Un gestor de almacenamiento es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel en la base de datos y los programas de aplicación y consultas emitidas al sistema. El gestor de almacenamiento es responsable del almacenamiento, recuperación y actualización de los datos en la base de datos.

Componentes del gestor de almacenamiento:

* **Gestor de autorización e integridad**
* **Gestor de transacciones**
* **Gestor de archivos**
* **Gestor de memoria compartida**

El gestor de almacenamiento implementa varias estructuras de datos como parte de la implementación física del sistema:

* **Archivos de datos**.
* **Diccionario de datos**.
* **Índices**.

Procesador de consultas

Componentes del procesador de consultas:

* **Intérprete del LDD**
* **Compilador del LMD**
* **Motor de evaluación de consultas**

**Capítulo 2**

**Modelo Entidad-Relación**

El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real consistente en objetos básicos llamados entidades y de relaciones entre estos objetos.

**Conceptos Básicos**

Hay tres nociones básicas que emplea el modelo de datos E-R: conjuntos de entidades, conjuntos de relaciones y atributos.

Conjuntos de entidades

Una entidad es una «cosa» u «objeto» en el mundo real que es distinguible de todos los demás objetos.

Un conjunto de entidades es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos.

Una entidad se representa mediante un conjunto de atributos. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. Para cada atributo hay un conjunto de valores permitidos, llamados el dominio, o el conjunto de valores, de ese atributo.

Conjuntos de relaciones

Una relación es una asociación entre diferentes entidades. Un conjunto de relaciones es un conjunto de relaciones del mismo tipo.

La asociación entre conjuntos de entidades se conoce como participación.

**Restricciones**

Un esquema de desarrollo E-R puede definir ciertas restricciones a las que los contenidos de la BD se deben adaptar.

Correspondencia de cardinalidades

Expresa el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones.

Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de cardinalidades debe ser una de las siguientes:

* **Uno a uno.** Una entidad en A se asocia con a lo sumo una entidad en B, y una entidad en B se asocia con a lo sumo una entidad en A.
* **Uno a varios.** Una entidad en A se asocia con cualquier número de entidades en B (ninguna o varias). Una entidad en B, sin embargo, se puede asociar con a lo sumo una entidad en A.
* **Varios a uno.** Una entidad en A se asocia con a lo sumo una entidad en B. Una entidad en B, sin embargo, se puede asociar con cualquier número de entidades en A.
* **Varios a varios.** Una entidad en A se asocia con cualquier número de entidades en B y una entidad en B se asocia con cualquier número de entidades en A.

Restricciones de participación

La participación de un conjunto de entidades E en un conjunto de relaciones R se dice que es **total** si cada entidad en E participa al menos en una relación en R. Si sólo algunas entidades en E participan en relaciones en R, la participación del conjunto de entidades E en la relación R se llama **parcial**.

**Claves**

Una clave permite identificar un conjunto de atributos suficiente para distinguir las entidades entre sí. Las claves también ayudan a identificar unívocamente a las relaciones y así a distinguir las relaciones entre sí.

Conjuntos de entidades

Una **superclave** es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en el conjunto de entidades. A menudo interesan las superclaves tales que los subconjuntos propios de ellas no son superclave. Tales superclaves mínimas se llaman **claves candidatas**.

Se usa el término **clave primaria** para denotar una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades.

**Diagrama Entidad-Relación**

Componentes principales:

* **Rectángulos**, que representan conjuntos de entidades.
* **Elipses**, que representan atributos.
* **Rombos**, que representan relaciones.
* **Líneas**, que unen atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.
* **Elipses dobles**, que representan atributos multivalorados (atributos que pueden tener más de un valor).
* **Elipses discontinuas**, que denotan atributos derivados.
* **Líneas dobles**, que indican participación total de una entidad en un conjunto de relaciones.
* **Rectángulos dobles**, que representan conjuntos de entidades débiles.

**Características del Modelo E-R Extendido**

Especialización

Un conjunto de entidades puede incluir subgrupos de entidades que se diferencian de alguna forma de las otras entidades del conjunto. Por ejemplo, un subconjunto de entidades en un conjunto de entidades puede tener atributos que no son compartidos por todas las entidades del conjunto de entidades.

El proceso de designación de subgrupos dentro de un conjunto de entidades se denomina **especialización**.

Generalización

Es una relación contenedora que existe entre el conjunto de entidades de nivel más alto y uno o más conjuntos de entidades de nivel más bajo. Los conjuntos de entidades de nivel más alto y nivel más bajo también se pueden llamar superclase y subclase, respectivamente.

Herencia de Atributos

Una propiedad crucial de las entidades de nivel más alto y más bajo creadas mediante especialización y generalización es la herencia de atributos. Los atributos de los conjuntos de entidades de nivel más alto se dice que son **heredados** por los conjuntos de entidades de nivel más bajo.

Agregación

Es una abstracción a través de la cual las relaciones se tratan como entidades de nivel más alto.

**Reducción de un Esquema E-R a Tablas**

Una base de datos que se ajusta a un esquema de bases de datos E-R se puede representar por una colección de tablas.

Representación tabular de los conjuntos de entidades fuertes

Sea E un conjunto de entidades fuertes con los atributos descriptivos . Esta entidad se representa mediante una tabla llamada E con columnas distintas, cada una de las cuales corresponde a uno de los atributos de E. Cada fila de la tabla corresponde a una entidad del conjunto de entidades E.

Representación tabular de los conjuntos de entidades débiles

Sea A un conjunto de entidades débiles con los atributos . Sea B el conjunto de entidades fuertes del que A depende. Sea la clave primaria de B el conjunto de atributos . Se representa el conjunto de entidades A mediante una tabla llamada A con una columna por cada uno de los atributos del conjunto: .

Representación tabular de los conjuntos de relaciones

Sea R un conjunto de relaciones, sean el conjunto de atributos formados por la unión de las claves primarias de cada uno de los conjuntos de entidades que participan en R, y sean los atributos descriptivos de R (si los hay). El conjunto de relaciones se representa mediante una tabla llamada R con una columna por cada uno de los atributos del conjunto: .

Combinación de tablas

Considérese un conjunto AB de relaciones varios a uno del conjunto de entidades A al conjunto de entidades B. Usando el esquema de construcción de tablas descrito previamente se consiguen tres tablas: A, B y AB. Supóngase además que la participación de A en la relación es total; es decir, cada entidad a en el conjunto de entidades A debe participar en la relación AB. Entonces se pueden combinar las tablas A y AB para formar una única tabla consistente en la unión de las columnas de ambas tablas.

En el caso de relaciones uno a uno, la tabla del conjunto de relaciones se puede combinar con las tablas de cualquiera de los conjuntos de entidades. Las tablas se pueden combinar incluso si la participación es parcial usando valores nulos.

Atributos compuestos

Los atributos compuestos se manejan creando un atributo separado para cada uno de los atributos componentes; no se crea una columna separada para el propio atributo compuesto.

Atributos multivalorados

Para un atributo multivalorado M se crea una tabla T con una columna C que corresponde a la clave primaria del conjunto de entidades o conjunto de relaciones del que M es atributo.

**Capítulo 18 - Arquitectura de los sistemas de BD**

La arquitectura de un sistema de bases de datos está influenciada en gran medida por el sistema informático subyacente en el que se ejecuta, en particular por aspectos de la arquitectura de la computadora como la conexión en red, el paralelismo y la distribución.

La conexión en red de varias computadoras permite que algunas tareas se ejecuten en un sistema servidor y otras en sistemas clientes. Esta división de trabajo ha conducido al desarrollo de sistemas de bases de datos cliente-servidor.

El procesamiento paralelo dentro de una computadora permite acelerar las actividades del sistema de base de datos, proporcionando a las transacciones unas respuestas más rápidas, así como la capacidad de ejecutar más transacciones por segundo. La necesidad del procesamiento paralelo ha conducido al desarrollo de sistemas de bases de datos paralelos.

La distribución de datos a través de las distintas sedes o departamentos de una organización permite que estos datos residan donde han sido generados o donde son más necesarios, pero continuar siendo accesibles desde otros lugares o departamentos diferentes. El hecho de guardar varias copias de la base de datos en diferentes sitios permite que puedan continuar las operaciones sobre la base de datos ante cualquier fallo. Se han desarrollado los sistemas distribuidos de bases de datos para manejar datos distribuidos geográfica o administrativamente a lo largo de múltiples sistemas de bases de datos.

**Sistemas centralizados**

Los sistemas de bases de datos centralizados son aquellos que se ejecutan en un único sistema informático sin interaccionar con ninguna otra computadora.

Se distinguen dos formas de utilizar las computadoras: como sistemas monousuario o multiusuario. En la primera categoría están las computadoras personales y las estaciones de trabajo que son utilizadas por una persona y disponen de una CPU. Por el contrario, un sistema multiusuario típico tiene más discos y más memoria, puede disponer de varias CPU, trabaja con un sistema operativo multiusuario y se encarga de dar servicio a un gran número de usuarios que están conectados al sistema.

Normalmente, los sistemas de bases de datos diseñados para sistemas monousuario no suelen proporcionar muchas de las facilidades que ofrecen los sistemas multiusuario. No tienen control de concurrencia, las características de recuperación no existen o son primitivas, la mayoría no admiten SQL y proporcionan un lenguaje de consulta muy simple. En cambio, los sistemas de bases de datos para sistemas multiusuario soportan todas las características de las transacciones vistas anteriormente.

Aunque hoy en día las computadoras de propósito general tienen varios procesadores. Las bases de datos que se ejecutan en tales máquinas habitualmente no intentan dividir una consulta simple entre los distintos procesadores, sino que ejecuta cada consulta en un único procesador posibilitando la concurrencia de varias consultas. Así, estos sistemas permiten ejecutar un mayor número de transacciones por segundo, a pesar de que cada transacción individualmente no se ejecute más rápido.

Las bases de datos diseñadas para las máquinas monoprocesador ya disponen de multitarea permitiendo que varios procesos se ejecuten a la vez en el mismo procesador, usando tiempo compartido, mientras que de cara al usuario parece que los procesos se están ejecutando en paralelo.

**Sistemas cliente-servidor**

Los sistemas cliente-servidor tienen su funcionalidad dividida entre el sistema servidor y múltiples sistemas clientes.

Los terminales conectados a un sistema central han sido suplantados por computadoras personales. De igual forma, la interfaz de usuario está pasando a ser gestionada por las computadoras personales. Como consecuencia, los sistemas centralizados actúan hoy como sistemas servidores que satisfacen las peticiones generadas por los sistemas clientes.

La funcionalidad de una base de datos se puede dividir a grandes rasgos en dos partes: la parte visible al usuario y el sistema subyacente. El sistema subyacente gestiona el acceso a las estructuras, la evaluación y optimización de consultas, el control de concurrencia y la recuperación. La parte visible al usuario de un sistema de base de datos está formado por herramientas como formularios, diseñadores de informes y facilidades gráficas de interfaz de usuario.

La interfaz entre la parte visible al usuario y el sistema subyacente puede ser SQL o una aplicación. Con el aumento de las interfaces estándares, es posible que diferentes distribuidores proporcionan la interfaz visible al usuario y el servidor del sistema subyacente.

**Arquitectura de Sistemas Servidores**

Los sistemas servidores pueden dividirse en servidores de transacciones y servidores de datos.

Sistemas servidores de transacciones: proporcionan una interfaz a través de la cual los clientes pueden enviar peticiones para realizar una acción que el servidor ejecutará y cuyos resultados se devolverán al cliente. Normalmente, las máquinas cliente envían las transacciones a los sistemas servidores, donde éstas se ejecutan, y los resultados se devuelven a los clientes que son los encargados de visualizar los datos. Las peticiones se pueden especificar utilizando SQL o mediante la interfaz de una aplicación especializada. Esta arquitectura es la más ampliamente utilizada.

Un sistema servidor de transacciones típico consiste en múltiples procesos accediendo a los datos en una memoria compartida. Procesos:

• *Procesos servidor*: son procesos que reciben consultas del usuario, las ejecutan, y devuelven los resultados. Las consultas deben enviarse a los procesos servidor desde la interfaz de usuario, o desde un proceso de usuario que ejecuta SQL incorporado. Algunos sistemas de bases de datos pueden ejecutar concurrentemente múltiples consultas.

• *Proceso gestor de bloqueos*: este proceso implementa una función de gestión de bloqueos que incluye concesión de bloqueos, liberación de bloqueos y detección de interbloqueos.

• *Proceso escritor de bases de datos*: hay uno o más procesos que vuelcan al disco los bloques de memoria intermedia modificados de forma continua.

• *Proceso escritor del registro*: este proceso genera entradas del registro en el almacenamiento estable a partir de la memoria intermedia del registro. Los procesos servidor simplifican la adición de entradas a la memoria intermedia del registro en memoria compartida y, si es necesario forzar la escritura del registro, le piden al proceso escritor del registro que vuelque las entradas del registro.

• *Proceso punto de revisión*: realiza periódicamente puntos de revisión.

• *Proceso monitor de proceso*: este proceso observa otros procesos y, si cualquiera de ellos falla, realiza acciones de recuperación para el proceso y lo reinicia.

Sistemas servidores de datos: permiten a los clientes interaccionar con los servidores realizando peticiones de lectura o modificación de datos en unidades tales como archivos o páginas. Por ejemplo, los servidores de archivos proporcionan una interfaz de sistema de archivos a través de la cual los clientes pueden crear, modificar, leer y borrar archivos. Los servidores de datos de los sistemas de bases de datos ofrecen muchas más funcionalidades; soportan unidades de datos de menor tamaño que los archivos, como páginas, tuplas u objetos. Proporcionan facilidades de indexación de los datos y de transacción.

Los sistemas servidores de datos se utilizan en redes de área local en las que se alcanza una alta velocidad de conexión entre los clientes y el servidor, las máquinas clientes son comparables al servidor en cuanto a poder de procesamiento y se ejecutan tareas de cómputo intensivo. En este entorno tiene sentido enviar los datos a las máquinas clientes, realizar allí todo el procesamiento y después enviar los datos de vuelta al servidor.

Esta arquitectura se ha hecho particularmente popular en los sistemas de bases de datos orientadas a objetos.

En esta arquitectura, el coste en tiempo de comunicación entre el cliente y el servidor es alto comparado al de acceso a una memoria local.

**Sistemas paralelos**

Los sistemas paralelos mejoran la velocidad de procesamiento y de E/S mediante la utilización de CPU y discos en paralelo. Estos sistemas surgen debido a que las aplicaciones tienen que manejar bases de datos extremadamente grandes o deben procesar un número enorme de transacciones por segundo.

En el procesamiento paralelo se realizan muchas operaciones simultáneamente. Una máquina paralela de grano grueso consiste en un pequeño número de potentes procesadores; una máquina masivamente paralela o de grano fino utiliza miles de procesadores más pequeños y son capaces de soportar un grado de paralelismo mucho mayor.

Un sistema que procese un gran número de pequeñas transacciones puede mejorar la productividad realizando muchas transacciones en paralelo. Un sistema que procese transacciones largas puede mejorar el tiempo de respuesta y la productividad realizando en paralelo las distintas subtareas de cada transacción.

Ganancia de velocidad y ampliabilidad

La ganancia de velocidad se refiere a la ejecución en menos tiempo de una tarea dada mediante el incremento del grado de paralelismo. La ampliabilidad se refiere al manejo de transacciones más largas mediante el incremento del grado de paralelismo.

La manera de medir el tamaño de las tareas da lugar a dos tipos de ampliabilidad relevantes en los sistemas paralelos de bases de datos:

* En la ampliabilidad por lotes aumenta el tamaño de la base de datos, y las tareas son trabajos más largos cuyos tiempos de ejecución dependen del tamaño de la base de datos.
* En la ampliabilidad de transacciones aumenta la velocidad con la que se envían las transacciones a la base de datos y el tamaño de la base de datos crece proporcionalmente a la tasa de transacciones. Este tipo de ampliabilidad es el relevante en los sistemas de procesamiento de transacciones. Este procesamiento de transacciones se adapta especialmente bien a la ejecución en paralelo, ya que las transacciones pueden ejecutarse concurrente e independientemente en procesadores distintos y cada transacción dura más o menos el mismo tiempo, aunque crezca la base de datos.

La ampliabilidad es normalmente el factor más importante para medir la eficiencia de un sistema paralelo de bases de datos. El objetivo del paralelismo en los sistemas de bases de datos suele ser asegurar que la ejecución del sistema continuará realizándose a una velocidad aceptable incluso en el caso de que aumente el tamaño de la base de datos o el número de transacciones.

Existen algunos factores que trabajan en contra de la eficiencia del paralelismo:

• Costes de inicio: En una operación paralela compuesta por miles de procesos el tiempo de inicio puede llegar a ser mucho mayor que el tiempo real de procesamiento, lo que influye negativamente en la ganancia de velocidad.

• Interferencia: Como los procesos que se ejecutan en un sistema paralelo acceden con frecuencia a recursos compartidos, pueden sufrir un cierto retardo como consecuencia de la interferencia de cada nuevo proceso en la competencia con los procesos existentes por el acceso a los recursos más comunes.

• Sesgo: Al dividir cada tarea en un cierto número de pasos paralelos se reduce el tamaño del paso medio. Es más, el tiempo de servicio de la tarea completa vendrá determinado por el tiempo de servicio del paso más lento. Normalmente es difícil dividir una tarea en partes exactamente iguales, entonces se dice que la forma de distribución de los tamaños es sesgada.

Redes de interconexión

Los sistemas paralelos están constituidos por un conjunto de componentes (procesadores, memoria y discos) que pueden comunicarse entre sí a través de una red de interconexión. Los tres tipos de redes de interconexión utilizados frecuentemente son *bus* (todos los componentes del sistema pueden enviar o recibir datos de un único bus de comunicaciones), *malla* (los componentes se organizan como los nodos de una retícula de modo que cada componente está conectado con todos los nodos adyacentes) e *hipercubo* (se asigna a cada componente un número binario de modo que dos componentes tienen una conexión directa si sus correspondientes representaciones binarias difieren en un solo bit).

Arquitecturas paralelas de bases de datos

Existen varios modelos de arquitecturas para las máquinas paralelas, tales como:

* *Memoria compartida*: En una arquitectura de memoria compartida los procesadores y los discos tienen acceso a una memoria común, normalmente a través de un bus o de una red de interconexión. De esta forma, cualquier procesador puede acceder a los datos de la memoria compartida sin necesidad de la intervención del software. Un procesador puede enviar mensajes a otros procesadores con mucha velocidad.

Cada procesador suele tener una memoria caché muy grande para evitar las referencias a la memoria compartida siempre que sea posible.

* *Disco compartido*: En el modelo de disco compartido todos los procesadores pueden acceder directamente a todos los discos a través de una red de interconexión, pero los procesadores tienen memorias privadas.

Proporciona una cierta tolerancia ante fallos: si falla un procesador los demás procesadores pueden hacerse cargo de sus tareas, ya que la base de datos reside en los discos, a los cuales tienen acceso todos los procesadores.

* *Sin compartimiento*: No se comparte ningún componente. Cada nodo de la máquina consta de un procesador, memoria y uno o más discos. Los procesadores de un nodo pueden comunicarse con un procesador de otro nodo utilizando una red de interconexión de alta velocidad. Un nodo funciona como el servidor de los datos almacenados en los discos que posee.
* *Jerárquica*: Combina las arquitecturas anteriores. A alto nivel el sistema está formado por nodos que están conectados mediante una red de interconexión y que no comparten ni memoria ni discos. Así, el nivel más alto es una arquitectura sin compartimiento. Cada nodo del sistema podría ser en realidad un sistema de memoria compartida con algunos procesadores. Alternativamente, cada nodo podría ser un sistema de disco compartido y cada uno de estos sistemas de disco compartido podría ser a su vez un sistema de memoria compartida. De esta manera, un sistema podría construirse como una jerarquía con una arquitectura de memoria compartida con pocos procesadores en la base, en lo más alto una arquitectura sin compartimiento y quizá una arquitectura de disco compartido en el medio.

**Sistemas distribuidos**

En un sistema distribuido de bases de datos se almacena la base de datos en varias computadoras. Varios medios de comunicación pueden poner en contacto las distintas computadoras de un sistema distribuido. No comparten ni memoria ni discos. Las computadoras de un sistema distribuido pueden variar en tamaño y función pudiendo abarcar desde las estaciones de trabajo a los grandes sistemas. Las BD normalmente se encuentran en varios lugares geográficos distintos, se administran de forma separada y poseen una interconexión más lenta que las BD paralelas sin compartimientos.

En un sistema distribuido se dan dos tipos de transacciones, las locales (acceder a datos alojados en el lugar donde se inició la transacción) y las globales (acceder a datos alojados en sitios distintos).

La principal ventaja de construir un sistema distribuido de bases de datos es poder disponer de un entorno donde los usuarios puedan acceder desde una única ubicación a los datos que residen en otras ubicaciones.

Existe un administrador de bases de datos global responsable de todo el sistema y administradores locales, por lo que cada ubicación es capaz de mantener un grado de control sobre los datos que se almacenan localmente.

Si un sitio de un sistema distribuido falla, los sitios restantes pueden seguir trabajando.

Aspectos de la implementación

La atomicidad de las transacciones es un aspecto importante de la construcción de un sistema distribuido de bases de datos. Si una transacción se ejecuta a lo largo de dos sitios, puede comprometerse en un sitio y cancelarse en otro, lo que conduciría a un estado de inconsistencia. Los protocolos de compromiso de transacciones aseguran que tales situaciones no se produzcan. El protocolo de compromiso de dos fases es el más utilizado.

La idea básica es que cada sitio ejecuta la transacción justo hasta antes del compromiso, y entonces deja la decisión del compromiso a un único sitio coordinador; se dice que en ese punto la transacción está en estado preparada en el sitio. El coordinador decide comprometer la transacción sólo si la transacción alcanza el estado preparada en cada sitio donde se ejecutó; en otro caso, el coordinador decide cancelar la transacción. Todos los sitios donde la transacción se ejecutó deben acatar la decisión del coordinador. Si un sitio falla cuando una transacción se encuentra en estado preparada, cuando el sitio se recupere del fallo debería estar en posición de comprometer o cancelar la transacción dependiendo de la decisión del coordinador.

El control de concurrencia es otra característica de una base de datos distribuida ya que una transacción puede acceder a elementos de datos de varios sitios. Si se utiliza bloqueo, se puede realizar de forma local en los sitios que contienen los elementos de datos accedidos, pero también existe posibilidad de un interbloqueo que involucre a transacciones originadas en múltiples sitios. Por lo tanto, es necesario llevar la detección de interbloqueos a lo largo de múltiples sitios.

El principal inconveniente de los sistemas distribuidos de bases de datos es la complejidad añadida que es necesaria para garantizar la coordinación apropiada entre los sitios: Coste de desarrollo del software, mayor probabilidad de errores, mayor sobrecarga de procesamiento.

**U2: SEGURIDAD E INTEGRIDAD DE BD**

Bibliografía: Libro Nº 1, Capitulo 8, items 8.1, 8.2, 8.3, 8.4, 8.5, 8.8, 8.9, Capitulo 9,

items 9.1, 9.2, 9.3, 9.4, 9.5, Capitulo 14, items 14.1, 14.2, 14.3, 14.4, 14.5, 14.6, Capitulo

15, items 15.1, 15.2, 15.3, 15.4, 15.5, 15.6, 15.7, 15.8, Capitulo 16, items 16.1, 16.2,

16.3, 16.5.

Libro Nº 2, Capítulo 6, items 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 6.7, 5.3, Capitulo 15,

items 15.1, 15.2, 15.3, 15.4, 15.5, 15.6, 15.7, Capitulo 17, items 17.1, 17.2, 17.3, 17.4,

17.5, 17.6.

**8.1 INTRODUCCIÓN**

El término integridad se refiere a la exactitud o corrección de los datos en la base de datos. Una base de datos determinada podría estar sujeta a cualquier cantidad de restricciones de integridad (en general) de una complejidad arbitraria. El DBMS necesita estar informado de dichas restricciones y por supuesto, necesita hacerlas cumplir de alguna manera (básicamente rechazando cualquier actualización que de otro modo las violaría).

Al declarar una nueva restricción, el sistema primero debe asegurarse de que la base de datos la satisfaga actualmente. Si no es así, la nueva restricción es rechazada; en caso contrario, es aceptada (es decir, se guarda en el catálogo) e impuesta desde ese momento.

Clasificación de las restricciones de integridad:

* Restricción de tipo (dominio): especifica los valores válidos para un tipo dado.
* Restricción de atributo: especifica el valor válido de un atributo dado.
* Restricción de varrel: especifica los valores válidos de una varrel determinada.
* Restricción de base de datos: especifica el valor válido de una base de datos dada.

**8.2 RESTRICCIONES DE TIPO**

Una restricción de tipo es (o es equivalente de manera lógica a) una sola enumeración de los valores válidos del tipo.

Las restricciones de tipo son verificadas durante la ejecución de alguna invocación al selector. Como consecuencia, podemos decir que las restricciones de tipo son verificadas de inmediato y, por lo tanto, que ninguna varrel puede adquirir nunca un valor para ningún atributo de cualquier tupia si éste no es del tipo apropiado.

**8.3 RESTRICCIONES DE ATRIBUTO**

Una restricción de atributo es básicamente sólo una declaración para que un atributo dado sea de un tipo en particular.

Las restricciones de atributo son parte de la definición del atributo en cuestión y pueden ser identificadas por medio del nombre de atributo correspondiente. De aquí que una restricción de atributo sólo pueda ser eliminada mediante la eliminación del propio atributo.

**8.4 RESTRICCIONES DE TABLA**

Una restricción de varrel es la que es impuesta a una varrel individual.

Las restricciones de varrels siempre son verificadas de inmediato, como parte de la ejecución de cualquier instrucción que pudiera ocasionar que fueran violadas.

**8.5 RESTRICCIONES DE BASE DE DATOS**

Una restricción de base de datos es aquella que relaciona dos o más varrels distintas.

En general, la verificación de restricciones de base de datos no puede hacerse de inmediato, sino que debe diferirse hasta el final de la transacción.

**8.8 CLAVES**

Claves candidatas

Sea R una varrel (entidad?). Por definición, el conjunto de todos los atributos de R tiene la propiedad de unicidad, lo que significa que ningún par de tupias que estén dentro del valor de R en un momento dado, pueden estar duplicadas entre sí. En la práctica, a menudo se da el caso de que algún subconjunto propio del conjunto de atributos de R tiene también la propiedad de unicidad. Estos hechos constituyen la intuición que está detrás de la definición de la clave candidata:

* Sea K un conjunto de atributos de la varrel R. Entonces K es una clave candidata de R si, y solamente si, posee las dos propiedades siguientes:
  + Unicidad: Jamás, ningún valor válido de R contiene dos tupias distintas con el mismo valor de K.
  + Irreductibilidad: Ningún subconjunto propio de K tiene la propiedad de unicidad.

Observe que toda varrel tiene por lo menos una clave candidata. La propiedad de unicidad de dichas claves es muy clara. En cuanto a la propiedad de irreductibilidad, la idea es que si especificáramos una "clave candidata" que no fuera irreducible, el sistema no estaría al tanto del estado de las cosas; y por lo tanto, no podría hacer cumplir adecuadamente la restricción de integridad asociada. Por ejemplo, suponga que definiéramos la combinación {V#,CIUDAD} —en vez de sólo {V#}— como la clave candidata para proveedores. Entonces el sistema no haría cumplir la restricción de que los números de proveedor son "globalmente" únicos; en su lugar, sólo haría cumplir la restricción más débil de que los números de proveedor son "localmente" únicos dentro de una ciudad. Por esta razón, entre otras, requerimos que las claves candidatas no incluyan atributo alguno que sea irrelevante para los fines de identificación única^

**U3: OPTIMIZACIÓN**

**LIBRO 1**

**CAPÍTULO 17**

**LIBRO 2**

**CAPÍTULO 14**

La optimización de consultas es el proceso de selección del plan de evaluación de consultas más eficiente de entre las muchas estrategias generalmente disponibles para el procesamiento de una consulta dada, especialmente si es compleja.

No se espera que los usuarios escriban las consultas de modo que puedan procesarse de manera eficiente. Por el contrario, se espera que el sistema cree un plan de evaluación de consultas que minimice el coste de la evaluación.

Un aspecto de la optimización de consultas tiene lugar en el nivel del álgebra relacional, donde el sistema intenta hallar una expresión que sea equivalente a la expresión dada, pero de ejecución más eficiente. Otro aspecto es la elección de una estrategia detallada para el procesamiento de la consulta, como puede ser la selección del algoritmo que se utilizará para ejecutar una operación, la selección de los índices concretos que se van a emplear, etcétera.

La diferencia en tiempo de evaluación entre una estrategia buena y una mala suele ser sustancial, y puede ser de varios órdenes de magnitud. Por tanto, merece la pena que el sistema pase una cantidad importante de tiempo en la selección de una buena estrategia para el procesamiento de la consulta, aunque esa consulta sólo se ejecute una vez.

**14.1. VISIÓN GENERAL**

Dada una expresión del álgebra relacional, es labor del optimizador de consultas diseñar un plan de evaluación de consultas que calcule el mismo resultado que la expresión dada, y que sea la manera menos costosa.

Para hallar el plan de evaluación de consultas menos costoso el optimizador necesita generar planes alternativos que produzcan el mismo resultado que la expresión dada, estimar los costes y escoger el menos costoso. Este cálculo no suele resultar posible sin evaluar realmente el plan. En lugar de eso, los optimizadores hacen uso de la información estadística sobre las relaciones, como los tamaños de las relaciones y las profundidades de los índices, para realizar una buena estimación del coste de cada plan. El acceso a los discos, que resulta lento en comparación con el acceso a la memoria, suele dominar el coste del procesamiento de las consultas.

La generación de planes de evaluación de consultas implica dos etapas:

1. La generación de expresiones que sean equivalentes lógicamente a la expresión dada y
2. La anotación de las expresiones resultantes en maneras alternativas de generar planes de evaluación de consultas alternativos.

Las dos etapas están entrelazadas en el optimizador de consultas –se generan y se anotan unas expresiones, luego se generan y se anotan otras expresiones, etcétera.

Las vistas materializadas ayudan a acelerar el procesamiento de ciertas consultas.

**14.2. ESTIMACIÓN DE LAS ESTADÍSTICAS DE LOS RESULTADOS DE LAS EXPRESIONES**

El coste de cada operación depende del tamaño y de otras estadísticas que se encuentran en el catálogo de BD.

El plan de evaluación de consultas que tenga el coste estimado de ejecución más reducido puede no tener el coste real de ejecución más bajo. Sin embargo, aunque las estimaciones no sean muy precisas, los planes con los costes estimados más reducidos tienen costes de ejecución reales que son los más reducidos o se hallan cercanos a los costes reales de ejecución más bajos.

**14.2.1. Información del catálogo 356**

Los catálogos de los SGDD almacenan la siguiente información estadística sobre las relaciones de las bases de datos:

* , el número de tuplas de la relación .
* , el número de bloques que contienen tuplas de la relación .
* , el tamaño de cada tupla de la relación en bytes.
* , el factor de bloqueo de la relación , es decir, el número de tuplas de la relación que caben en un bloque.
* , el número de valores distintos que aparecen en la relación para el atributo . Este valor es igual que el tamaño de . Si es una clave de la relación , es .

La última estadística, , también puede calcularse para conjuntos de atributos, en vez de sólo para atributos aislados. Por tanto, dado un conjunto de atributos, , es el tamaño de .

Si se supone que las tuplas de la relación se almacenan físicamente juntas en un archivo, se cumple la siguiente ecuación:

Las estadísticas sobre los índices, como las alturas de los árboles B+ y el número de páginas hojas de los índices, también se conservan en el catálogo. Los optimizadores verdaderos suelen conservar información estadística adicional para mejorar la precisión de sus estimaciones de costes de los planes de evaluación.

Si se desean conservar estadísticas precisas, cada vez que se modifica una relación también hay que actualizar las estadísticas. Esta actualización supone una sobrecarga sustancial. Por tanto, la mayor parte de los sistemas no actualizan las estadísticas con cada modificación. En lugar de eso, actualizan las estadísticas durante los periodos de poca carga del sistema. En consecuencia, puede que las estadísticas utilizadas para escoger una estrategia de procesamiento de consultas no sean completamente exactas. Sin embargo, si no se producen demasiadas actualizaciones en los intervalos entre las actualizaciones de las estadísticas, éstas serán lo bastante precisas como para proporcionar una buena estimación de los costes relativos de los diferentes planes.

**14.2.2. Estimación del tamaño de la selección**

La estimación del tamaño del resultado de una operación de selección depende del predicado de la selección. En primer lugar, se considerará un solo predicado de igualdad, luego un solo predicado de comparación y, finalmente, combinaciones de predicados.

* : Si se supone una distribución uniforme de los valores (es decir, que cada valor aparece con igual probabilidad), se puede estimar que el resultado de la selección tiene tuplas, suponiendo que el valor aparece en el atributo de algún registro de . La suposición de que el valor de la selección aparece en algún registro suele ser cierta, y las estimaciones de costes suelen hacerla de manera implícita. No obstante, no suele ser realista suponer que cada valor aparece con igual probabilidad. Pese al hecho de que la suposición de distribución uniforme no suele ser correcta, resulta una aproximación razonable de la realidad en muchos casos, y ayuda a mantener la representación relativamente sencilla.
* : Considérese una selección de la forma . Si el valor real utilizado en la comparación está disponible en el momento de la estimación del coste, puede hacerse una estimación más precisa. Los valores mínimo y máximo del atributo pueden almacenarse en el catálogo. Suponiendo que los valores están distribuidos de manera uniforme, se puede estimar el número de registros que cumplirán la condición como 0 si , como si y como en otro caso.
* Selecciones complejas:
  + Conjunción
  + Disyunción
  + Negación

**14.2.3. Estimación del tamaño de las reuniones**

El producto cartesiano contiene tuplas. Cada tupla de ocupa bytes, de donde se puede calcular el tamaño del producto cartesiano. La estimación del tamaño de una reunión natural resulta algo más complicada que la estimación del tamaño de una selección del producto cartesiano. Sean y dos relaciones.

* Si —es decir, las relaciones no tienen ningún atributo en común— entonces es igual que , y se puede utilizar la técnica de estimación anterior para los productos cartesianos.
* Si es una clave de , entonces se sabe que cada tupla de se combinará como máximo con una tupla de . Por tanto, el número de tuplas de no es mayor que el número de tuplas de . El caso de que sea una clave de es simétrico al caso que se acaba de describir. Si forma una clave externa de , que haga referencia a , el número de tuplas de es exactamente el mismo que el número de tuplas de .

**Bla bla bla que ni da**

**14.5. VISTAS MATERIALIZADAS**

Cuando se define una vista, normalmente la base de datos sólo almacena la consulta que define la vista. Por el contrario, una vista materializada es una vista cuyo contenido se calcula y se almacena. Las vistas materializadas constituyen datos redundantes, en el sentido de que su contenido puede deducirse de la definición de la vista y del resto del contenido de la base de datos. No obstante, resulta mucho más económico en muchos casos leer el contenido de una vista materializada que calcular el contenido de la vista ejecutando la consulta que la define. Las vistas materializadas resultan importantes para la mejora del rendimiento de algunas aplicaciones.

**14.5.1. Mantenimiento de las vistas**

Un problema con las vistas materializadas es que hay que mantenerlas actualizadas cuando se modifican los datos empleados en la definición de la vista.

Las vistas pueden mantenerse mediante código escrito a mano. Otra opción es la definición de desencadenadores para la inserción, la eliminación y la actualización de cada relación de la definición de la vista. Los desencadenadores deben modificar el contenido de la vista materializada para tener en cuenta el cambio que ha provocado que se active el desencadenador. Una manera simplista de hacerlo es volver a calcular completamente la vista materializada con cada actualización. Una opción mejor es modificar sólo las partes afectadas de la vista materializada, lo que se conoce como mantenimiento incremental de la vista.

Los sistemas modernos de bases de datos proporcionan más soporte directo para el mantenimiento incremental de las vistas. Los programadores de bases de datos ya no necesitan definir desencadenadores para el mantenimiento de las vistas. Por el contrario, una vez que se ha declarado materializada una vista, el sistema de bases de datos calcula su contenido y actualiza de manera incremental el contenido cuando se modifican los datos subyacentes.

**14.5.2. Mantenimiento incremental de las vistas**

Para comprender el modo de mantener de manera incremental las vistas materializadas se comenzará por considerar las operaciones individuales y luego se verá la manera de manejar una expresión completa. Los cambios de cada relación que puedan hacer que se quede desactualizada una vista materializada son las inserciones, las eliminaciones y las actualizaciones. Para simplificar la descripción se sustituyen las actualizaciones a cada tupla por la eliminación de esa tupla seguida de la inserción de la tupla actualizada. Por tanto, sólo hay que considerar las inserciones y las eliminaciones. Los cambios (inserciones y eliminaciones) en la relación o en la expresión se denominan su diferencial.

**14.5.3. Optimización de consultas y vistas materializadas**

La optimización de consultas puede llevarse a cabo tratando las vistas materializadas igual que a las relaciones normales. No obstante, las vistas materializadas ofrecen más oportunidades para la optimización como, por ejemplo, reescritura de las consultas para el empleo de vistas materializadas y sustitución del empleo de una vista materializada por la definición de la vista.

Otro problema de optimización relacionado es el de la selección de las vistas materializadas, es decir, la identificación del mejor conjunto de vistas para su materialización. Esta decisión debe tomarse con base en la carga de trabajo del sistema, que es una secuencia de consultas y de actualizaciones que refleja la carga típica del sistema. Un criterio sencillo sería la selección de un conjunto de vistas materializadas que minimice el tiempo global de ejecución de la carga de trabajo de consultas y de actualizaciones, incluido el tiempo empleado para conservar las vistas materializadas. Los administradores de bases de datos suelen modificar este criterio para tener en cuenta la importancia de las diferentes consultas y actualizaciones: puede ser necesaria una respuesta rápida para algunas consultas y actualizaciones, mientras que puede resultar aceptable una respuesta lenta para otras. Los índices son como las vistas materializadas, en el sentido de que también son datos obtenidos, pueden acelerar las consultas y pueden desacelerar las actualizaciones. Por tanto, el problema de la selección de índices se halla íntimamente relacionado con el de la selección de las vistas materializadas, aunque resulta más sencillo.