**BASES DE DATOS II**

Docentes: Marcela Andrea Vera, Alejandro Aguirre

Unidades:

1. Administración de BD
2. Seguridad e Integridad de BD
3. Optimización
4. BD Distribuidas
5. BD orientada a objetos
6. Nuevas tendencias en BD

Regularizar:

* 80% asistencia
* Parciales o recuperatorios: nota >= 40%
* Trabajos prácticos: nota >= 50%
* Participación activa en el aula virtual
* Aprobación con 60% o más de las evaluaciones virtuales por unidad.

Promocionar:

* 80% asistencia
* Parciales o su recuperatorios: nota >= 80
* Trabajos prácticos: nota >= 80
* Participación activa en el aula virtual
* Aprobación con 60% o más de las evaluaciones virtuales por unidad.

Cronograma:

* 1º TP: 16 de mayo
* 1º Parcial: 19 de mayo
* 2º TP: presentación oral 30 de junio
* 2º Parcial: 16 de junio
* Recuperatorio 1º Parcial: 23 de junio
* Recuperatorio 2º Parcial: 23 de junio

Aula Virtual: Moodle

* Mail: marcelaandrea.vera@gmail.com
* Aula virtual: http://107.170.76.119/moodle/

**U1: ADMINISTRACIÓN DE BD**

**Conceptos básicos:**

* BD: es una colección de archivos, datos, información; ordenada, organizada y relacionada con la finalidad de permitir el manejo de la información para su procesamiento. Cada uno de los archivos representan una colección de registros y cada registro está compuesto de una colección de campos. Cada uno de los campos de cada registro permite llevar información de alguna característica o atributo de alguna entidad del mundo real.
* DBMS: es un conjunto de programas que se encargan de manejar la creación y todos los accesos a las BD. Se compone de un Lenguaje de Definición de Datos (DDL: Data Definition Languaje), de un Lenguaje de Manipulación de Datos (DML: Data Manipulation Languaje), y de un Lenguaje de Consulta (SQL: Structured Query Languaje).
* DBA: es el responsable del control general del sistema a nivel técnico. Es el responsable de los permisos que poseen los usuarios de la BD.

**Funciones del DBA**

* Definir el esquema conceptual de la BD.
* Definir el esquema interno de la BD.
* Establecer el enlace con los usuarios (desarrolladores).
* Definir las restricciones de seguridad e integridad.
* Definir las políticas de backup y recuperación.
* Supervisar la performance y responder a los requerimientos cambiantes.
* Administra los datos.

**Funciones del DBMS**

* Definición de datos
* Manipulación de datos
* Optimización y ejecución
* Seguridad e integridad de datos
* Recuperación de datos
* Diccionario de datos
* Concurrencia

**DBMS**

El DBMS sirve como interface entre la BD física y las peticiones del usuario. El DBMS interpreta las peticiones de entrada/salida del usuario y las manda al sistema operativo para la transferencia de datos entre la unidad de memoria secundaria y la memoria principal. En sí, un sistema manejador de BD es el corazón de la BD, ya que se encarga del control total de los posibles aspectos que la puedan afectar. La principal ventaja de utilizarlo es que devuelve tanto los datos como la estructura. Antes se necesitaba un conocimiento completo del sistema de almacenamiento.

**Implementación del Esquema Conceptual** (responsabilidad del DBA)

* Describe la estructura lógica global de la BD mediante un modelo abstracto de datos comprensible por el SGBD. Se definen la descripción de atributos, de entidades, las conexiones y las restricciones de integridad asociadas a la semántica (significado). Podemos decir que describe qué datos son almacenados realmente en la BD y las relaciones que existen entre los mismos, describe la BD completa en términos de su estructura de diseño. El nivel conceptual de abstracción lo usan los administradores de BD, quienes deben decidir qué información se va a guardar en la BD.
* El esquema conceptual representa la visión organizacional de la BD que se obtiene al integrar los requerimientos de todos los usuarios en una empresa; y es totalmente independiente de las estructuras físicas de almacenamiento y de la representación final de los datos que los usuarios manejan. La implantación de este esquema es responsabilidad del DBA.
* Partes:
  + *Definición de datos*: se describen el tipo de datos y la longitud de campo de todos los elementos direccionables y precisión de la información que será guardada en la BD. Los elementos por definir incluyen artículos elementales (atributos), totales de datos y registros conceptuales (entidades).
  + *Relaciones entre datos*: Se definen las relaciones entre datos para enlazar tipos de registros relacionados para el procesamiento de archivos múltiples para obtener información compuesta y procesos complejos.
* En el nivel conceptual la BD aparece como una colección de registros lógicos, sin descriptores de almacenamiento. En realidad, los archivos conceptuales no existen físicamente. La transformación de registros conceptuales a registros físicos para el almacenamiento se lleva a cabo por el sistema y es transparente al usuario.

**Implementación del Esquema Interno**

Es el nivel más bajo de abstracción, describe qué datos son almacenados realmente en la BD y las relaciones que existen entre los mismos. Describe la BD completa en términos de su estructura de diseño. El diseño físico es el proceso de producir la descripción de la implementación de la BD en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos. Entre el diseño físico y el diseño lógico hay una retroalimentación, ya que alguna de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

**Implementación del Esquema Físico**

El propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior.

* Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas
* Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
* Diseñar el modelo de seguridad del sistema.

En el nivel físico se debe especificar:

* Estrategias de almacenamiento: asignación de espacio de almacenamiento para el conjunto de datos.
* Estrategias de emplazamiento de los datos: para optimizar los recursos a la hora de exportar la BD (tiempo de respuesta, disco, memoria, etc.).
* Caminos de acceso: se incluye la especificación de claves como la de índices y punteros.

**Implementación de los Esquemas Externos**

* Es el nivel más alto de abstracción, es lo que el usuario final puede visualizar del sistema terminado, describe sólo una parte de la BD al usuario acreditado para verla. El sistema puede proporcionar muchas visiones para la misma BD.
* *Vistas*: son una especie de tablas virtuales; es decir no existen físicamente, sino que forman mediante la selección y/o filtrado de los componentes de otras tablas, una vista puede ser definida en base a una lista previa. Esto significa que puede crearse dependencia entre las vistas. Cuando una vista es definida en base a otra, se dice que es dependiente de esta, por lo tanto, se suprimirá automáticamente la vista dependiente si se suprime la vista original.
  + *Objetivo*: Facilitar al usuario la percepción que éste tiene de la BD así como el trabajo que van a desarrollar sobre ésta.
  + *Ventajas*:
* Perspectivas directas: proporcionarse diversos modelos de información basados en los mismos datos, enfocándolos hacia distintos usuarios con necesidades específicas. El mostrar la información desde distintos ángulos nos ayuda a crear ambientes de trabajo y operación acordes a los objetivos de la empresa.
* Transparencias en las modificaciones: El usuario final no se verá afectado por el diseño o alteraciones que se realicen en el esquema conceptual de la BD.
* Seguridad: Las vistas proporcionan de manera natural un medio para ocultar y proteger datos, dado que solo se presenta al usuario una selección de los atributos existentes.

**Diccionario de datos:**

* Es una herramienta que facilita el control y manejo de la información acerca de datos en el diseño, implementación, operación y expansión de fases de una BD**.**
* El DD es un lugar donde se deposita información acerca de datos como origen, descripción, relaciones y otros datos, es decir, el diccionario de datos es una BD misma, la cual deposita datos acerca de los datos, el DD es una guía y contiene “mapas guías” para la BD en vez de “nuevos datos”, es decir, es un lugar en donde se almacena o se mantiene un conjunto de estados (controles), información relacionada con los diferentes tipos de registros (tablas) privilegios de los usuarios y estadísticas (cuántos registros tiene cada tabla, índices, etc.).
* Los DD de los sistemas de BD (DBMS) no son iguales, aunque mantienen los mismos lineamientos o las mismas características.
* Es un catálogo de todo lo que tiene el sistema. Contiene los alias, procedimientos de almacenado, etc.
* Contiene las características lógicas de los sitios donde se almacenan los datos del sistema, incluyendo nombre, descripción, alias, contenido y organización. Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.
* En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos en todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario guarda los detalles y descripciones de todos estos elementos.
* Si los analistas desean conocer cuántos caracteres abarca un determinado dato o qué otros nombres recibe en distintas partes del sistema, o dónde se utiliza, encontrarán las respuestas en un diccionario de datos desarrollado en forma apropiado.

*Ventajas*:

* Permite mantener la historia.
* Sirve como documentación de cómo fue la BD
* Facilita el análisis de la performance de la BD (tablas más accedidas, de más crecimiento, más actualizadas o modificadas)
* Sirve para hacer la recuperación de una BD que entra en problemas
* Sirve para encontrar problemas, errores (tablas corruptas, por ejemplo).

**Conceptos Básicos**

* Un SI es un conjunto de herramientas a través de las que se generan y gestionan datos, diferentes actores interactúan con él a través de distinto perfiles. Ese conjunto de datos, que forman el núcleo del sistema de información, deben ser almacenados en un repositorio confiable capaz de resguardarlos asegurando la integridad y la seguridad de los mismos. Los sistemas de información modernos utilizan BD para tal fin.
* BD es un SW capaz de almacenar en forma confiable y segura la información, manteniéndola disponible para que pueda ser recuperada cuando lo usuarios del sistema lo soliciten. En la actualidad, el valor de la información hizo que los sistemas de gestión de los datos ocupen un rol fundamental en la vida de toda organización. Por este motivo surge la necesidad de contar con personal adecuado que se encargue de llevar adelante la administración de los sistemas y en particular de sus BD. Para asegurar el correcto funcionamiento de la base y optimizar el acceso a los datos es imprescindible la figura de un administrador de BD o DBA (DataBase Administrator).

**Funciones del DBA**

* Debe conocer a la perfección el lenguaje que utiliza el sistema para comunicarse con la BD, cuando un usuario necesita agregar, modificar o extraer información en el sistema, denominado SQL.
* Otra de sus tareas será definir, de acuerdo a las características de cada herramienta y de la institución en la que se implementará, las políticas de seguridad y asegurar la performance de la BD. En cuanto a performance, es el encargado de realizar todas las tareas de mantenimiento, tanto de HW como de SW, para que la BD responda de manera eficiente a las diferentes, esto incluye actualizaciones, configuraciones, etc.
* Seguridad: definir las políticas de acceso a la información. Políticas de resguardos o backups y cuáles serán los pasos a seguir ante un fallo de HW o de SW.
* El rol del DBA es continuo, ya que no se centra en una parte específica del proceso de evolución de los sistemas. Se encuentra presente desde el desarrollo hasta la implementación y durante su posterior uso. Debe monitorear constantemente el funcionamiento de las bases, realizando tareas preventivas y correctivas.
* El rol del DBA excede el alcance de un sistema, por lo que deberá tener una visión integral de todos los módulos que integran un único sistema de información.

**Herramientas de un DBMS**

* Afinación de la BD (tunning)
  + Ajustes y cambios en la BD para optimizarla y mejorar su performance. No es un proceso inicial, sino continuo. El responsable debería tener toda la libertad necesaria para efectuar los cambios que crea convenientes.
  + Requisitos de una correcta afinación:
    - Independencia física de los datos: modificar el esquema interno sin alterar el esquema conceptual ni los programas de aplicación.
    - Medios: para supervisar automáticamente el uso de la BD con el fin de que puedan hacerse los ajustes necesarios.
  + Los manejadores actuales de BD ya incorporan medios para la afinación automática (comandos, etc.).
* Medición del desempeño
  + Es responsabilidad del DBA organizar el sistema de modo que se obtenga el desempeño que sea “mejor para la empresa”, y realizar los ajustes apropiados cuando cambien los requerimientos. Es necesario reorganizar la BD en forma periódica (descargarla y volverla a cargar) con el fin de garantizar que los niveles de desempeño sigan siendo aceptables.
  + Los datos obtenidos del desempeño se comparan con aquellos datos esperados. Esta razón proporciona los factores de multiprogramación, Mn, para cada uno de los procesos básicos h que son parte de los cálculos de la BD. En la práctica, estos factores pueden variar de 1 a 0,1.
  + Para estas pruebas no se requiere la existencia de la BD completa y puede ser que ni siquiera exista el estudio piloto, que consiste en la ejecución de una secuencia adecuada de operaciones aleatorias o secuenciales de lectura o escritura, junto con una cantidad equivalente del uso del CPU. Es posible escribir y ejecutar un programa que imite la operación propuesta, en el sistema real que se va a utilizar, con el costo del esfuerzo de unos cuantos días.
* Reorganización física y lógica
  + La reorganización consiste en leer el archivo en forma en que se utilizaría al realizar el procesamiento serial y escribir los registros nuevos y viejos en el archivo nuevo, dejando fuera todos los registros marcados como eliminados lógicamente; y se crearán nuevos índices con base a los nuevos valores.
  + La frecuencia de reorganización depende de la actividad de inserción dentro del archivo; y se debe realizar antes de que el archivo esté realmente lleno para evitar problemas en tiempos de mucha actividad. Las reorganizaciones físicas son necesarias para mejorar el rendimiento, añadir una nueva estructura de acceso, agilizar las operaciones de obtención y actualización, disminuir los tiempos de respuesta, minimizar el espacio de almacenamiento y optimizar el consumo de recursos. Las reorganizaciones lógicas pueden modificar el esquema conceptual, pero no alterar el esquema externo ni los programas de aplicación; puede ser un orden de visualización en las vistas.

**Respaldo y Recuperación**

* El DBA debe definir y poner en práctica un plan (política) de recuperación adecuado que incluya una descarga o vaciado periódico de la BD en un medio de almacenamiento de respaldo y procedimientos para volver a cargar la BD a partir del vaciado más reciente. El respaldo y recuperación consiste en contar con un mecanismo que permite la fácil recuperación de los datos en caso de ocurrir fallos en el sistema de BD.
* El objetivo del concepto de recuperación es el de proteger la BD contra fallas lógicas y físicas que destruyan los datos en todo o en parte independientemente de la naturaleza de las fallas estas pueden afectar los aspectos de almacenamiento de la BD como son:
  + Fallas que provocan la pérdida de la memoria volátil.
  + Fallas que provocan la pérdida del contenido de la memoria secundaria.
* En un sistema de BD, recuperación significa restaurar la BD a un estado que se sabe que es correcto, después de una falla que provoca que se considere que el estado actual es incorrecto. Podemos tener la seguridad de que la BD es recuperable, si aseguramos que cualquier parte de la información que contiene, puede ser reconstruida, a partir de otra información que se encuentra almacenada redundantemente en algún lugar del sistema.

Pasos para recuperar la información

* *Detección del error*: el proceso de recuperación se inicia al detectar la existencia de un error. Es posible distinguir una variedad de puntos de entrada en el proceso de recuperación. Se considerarán fallas de sistemas detectadas por falta de acción del sistema o por verificaciones irrecuperables de redundancia y salida incorrecta observada por un usuario.
* *Determinación de la fuente del error*: para decidir cuál es la mejor acción correctora es necesario determinar la extensión del daño. Desde luego este esfuerzo es muy relacionado con la determinación del tiempo y la causa del error. Después de una caída o cuando el procesamiento sea interrumpido debido a una señal de error, es necesario determinar tanto aquellas áreas del archivo de datos que sean sospechosas como cuál fue la transacción que no se concluyó.
* *Ubicación de errores secundarios*: cuando se ha detectado un error que provocó una modificación inadecuada a un archivo un rastreo a través de las listas de actividad encontrara aquellas transacciones que emplearon el bloque correcto. Entonces es posible volver a introducir automáticamente el bloque correcto de las transacciones afectadas y producir resultados correctos. Si se actualizaron bloques mediante transacciones que leyeron bloques incorrectos antes de existir es necesario restaurar aún más el archivo.
* *Aplicación de correcciones*: si la extensión del daño es limitada, puede utilizarse un proceso de volver a aplicar. Las porciones dañadas del archivo se restauran aplicando primero aquellas imágenes anteriores a los bloques en error reemplazando después de las transacciones incompletas.

**Tipos de Backup**

* *Full*(o de nivel 0): copia toda la BD tal como está. Depende del tamaño y del crecimiento de la BD.
* *Diferencial* (o de Nivel N: 1, 2,…): copia los cambios que ocurrieron desde el último Full.

En una restauración diferencial, primero restauro el full y, sobre éste, se le pone el último diferencial (o el que se quiere). Los niveles de Diferencial indican desde dónde se recuperan, posiblemente.

* *Incremental* (o Backup de Log): copia los cambios desde el último Incremental. Únicamente existe para el log de transacciones, y es útil para el mismo, no tanto para los datos.

En una restauración de backups incrementales, se restaura el Full y luego los incrementales en orden (no el que se quiere, porque están ordenados).

Una política de Backups completa tiene los 3 niveles en su debida periodicidad. El tiempo de pérdida de datos debe quedar claro con la empresa, ya que debe informar la tolerancia de pérdida. El tiempo de Backup del log depende más del tiempo de pérdida tolerado.

**Evolución de los Sistemas de Administración de BD**

Ha traído dispersión de los datos en sitios local o geográficamente dispersos. La necesidad de integrar y compartir dicha información implica el nacimiento de una nueva tecnología capaz de conformar de manera consistente la información de las organizaciones. Una de las tecnologías que trabaja el problema de integración de información, es la de BD distribuidas.

BD Centralizadas → BD Clientes-servidor → BD Distribuidas.

Centralizadas: Los equipos están conectados a un servidor centralizado, donde está almacenada laBD. La interacción con el usuario como la aplicación residen en el cliente, siendo el servidor del depositario de los datos. En un sistema centralizado, la caída o desconexión del servidor, provoca la caída total del sistema.

Cliente-Servidor: Es la integración distribuida en un sistema de red, con los recursos, medios y aplicaciones, que, definidos modularmente en los servidores, administran, ejecutan ya tienden a las solicitudes de los clientes; todos interrelacionados física y lógicamente. En un esquema clientes-servidor, se denomina cliente a la máquina que solicita un determinado servicio y servidor a aquella que proporciona tal servicio.

Características:

* Varias computadoras están conectadas por medio de una red
* El sistema es abierto esto implica que las computadoras pueden ser de diferente tipo y/o marca corriendo diferentes S.O.
* Por lo general las computadoras que procesan los programas de aplicación se llaman “clientes”
* La computadora que soporta el DBMS y el servicio de los datos se llama “servidor”
* La comunicación entre el cliente y el servidor se hace, por lo general, por medio Remote Procedure Calls (RPC) o servicios de SQ.
* Existen dos variantes principales de esta arquitectura:
  + Cliente-Servidor de 2 capas (bandas)
  + Cliente-Servidor de 3 capas (bandas)

Cliente-Servidor de dos capas

Primera: Lógica de las aplicaciones

Segunda: Administración de los recursos de datos (DBMS)

La intención es mejorar la usabilidad por medio del soporte de interfaces gráficas amigables para el usuario. Mejora la escalabilidad permitiendo trabajar con cierta comodidad hasta 100 usuarios. Mejora la flexibilidad permitiendo que los datos puedan ser compartidos usualmente dentro de ambientes homogéneos.

Detalles técnicos:

* Interface de usuarios del sistema: sesión, entrada/salida de datos, diálogo.
* Administración de los procesos desarrollo y verificación de procesos, servicios y recursos de los procesos.
* Administración de la BD: servicios de archivos y datos. Los servicios GUI se asignan exclusivamente al cliente, los del DBMS al servidor y la administración del procesamiento se distribuye en ambas capas.
* Es muy común que en un cliente servidor de dos capas, las reglas del negocio se efectúen como procedimientos y triggers almacenados en el servidor, y el procesamiento de la loica de la aplicación en el ambiente del cliente.
* Uso: se emplean extensivamente en procesamiento de información cuyo procesamiento no es crítico en tiempo y donde la administración y operación del sistema no es complejo, y en donde la administración y operación del sistema no es complejo y donde la carga de transacciones no es alta.

Costos y limitaciones:

* Escalabilidad: se pueden manejar bien hasta aproximadamente 100 usuarios. Si las reglas del negocio + procedimientos + funciones se alojan en el servidor entonces se reduce el número de usuarios que se pueden agregar posteriormente.
* Interoperabilidad: cuando se emplean procedimientos almacenados para implementar reglas de negocio complejas la posibilidad de operar con DBMS distintos se ve limitada (procedimiento en lenguajes propietarios con poca capacidad de “debugging”).
* Administración y configuración: cuando las aplicaciones residen en el cliente las actualizaciones se deben instalar y testear en cada cliente. La típica falta de uniformidad en la configuración de cada cliente y la falta de control sobre los cambios de configuración subsecuentes incrementa la carga de trabajo.

Ventajas:

* División del procesamiento entre el sistema cliente y el servidor de BD.
* Sistema abierto los clientes y los servidores no están atados por un modelo/marca/SO.
* Independencia de las aplicaciones, los usuarios pueden emplear el SW que les resulte más familiar para acceder al DBMS

Desventajas:

* Mantenimiento de los clientes (aplicaciones y HW)
* Costo en aprendizaje y en personal (heterogeneidad de los componentes)
* Configuración de los distintos componentes del sistema y hacer que funcionen coordinadamente es complejo.

Cliente-Servidor de tres capas

*Propósito y Origen*: Intentaba cubrir las falencias del modelo de dos capas. Mejora la “escalabilidad” permitiendo trabajar con cierta comodidad hasta miles de usuarios. Mejora la flexibilidad permitiendo que los datos puedan ser compartidos en ambientes heterogéneos. Mejora el mantenimiento de los clientes, ubicando gran parte del procesamiento en la capa del medio haciendo los clientes más “flacos”.

Detalles técnicos:

* La capa del medio que se ubica entre el cliente y el servidor de datos es el lugar donde se ubican la administración de los procesos y las reglas de negocio (procedimientos y triggers del modelo de dos capas).
* La capa del medio administra las funciones de administración de la BD (servicios de datos y archivos) sin usar lenguajes propios del DBMS.
* La capa del medio también se ocupa de la administración de los procesos, desarrollo y verificación de procesos, servicios y recursos de los procesos
* El servidor de la capa del medio, también referido como el Servidor de Aplicaciones (“Application Server”) es el que centraliza la lógica del proceso. La centralización hace que la lógica de los procesos y la administración de los cambios sea más fácil.
* La capa del medio hace un control de las transacciones y pone en “cola” las mismas para asegurar que las transacciones lleguen a su fin de manera segura.
* En algunos casos la capa del medio se divide en dos o más unidades con diferentes funciones y la arquitectura aquí se dice que es “multicapa”. Es el caso de aplicaciones de Internet, cuyos clientes son “flacos” escritos en HTML y cuentan con un servidor de aplicaciones escritos en C++ o Java.
* El “link” entre las dos aplicaciones la hace un “Web Server” que recibe los requerimientos de los usuarios y genera documentos HTML en conjunto con los servicios que le provee el resto de los servidores. Esta es la arquitectura de una aplicación Intranet/Extranet.
* USO: Se emplean en ambientes cliente/servidor distribuidos donde se necesita compartir recursos tales como DBMS heterogéneos y reglas de negocios.
* La arquitectura de tres capas facilita el desarrollo porque cada capa puede ser construida y ejecutada sobre una plataforma diferente, haciendo la organización e implementación más fácil.
* Permite, además que las capas se puedan desarrollar en diferentes lenguajes: Lenguaje GUI, clientes Internet“flacos”(HTML) para la capa del cliente
* C, C++, Java, VisualBasic, SmallTalk, ADA para la capa del medio.
* SQL para la capa de la BD.
* El modelo de tres capas facilita la migración de las aplicaciones, ya que el sistema nuevo podría ir generándose y ejecutándose en paralelo con el nuevo hasta que el sistema se haya migrado completamente.

Costos y limitaciones:

* La construcción de una arquitectura de tres capas es un trabajo complejo.
* Las herramientas de programación de la capa del medio no proveen aún de todas las funcionalidades deseables de un ambiente de computación distribuido.
* Un problema que se presenta en el desarrollo de las aplicaciones es la separación de la lógica del cliente, la separación entre la lógica del proceso y la lógica de los datos no es siempre clara.
* Los criterios para juzgar la ubicación de una función en una capa determinada son los siguientes:
  + Fácil de desarrollar y testear
  + Fácil de administrar
  + Escala del servidor
  + Performance del procesamiento y de la red

Distribuidas: Una BD está dividida en fragmentos almacenados en diferentes sitios de la red. Todas las operaciones realizadas en los sitios se ven reflejadas en la BD que está almacenada en el servidor. El cliente debe conocer la topología de la red, así como la disposición y ubicación de los datos. Se delega parte de la gestión de la BD a los clientes.

Un Sistema de BD Distribuidas es un conjunto de sitios autónomos de procesamiento de datos que se encuentran a una distancia determinada unos de otros conectados por una red y cooperando entre sí.

Características:

* Un DDBMS debe proveer acceso a los datos distribuidos.
* Los datos físicos están distribuidos en computadoras diferentes, pero el sistema presenta una sola imagen al usuario.
* Técnicamente es muy difícil proveer una imagen simple de la BD y pocos proveedores presentan una solución completa
* El DDBMS debe proveer el acceso a los datos como si ellos estuvieran en la computadora local.
* No se comparte memoria principal.

Ventajas:

* *Mayor paralelismo*: Al estar los datos distribuidos en varias localidades una consulta puede ser subdividida en varias subconsultas que se ejecuten en paralelo
* *Mayor independencia*: Cada localidad puede administrar sus propios datos de manera local y, por lo tanto, definir sus estrategias de acceso, seguridad, integridad, etc.
* *Mayor flexibilidad*: La introducción de cambios es más sencilla. Es más fácil agregar nuevas localidades si es necesario, que mover o cambiar una computadora centralizada.
* *Mayor disponibilidad*: Si se produce una falla en una localidad las demás localidades puedan seguir trabajando. Además, si los datos están duplicados en otras localidades, es posible encontrar el dato que se necesita en otro lado.

Desventajas:

* *Mayor costo/complejidad*: El desarrollo del SW en un sistema distribuido es más difícil de desarrollar y estructurar, por lo tanto, es más complejo y costoso. Además, hay costo de comunicación y de almacenamiento adicional de las copias.
* *Mayor dificultad en el control*: Al tener las localidades que trabajan en paralelo, y copias de la BD distribuida en diferentes localidades, es más difícil de garantizar que los algoritmos sean correctos.
* *Mayor riesgo de seguridad*: Al estar el sistema distribuido, es más vulnerable a que existan sabotajes, pérdidas y fugas de información.

**U2: SEGURIDAD E INTEGRIDAD**

**Backup**

Es una copia de información con el fin de que estas copias adicionales puedan utilizarse para restaurar el original después de una eventual pérdida de datos.

Fundamentalmente son útiles para recuperarse de una catástrofe informática o recuperar una pequeña cantidad de archivos que pueden haberse eliminado accidentalmente o corrompido.

La pérdida de datos es muy común: El 66% de los usuarios de internet han sufrido una seria pérdida de datos.

**Tipos de Backup de BD:** (son implementados en formas diversas por cada DBMS)

* Backup Completo (Full): Guarda todos los archivos que sean especificados al tiempo de ejecutarse el respaldo. En el caso que se trate de una BD, realiza el respaldo completo: estructura, datos, procedimientos, triggers, etc.
* Únicamente es posible recuperar toda la información.
* Tiempo de ejecución.
* Backup Diferencial: Se guardan todos los cambios realizados desde el ultimo Backup Completo.
* Sólo se requiere del último respaldo completo y del último respaldo diferencial.
* Velocidad.
* Incrementa su tamaño y tiempo cuando más alejado se ejecute un respaldo full.
* Backup Logs: Se guardan las transacciones.
* Permite que la BD vuelva a un estado consistente.
* Velocidad.

**Duplicado de Información en línea**

RAID (Redundant Array of Inexpensive Disks): conjunto de 2 o más Discos Duros que operan como grupo y logran ofrecer una forma más avanzada de respaldo ya que:

* Es posible mantener copias en línea (redundancia)
* Agiliza las operaciones del sistema.
* El sistema es capaz de recuperar información sin intervención de un administrador.

Configuraciones de tipo RAID:

* *RAID-0*: En esta configuración cada archivo es dividido ("Striped") y sus fracciones son colocadas en diferentes discos. Este tipo de implementación sólo agiliza el proceso de lectura de archivos, pero en ningún momento proporciona algún tipo de respaldo ("redundancy").
* *RAID-1*: este es el primer tipo de RAID que otorga cierto nivel de respaldo. Cada vez que se vaya a guardar un archivo en el sistema éste se copiará integro a DOS discos (en línea), es por esto que RAID-1 también es llamado "Mirroring". También agiliza la lectura de archivos (si se encuentran ocupadas las cabezas de un disco "I/O") ya que otro archivo puede ser leído del otro disco y no requiere esperar a finalizar el "I/O" del primer disco.
* *RAID-3*: Esta configuración al igual que RAID-0 divide la información de todos los archivos ("Striping") en varios discos, pero ofrece un nivel de respaldo que RAID-0 no ofrece. RAID-3 opera con un disco llamado de paridad que guarda fracciones de los archivos necesarias para recuperar toda su información, con esto, es posible reproducir el archivo que se perdió.
* El "disco de paridad" es un punto crítico en el sistema si falla.
* *RAID-5*: RAID-5, no sólo distribuye todos los archivos en un grupo de discos ("Striping"), sino también la información de paridad es guardada en todos los discos del sistema ("Striping"). Esta configuración RAID suele ser usada en sistemas que requieren un "alto nivel" de disponibilidad.

**Implementación Backup**

* Horarios: Programar un horario de ejecución de las copias de seguridad aumenta considerablemente su efectividad y nivel de optimización.
* Autentificación: Sobre el curso de operaciones regulares, las cuentas de usuario y/o los agentes del sistema que representan la copia de seguridad necesitan ser autentificados a cierto nivel. Utilizar un mecanismo de autentificación es una buena manera de evitar que el esquema de la copia de seguridad sea usado por actividades sin autorizar.
* Cadena de confianza: Los soportes de almacenamiento portátiles son elementos físicos y deben ser gestionados sólo por personas de confianza. Establecer una cadena de confianza individual es crítico para defender la seguridad de los datos.
* Validación de copias de seguridad: El proceso por el cual los dueños de los datos pueden obtener información considerando como fueron copiados esos datos. El mismo proceso es también usado para probar conformidad para los cuerpos reguladores fuera de la organización. Terrorismo, complejidad de datos, valores de datos y aumento de la dependencia sobre volúmenes de datos crecientes, todos contribuyen a una ansiedad alrededor y dependencia sobre copias de seguridad satisfactorias.
* Reportando: En configuraciones más largas, los reportes son útiles para monitorizar los medios usados, el estado de dispositivos, errores, coordinación de saltos y cualquier otra información sobre el proceso de copia de seguridad.
* Registrando: En suma a la historia de los reportes generados por el ordenador, actividades y registros de cambio son útiles para así entender mejor la copia de seguridad.
* Verificación: Muchos programas de copia de seguridad hacen uso de Sumas de verificación o hashes. Esto ofrece muchas ventajas. Primero, estos permiten a la integridad de los datos ser verificados sin hacer referencia al archivo original: si el archivo guardado en un medio de copia tiene la misma suma de verificación que el valor salvado, después es muy probable que sea correcto.

**Seguridad e Integridad**

La seguridad en BD frecuentemente se asocia con la integridad, pero ambos conceptos son bastante diferentes. La seguridad de los datos refiere a la protección de los datos contra su revelación, su alteración o destrucción no autorizada, mientras que la integridad se refiere a la precisión o validez de esos datos.

Seguridad significa proteger los datos ante usuarios no autorizados. Integridad significa protegerlos de usuarios autorizados.

**Seguridad**

Existen muchos aspectos sobre el problema de la seguridad, entre ellos se encuentran:

* *Aspectos legales, sociales y éticos*: La persona que hace la petición ¿tiene derecho legal para conocer la información solicitada? En Argentina: La ley 24.766 protege la información confidencial a través de acciones penales y civiles, considerando información confidencial aquella que cumple los siguientes puntos:
  + Es secreta en el sentido que no sea generalmente conocida ni fácilmente accesible para personas introducidas en los círculos en que normalmente se utiliza el tipo de información.
  + Tenga valor comercial para ser secreta.
  + Se hayan tomado medidas necesarias para mantenerla secreta, tomadas por la persona que legítimamente la controla.
  + Por medio de esta ley la sustracción de disquetes, acceso sin autorización a una red o computadora que contenga información confidencial.
* *Controles físicos*: El lugar donde se encuentra la computadora ¿está bajo llave o tiene alguna otra protección?
* *Cuestiones de política*: Si se utiliza un esquema de contraseñas ¿Cómo se las mantiene en secreto? ¿Con que frecuencia son cambiadas?
* *Controles de HW*: ¿La unidad de procesamiento posee características de seguridad como claves de protección de almacenamiento o un modo de operación protegido?
* *Soporte del SO*: ¿El sistema operativo borra el contenido de la memoria principal y los archivos de disco cuando termina de utilizarlos?

**Seguridad de un DBMS**

Actualmente, los DBMS soportan uno o ambos enfoques:

* Control discrecional: un usuario específico tendrá generalmente diferentes derechos de acceso (privilegios) sobre diferentes objetos. Los esquemas discrecionales son muy flexibles. No se puede poner restricciones a los registros por lo que se usarían vistas.

Se necesita un lenguaje que soporte la definición de las restricciones de seguridad discrecionales. Generalmente se define qué está permitido en lugar de lo que no está, por lo tanto, los lenguajes generalmente soportan la definición de las autorizaciones, en vez de restricciones de seguridad.

La granularidad permitida para el control de acceso discrecional dependerá siempre de cada DBMS.

* Control obligatorio: cada objeto de datos esta etiquetado con un nivel de clasificación determinado y a cada usuario se le da un nivel de acreditación. Un objeto de datos especifico solo puede ser accedido por los usuarios que tengan el nivel de acreditación adecuado. Los esquemas obligatorios tienden a ser jerárquicos y comparativamente rígidos.

Los controles de acceso obligatorio son aplicables a las BD en la que los datos tienen una estructura bastante estática y rígida. Cada dato tiene un nivel de clasificación y cada usuario tiene un nivel de acreditación.

El usuario j puede recuperar el objeto i solo si el nivel de acreditación de j es igual o mayor que el nivel de clasificación de i. (Propiedad de seguridad simple).

El usuario j puede actualizar el objeto i solo si el nivel de acreditación de j es igual al nivel de clasificación de i. (Propiedad de estrella)

Podemos pensarlo como una columna más en cada tupla, en cada tabla, donde se coloca el nivel de clasificación del dato: 1, 2, 3, 4 o 5. Cada vez que se hace una operación, el DBMS agrega en la consulta:

*SELECT …. WHERE C= ‘k’ AND CLASE <= acreditación del usuario*

**Concurrencia**

Los DBMS permiten que muchas transacciones accedan a una misma BD a la vez. Para poder mantener la integridad de los datos, se necesita un mecanismo de control de concurrencia para asegurar que las transacciones concurrentes no interfieran entre sí.

Los 3 problemas comunes de la concurrencia son:

* *Actualización perdida*:
  + La transacción A recupera alguna tupla t en el tiempo t1.
  + La transacción B recupera la misma tupla t en el tiempo t2.
  + La transacción A actualiza la tupla t en el tiempo t3.
  + La transacción B actualiza la tupla t, en el tiempo t4.
  + La actualización realizada por la transacción A se pierde en el tiempo t4, ya que la transacción B la sobreescribe.
* *Dependencia no confirmada*:
  + La transacción A ve una actualización no confirmada en el tiempo t2.
  + Esta actualización es posteriormente deshecha en el tiempo t3.
  + La transacción A está operando sobre la suposición falsa de que la tupla t tiene el valor visto en el tiempo t2, siendo que tiene el valor que tenía en el tiempo t1.
  + La transacción A producirá una salida incorrecta.
* *Análisis inconsistente*:
  + La transacción A esta sumando saldos de cuentas.
  + La transacción B esta transfiriendo una cantidad X a la cuenta Y.
  + El resultado producido por A, es obviamente incorrecto, y si A continuara, y escribiera ese dato en la BD, dejaría en efecto, la BD en estado inconsistente.
  + Decimos que A vio un estado inconsistente de la BD, y por lo tanto, ha realizado un análisis inconsistente

Todos los problemas pueden ser resueltos por medio de una técnica de control de concurrencia llamada bloqueo.

**Bloqueo**

Cuando una transacción debe asegurarse de que algún objeto en el que está interesado no cambiará de ninguna forma mientras lo está usando, adquiere un bloqueo sobre ese objeto. De esta forma, la transacción tiene la certeza que el objeto permanecerá estable durante el tiempo que lo desee.

Tipos:

* *Bloqueo Exclusivo o de escritura X*: Si una transacción pone un bloqueo exclusivo sobre la tupla t, entonces se rechazará una petición de cualquier otra transacción para un bloqueo de cualquier tipo sobre la tupla t.
* *Bloqueo compartido o de lectura S*: Si una transacción pone un bloqueo compartido sobre la tupla t, entonces:
  + Se rechazará la petición de cualquier transacción para un bloqueo exclusivo sobre t.
  + Se otorgará la petición de cualquier transacción para un bloqueo compartido sobre t.

Para garantizar que no ocurran los problemas de concurrencia, se tiene un protocolo de acceso a los datos o protocolo de bloqueo:

* Una transacción que desea recuperar una tupla debe primero adquirir un bloqueo S sobre esa tupla.
* Una transacción que desea actualizar una tupla, debe primero adquirir un bloqueo de tipo X sobre esa tupla. Si ya tiene un bloque S sobre la tupla, debe modificar ese bloqueo hacia el nivel X.
* Si una petición de bloqueo es rechazada porque entra en conflicto con un bloqueo ya existente, la transacción pasa a un estado de espera y permanecerá así hasta que la 1º transacción libere el bloqueo.

Resolución de los problemas:

* Problema de actualización perdida: No se soluciona. Hay bloqueo mortal.
* Problema de la dependencia no confirmada: Se soluciona.
* Problema de análisis inconsistente: Se soluciona.

**Bloqueo mortal**

Es una situación donde dos o más transacciones se encuentran en estado simultáneos de espera, cada una esperando que alguna de las demás libere un bloqueo para poder continuar.

La detección de un bloqueo mortal implica la detección de un ciclo en el grafo de espera. La ruptura del bloqueo mortal implica seleccionar una de las transacciones bloqueadas mortalmente como víctima, y entonces deshacerla liberando por lo tanto sus bloqueos y permitiendo que continúen las demás transacciones.

En la práctica no todos los DBMS detectan los bloqueos mortales, usan un mecanismo de tiempos y asumen que una transacción que no ha realizado ningún trabajo durante cierto periodo preestablecido, está bloqueada mortalmente.

**Seriabilidad**

Es el criterio de corrección aceptado comúnmente para la ejecución de un conjunto dado de transacciones. Se considera que la ejecución de un conjunto dado de transacciones es correcta cuando es seriable; cuando produce el mismo resultado que una ejecución serial de las mismas transacciones, ejecutando una a la vez.

* Las transacciones individuales son tomadas como correctas: se da por hecho que transforman un estado correcto de la BD en otro estado correcto.
* También es correcta la ejecución de una transacción a la vez en cualquier orden serial.
* Una ejecución intercalada es correcta cuando equivale a alguna ejecución serial, es decir, es seriable.

Ejemplo:

T1: Sumar $10 al saldo del cliente 1.

T2: Descontar 2% de gastos administrativos del saldo.

T3: Incrementar intereses de la cuenta en un 5%.

Saldo inicial: $200

¿Cuántos resultados consistentes posibles hay?

Solución: 3! = 6. Son todas las combinaciones posibles del conjunto de transacciones.

Todos los resultados son consistentes, pero puede que no sean iguales.

**Niveles de Aislamiento**

Es el grado de interferencia que una transacción dada es capaz de tolerar por parte de transacciones concurrentes.

Si queremos garantizar la seriabilidad, no podemos aceptar ningún tipo de interferencia, es decir, el nivel de aislamiento deber ser el máximo imposible.

En teoría se pueden definir 5 niveles de aislamiento, pero el SQL estándar define solo 4.

**Bloqueo por Aproximación**

Granularidad del bloqueo

Hasta el momento, supusimos bloqueos a nivel de tuplas, pero se pueden bloquear unidades más grandes o más pequeñas. A esto lo llamamos granularidad del bloqueo.

Definimos 3 nuevos niveles de bloque llamados bloqueos por aproximación, que tiene sentido sobre las tablas pero no sobre las tuplas.

* *Bloqueo de aproximación compartida* (IS): bloqueos S sobre tuplas individuales de R.
* *Bloqueo de aproximación exclusiva* (IX): Igual que IS, pero además puede actualizar tuplas individuales, por lo que adquiere bloqueos X sobre esas tuplas.
* *Bloqueo de aproximación compartida exclusiva* (SIX): Combina S e IX, la transacción puede tolerar lectores concurrentes, pero no actualizadores concurrentes en R. Puede actualizar tuplas individuales en R y por lo tanto pondrá bloqueos X sobre esas tuplas.

Antes de que una transacción dada pueda adquirir un bloqueo S sobre una tupla dada deber adquirir un bloqueo IS o una más fuerte sobre la tabla que contiene a esa tupla.

Antes de que una transacción dada pueda adquirir un bloqueo X sobre una tupla dada, primero debe adquirir un bloqueo IX o uno más fuerte sobre la tabla que contiene esa tupla.

**Aislamiento de transacciones**

Los niveles posibles son: READ UNCOMMITTED (leer datos que todavía no se grabaron en la BD), READ COMMITTED (lectura de datos ya actualizados), REPETEABLE READ o SERIALIZABLE.

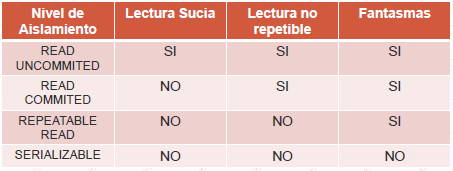
Generalmente, el nivel predeterminado una BD es SERIALIZABLE (el más alto), pero se puede especificar uno menor. Trabaja de manera secuencial por lo que no habría problemas.

Si alguna transacción se efectúa a un nivel menor de aislamiento, la seriabilidad podría ser violada.

Se definen tres formas específicas en las que se puede violar la seriabilidad:

* Lectura sucia: La transacción T1 realiza una actualización sobre alguna fila, la transacción T2 recupera esa fila y la transacción T1 termina con una transacción deshacer. La transacción T2 ha visto una fila que ya no existe, y que en cierto sentido nunca existió.
* Lectura no repetible: T1 recupera una fila, luego T2 actualiza esa fila y después la transacción T1 recupera nuevamente la misma fila. La transacción T1 ha recuperado la misma fila dos veces, pero ve dos valores diferentes en ella.
* Fantasmas: La transacción T1 recupera el conjunto de las filas que satisfacen alguna condición. La transacción T2 inserta entonces una nueva fila que satisface la misma condición. Si la transacción T1 repite ahora su petición de recuperación, vera una fila que antes no existía, un “fantasma”.

**Niveles de Aislamiento de SQL**



Podemos decir que la única forma de evitar los fantasmas es bloquear la ruta de acceso que se utiliza para acceder a los datos en consideración. Este bloqueo puede ser a nivel de un índice.

**Bloqueo Pesimista**

Se adquieren bloqueos sobre entradas y luego mantiene losbloqueos hasta que se realiza la confirmación. Esta estrategia de bloqueo proporciona una mayor coherencia a costa del rendimiento.

Como se bloquean los datos, no se pueden acceder mientras se están haciendo modificaciones.

**Bloqueo Optimista**

Se toma una imagen anterior (copia) de cada registro que toca la transacción y compara la imagen con los valores de entrada actuales cuando se confirma la transacción. Si los valores de entrada cambian, la transacción se retrotrae (sigue insistiendo hasta que pueda ingresar la transacción). No se mantiene ningún bloqueo hasta el momento de la confirmación. Esta estrategia de bloqueo proporciona una mejor concurrencia que la estrategia pesimista, con el riesgo de que la transacción se retrotraiga y el costo de memoria de realizar una copia adicional de la entrada.

No se bloquea, sólo se copia, por lo que se pueden ver los datos mientras se está trabajando sobre la copia. Cuando quiera ingresar los datos, se bloquea.

Bloqueos pesimistas y optimistas

* La gran diferencia es la cantidad de tiempo durante el cual el registro queda bloqueado. En el optimista el tiempo es menor.
* El bloqueo pesimista tiende a saturar el uso del log porque fuerza a la BD a mantener las transacciones abiertas por mucho tiempo.
* En BD donde los accesos de actualización son de menor porcentaje, es preferible el optimista porque el pesimista tiende a arruinar el rendimiento de la BD, a crear situaciones de deadlock (bloqueos mortales) con más frecuencia, a generar filas de espera por la disponibilidad de los datos y a aumentar la competición entre los usuarios.
* Cuanto mayor sea la cantidad de usuarios mayores serán los trastornos que el tratamiento pesimista causará porque a mayor cantidad de bloqueos, mayor cantidad de problemas.

**Cifrado de Datos**

Cuando hablamos de seguridad, también tenemos que tener en cuenta al usuario que trata de causar daño sobre el sistema. La medida más efectiva contra este tipo de amenazas es el cifrado de datos o encriptación, que…

* Un texto plano es cifrado sometiéndolo a un algoritmo de cifrado.
* Los detalles del algoritmo de cifrado son públicos pero la clave se mantiene en secreto.
* El texto cifrado tiene que ser ininteligible para cualquiera que no posea la clave de cifrado, y es lo que se guarda en la BD o se transmite por la línea de comunicación.
* El esquema de cifrado debería ser tal, que el trabajo involucrado en romperlo sobrepase cualquier ventaja potencial que pudiera obtenerse al hacerlo.
* Fundamental en seguridad: El costo de romperla sea significativamente mayor que el beneficio potencial.

Estándar de cifrado de datos

*Sustitución*: se usa una clave de cifrado para determinar para cada carácter del texto plano un carácter de texto cifrado que va a sustituir a ese carácter.

*Permutación*: los caracteres del texto plano son simplemente reorganizados en una secuencia diferente.

Los algoritmos que usan una combinación de ambos, brindan un alto grado de seguridad.

Uno de estos algoritmos es el Estándar de Cifrado de Datos (DES), desarrollado por IBM.

Cifrado de clave pública

En el esquema de clave pública, tanto el algoritmo de cifrado como la clave de cifrado están disponibles y, por lo tanto, cualquier persona puede convertir texto plano en cifrado. La clave de descifrado correspondiente se mantiene en secreto.

Todo esquema de clave pública involucra dos claves, una para cifrado y otra para descifrado.

La clave de descifrado no puede ser deducida de manera realista a partir de la clave de cifrado.

**U3: OPTIMIZACIÓN**

**Optimización**

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

Un aspecto de la optimización de consultas tiene lugar en el nivel del algebra relacional, donde el DBMS 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 de un algoritmo que se utilizará para ejecutar una operación, la selección de índices concretos, que se van a emplear, etc.

La diferencia de costos (en términos de tiempo de evaluación) entre una estrategia buena y una mala suele ser sustancial. Por lo tanto, vale la pena que el sistema pase una importante cantidad de tiempo en la selección de una buena estrategia.

Ejemplo de optimización

Objetivo: obtener el nombre de los proveedores que proporcionan la parte P2.

Supongamos que la BD contiene 100 proveedores y 10000 envíos, de los cuales solo 50 son de la parte P2. Supongamos que las tablas proveedor y envío están representadas en disco como dos archivos guardados por separado.

Si el sistema evalúa la expresión “directamente” (sin optimizar), la secuencia sería:

* *Juntar* proveedor con envío: este proceso implica la lectura de 10000 envíos, la lectura de cada uno de los 100 proveedores 10000 veces (una vez por cada envío) la construcción de 10000 tuplas juntadas y la escritura de 10000 tuplas de nuevo en disco.
* *Restringir* el resultado del anterior paso, solo a las tuplas para la parte P2: involucra la lectura de las 10000 tuplas juntadas hacia memoria, y produce un resultado de 50 tuplas que quedan en memoria principal.
* *Proyectar* el resultado sobre proveedor: esto produce el resultado final deseado.

El siguiente procedimiento es equivalente (produce el mismo resultado) pero es más eficiente:

* *Restringir* los envíos solamente a las tuplas para la parte P2: esto involucra la lectura de 10000 tuplas, pero produce un resultado de 50 tuplas, que pueden mantenerse en memoria principal.
* *Juntar* el resultado del paso anterior, con proveedor: este paso involucra la lectura de 100 proveedores (solo una vez, y no por cada envío de P2) y produce nuevamente un resultado de 50 tuplas.
* *Proyectar* el resultado del paso anterior sobre proveedor.

Si los envíos estuvieran indexados por la parte, sería mucho mejor el desempeño.

**Diccionario o catálogo del optimizador**

nr, el número de tuplas de la relación r.

br, el número de bloques que contiene tuplas de la relación r.

tr, el tamaño de cada tupla de la relacion r en bytes.

fr, el factor de bloqueo de la relación r, es decir, el número de tuplas de la relación r que caben en un bloque.

V(A,r), el número de valores distintos que aparecen en la relación r para el atributo A. Si A es una clave de la relación r, entonces V(A,r) es nr.

La última estadística también se puede calcular para conjuntos de atributos, en vez de solo para atributos aislados.

Las estadísticas sobre los índices, como las alturas de los arboles 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 costos de los planes de evaluación.

Actualización de estadísticas

Si se desean conservar estadísticas precisas, cada vez que se modifica una relación también hay que actualizar las estadísticas. Esto supone una “sobrecarga adicional”.

La mayoría de los DBMS no actualizan las estadísticas con cada modificación, sino en los momentos de menor carga. También se pueden forzar. Puede que el plan que elija no sea el mejor. En consecuencia, puede que las estadísticas utilizadas para escoger una estrategia no sean lo suficientemente exactas.

**Estimación del tamaño de la selección**

La estimación del tamaño del resultado de una operación de selección depende del predicado de la selección.

Si hacemos una selección por igualdad, en la que el atributo A tome el valor a si se supone una distribución uniforme de los valores, se puede estimar que el resultado de la selección sería: tuplas, siendo la cantidad de registros y la cantidad de valores distintos, suponiendo que el valor aparece en el atributo de algún registro de . Pese al hecho de que la suposición de distribución uniforme no suele ser correcta, resulta una aproximación razonable de la realidad.

El diccionario de datos guarda la cantidad de registros diferentes.

Cómo resuelve el optimizador:

Considere una selección donde , si el valor real utilizado en la comparación está disponible en el momento de cálculo de costos, puede hacerse una estimación más precisa. Los valores mínimo y máximo del atributo están almacenados en el catálogo. Suponiendo nuevamente una distribución uniforme, se puede estimar el nº de registros que cumplirán con la condición como si , y como si y como

en cualquier otro caso.

Ejemplo:

Select \* from alumnos where edad <= 20

Catalogo: Edad entre 17 y 30

Alumnos tiene 2000 registros

Cantidad de valores distintos: 14

Cuántos registros devuelve?

Cuántos registros iguales a 20 hay?

Consideraciones:

Cuando la selección involucra una condición simple, lo que se hace es verificar si existe un camino de acceso para el atributo involucrado en la selección y emplear el acceso si existe. Si no, se emplea la fuerza bruta.

Cuando se tienen condiciones de selección del tipo conjuntivas (and), y si hay varios caminos de acceso, el optimizador debe elegir el camino de acceso más directo, es decir que recupera la menor cantidad de registros al menor costo.

En este caso se debe tener en cuenta:

El diccionario de datos también guarda estimaciones de la selectividad.

Cuando se tienen condiciones de selección del tipo disyuntiva (or) son más difíciles de procesar y optimizar.

En este caso si cualquiera de las condiciones no tiene un camino de acceso, lo único que se puede aplicar es la fuerza bruta.

Se puede optimizar sólo si todas las condiciones tienen un camino de acceso.

En algunos casos, cuando la consulta forma parte de un procedimiento almacenado, puede que el valor v no esté disponible cuando se optimice la consulta. En estos casos se supondrá que aproximadamente la mitad de los registros cumplen la condición de comparación. Esta estimación puede resultar muy imprecisa, pero es lo mejor que se puede hacer sin información.

Estimación del tamaño de las reuniones

El producto cartesiano contiene tuplas. Cada tupla de ocupa bytes (tamaño+tamaño), de donde se puede calcular el tamaño del producto cartesiano.

El tamaño de cada tupla también se guarda en el catálogo.

* Si la intersección entre R y S es vacía, es decir las relaciones no tienen atributos en común, entonces se puede usar la técnica de estimación anterior para los productos cartesianos.
* Si la intersección entre R y S es una clave de R, entonces se sabe que cada tupla de se combinará como máximo con una tupla de . Por lo tanto, el nº de tuplas no es mayor que el número de tuplas de S.
* El costo más difícil es que la intersección entre R y S no sea una clave de R ni de S.
* …. Se realizan estimaciones para todas las consultas: proyección, agregación, operaciones de conjunto, etc.

**Procesamiento de consultas del optimizador para optimizar**

1. Convertir la consulta a su forma interna: Implica transformar la consulta original en su representación interna que sea más adecuada para manejarla en la máquina. El procesamiento de vistas también se realiza en esta etapa.

Por lo general, la forma interna seleccionada es algún tipo de árbol de sintaxis abstracto o árbol de consulta. Podemos pensar que esta representación interna representa alguno de los formalismos del algebra relacional o el cálculo relacional.

Una sentencia SQL se descompone en bloques identificando primero expresiones internas…..

1. Convertir a la forma canónica: En esta etapa, el optimizador hace optimizaciones que son garantizadas como buenas, sin tener en cuenta las rutas de acceso físicas (índices) ni los valores actuales de los datos. En este paso, el optimizador convierte la representación interna en alguna forma canónica equivalente, con el objeto de encontrar una representación que sea en cierta forma, más eficiente que la anterior.
2. Seleccionar procedimientos candidatos de bajo nivel: El optimizador debe elegir como ejecutar dicha consulta transformada. Aquí entran en juego consideraciones tales como la existencia de índices u otras rutas de acceso físicas, la distribución de los valores, el agrupamiento físico de los datos almacenados, etc. La estrategia básica consiste en considerar la expresión como la especificación de una serie de operaciones de “bajo nivel” con cierta independencia entre sí. Luego para cada operación de bajo nivel, el implementador tendrá una serie de procedimientos de implementación predefinidos. Cada procedimiento tendrá una fórmula de costo asociada, que indique el costo de ejecutar ese procedimiento.
3. Generar planes de consulta y seleccionar el de menor costo: La última etapa del proceso de optimización involucra la construcción de un conjunto de planes de consulta candidatos, seguidos de una selección del mejor de estos planes. Cada plan se construye por medio de combinación de una serie de procedimientos predefinidos candidatos. La elección del plan de menor costo, requiere de un método para asignar un costo a cualquier plan dado. Este costo es básicamente la suma de los costos de los procedimientos individuales que conforman ese plan. La estimación precisa de los costos puede ser un problema difícil.

Divide y vencerás

Una estrategia para optimizar es la descomposición de la consulta, y es una estrategia muy usada en ambientes distribuidos y de procesamiento en paralelo.

La idea básica de la descomposición de la consulta es dividir una consulta que involucra muchas variables en una secuencia de consultas pequeñas que generalmente involucren cada una, a una o dos de estas variables, usando separación y sustitución de tuplas para lograr esta descomposición.

Factores que influyen en el tunning

*Análisis de consultas y transacciones en la BD*

1° Análisis de consultas

1. Los archivos que se acceden con las consultas
2. Los archivos con los que se realizan las condiciones de selección de las consultas
3. Los atributos con los que se realizan las condiciones de join o las condiciones sobre tablas y objetos múltiples
4. Los atributos que se recuperan con las consultas

Los atributos que se listen en 2 y 3 son candidatos para la definición de estructuras de acceso (índices).

2° Análisis de transacciones de actualización

1. Los archivos a ser actualizados.
2. Tipos de operación sobre cada archivo (insert, update o delete).
3. Los atributos con los que se realizan las condiciones de selección para un delete o update.
4. Los atributos cuyos valores cambian por una operación de actualización.

Los atributos que se listen en 3 son candidatos para la definición de estructuras de acceso, en cambio los que se listan en 4 deben evitarse que formen parte de una estructura de acceso.

Otros:

1. *Análisis de la frecuencia esperada de consultas y transacciones en la BD*: recoger estadísticas y frecuencias de invocación de las consultas y transacciones, por lo general se cumple la regla del 80-20.
2. *Análisis de las restricciones de tiempo de consultas y transacciones en la BD*: identificar las consultas y transacciones que tienen restricciones de tiempo para su ejecución, por ejemplo, transacciones que se debe completar en menos de 5 segundos en el 95% de los casos, y que nunca debe pasar de los 20 segundos.
3. *Análisis de la frecuencia de operaciones de actualización*: un archivo que se actualiza frecuentemente debe tener un mínimo de caminos de acceso.
4. *Análisis de las restricciones de unicidad en los atributos*: es para especificar los caminos de acceso de todos los atributos que son claves primarias o candidatas (condición de unicidad).

Índices

Son fundamentales para operaciones de Join y Selección.

*Cuándo indexar*: El atributo debe ser clave o debe ser de una consulta que se usa el atributo en una condición de selección.

*Qué atributos indexar*: Se puede construir sobre uno o múltiples atributos, en el caso de atributos múltiples, el índice se debe generar con el mismo orden de atributos que cuando se realizan las consultas.

*Cuándo generar un índice-agrupación*: Cuando el archivo es ordenado físicamente con el atributo que se indexa y no es clave. Si una tabla requiere varios índices la decisión es si es necesario un índice de agrupación. S se busca por rango, la generación de este tipo de índices puede ser beneficiosa.

*Desventajas*:

* Son desventaja en aquellas tablas en las que se utiliza frecuentemente operaciones de escritura porque se actualizan cada vez que se modifica una columna.
* En tablas demasiado pequeñas porque no necesitamos ganar tiempo en consultas.
* No son aconsejables cuando pretendemos que la tabla sobre la que se aplica devuelva una gran cantidad de datos en cada consulta.
* Debemos tener en cuenta que ocupan espacio y en algunas ocasiones incluso más espacio que los propios datos.

Tipos de índices: Algo de árbol B, B+ y B\*, índices Hash…

**Buenas prácticas de optimización**

*Buenas prácticas Generales*: recomendaciones de diseño y configuración de carácter general, aplicable a cualquier motor de BD relacional.

*Buenas prácticas Particulares*: recomendaciones únicamente aplicables a un determinado motor de BD particular (ej. MySQL, SQL Server, Oracle, etc.).

Buenas prácticas Generales: Diseño de la BD

* *Normalizar* las tablas, hasta la tercera forma normal, para asegurar que no hay duplicidad de datos y se aprovecha al máximo el almacenamiento en las tablas. Si hay que desnormalizar alguna tabla, se debe pensar en la ocupación y en el rendimiento antes de proceder.
* Los primeros campos de cada tabla deben ser aquellos campos requeridos (no nulos) y dentro de los requeridos primero se definen los de longitud fija y después los de longitud variable.
* Ajusta al máximo el tamaño de los campos para no desperdiciar espacio. (ej: en BD móviles)
* Es muy habitual dejar un campo de texto para observaciones en las tablas. Si a este campo se va a acceder con poca frecuencia o si se ha definido con gran tamaño, es mejor crear una nueva tabla que contenga la clave primaria de la primera y el campo para observaciones.

Buenas prácticas Generales: Gestión y elección de índices

Los índices son campos elegidos arbitrariamente por el constructor de la BD que permiten la búsqueda a partir de dicho campo a una velocidad notablemente superior. Esto se ve contrarrestada por el hecho de ocupar mucha más memoria (el doble más o menos) y de requerir para su inserción y actualización un tiempo de proceso superior.

No podemos indexar todos los campos de una tabla extensa ya que doblamos el tamaño de la BD. Tampoco sirve de mucho el indexar todos los campos en una tabla pequeña ya que las selecciones pueden efectuarse rápidamente de todos modos.

Un caso en el que los índices pueden resultar muy útiles es cuando realizamos peticiones simultáneas sobre varias tablas.

Los índices pueden resultar contraproducentes si los introducimos sobre campos triviales a partir de los cuales no se realiza ningún tipo de petición ya que, además del problema de memoria ya mencionado, estamos ralentizando otras tareas de la BD como son la edición, inserción y borrado.

Entonces ¿Qué indexamos? ….

Buenas prácticas Generales: Donde escribir las sentencias

* Siempre es preferible utilizar consultas almacenadas dentro del motor de BD y no dentro de la aplicación, una consulta almacenada en un procedimiento almacenado, se ejecuta mucho más rápido y directamente sobre el motor que cualquier consulta externa.
* Muchas de las aplicaciones sobre todo de reporting permiten unir y realizar los joins y consultas directamente en la herramienta produciendo una baja de performance realmente considerable. Lo correcto y más eficiente seria generar la consulta en un sp con todos los joins y guardar el resultado en una tabla temporal, de donde posteriormente el reporte solo deberá mostrar los datos sin necesidad de trabajarlos primero. Dependiendo los joins que intervengan y los registros involucrados se pude mejorar drásticamente la performance.

Buenas prácticas Generales: Formulación de las consultas

* No utilizar sin necesidad SELECT \* porque el motor debe leer primero la estructura de la tabla antes de ejecutar la sentencia.
* Seleccionar solo aquellos campos que se necesiten, cada campo extra genera tiempo extra.
* Utilizar *Inner Join*, *left join*, *right join*, para unir las tablas en lugar del where, esto permite que a medida que se declaran las tablas se unan mientras que si utilizamos el where el motor genera primero el producto cartesiano de todos los registros de las tablas para luego filtrar las correctas, un trabajo definitivamente lento.

Buenas prácticas Generales: Refinamiento de las consultas

Los indicadores de que una consulta debe ser refinada son:

* Una consulta realiza demasiados accesos a disco.
* El plan de ejecución muestra que no se emplean índices relevantes.

Problemas típicos:

* Muchos optimizadores no emplean los índices en presencia de cálculos aritméticos, comparación numérica de atributos de diferente tipo y precisión, comparaciones con null o con subcadenas
* Los índices no se emplean para subconsultas que tienen el operador IN
* Algunos Distinct pueden ser redundantes y evitarse sin cambiar el resultado. Puede ocasionar una clasificación del resultado que debe evitarse
* Se debe evitar el uso de tablas resultado temporarias colapsando múltiples consultas en una simple
* El procesamiento de joins se ve afectado por el orden de las tablas en el from.
* Si existen varias opciones de join para una consulta se debe elegir la que usa un índice en lugar de comparaciones de cadenas.
* Algunos optimizadores funcionan peor sobre consultas anidadas comparadas con las no anidadas.
* No se debe abusar del uso de vistas porque a veces es preferible hacer una consulta sobre la tabla base que sobre una vista.
* Una consulta con condiciones múltiples conectadas por operadores OR el optimizador puede no usar el índice.

Para hacer más rápida una consulta se debe intentar las siguientes transformaciones:

* Las condiciones not se deben transformar en condiciones positivas.
* Bloques select anidados usan IN, = ALL, =ANY e =SOME, se deben reemplazar por joins.
* Las condiciones where se pueden reescribir para utilizar índices sobre columnas múltiples.

Optimización de consultas: basado en heurísticas – Árboles y grafos para las consultas

*Consulta*: para todos los proyectos localizados en Santa Fe liste el número del proyecto, el departamento de la empresa que lo ejecuta, el apellido del administrador del departamento, si dirección y fecha de nacimiento.

SEELCT P.numero, D.Dnum, E.Apellido, E.direccion, E.Fnac

FROM Proyecto P, Departamento D, Empleado E

WHERE P.Dnum = D.Dnum AND D.DNI = E.DNI AND P.ubicacion = “Santa Fe”

*Para optimizar*: agregar índice en ubicación, número de departamento y dni de empleado.

Al tener índices, el optimizador se da cuenta de que tiene que empezar por el where y no por el from para ahorrar operaciones.

Los índices son más útiles en las igualdades, en las desigualdades sólo si los registros están ordenados.

El DBMS va a hacer un árbol de la consulta: Primero filtra por ubicación, luego hace el producto cartesiano con la tabla de departamentos y empleados, luego la proyección.

En los árboles, las relaciones de entrada que participan en la consulta son nodos hoja, y las operaciones del álgebra relacional que se ejecutan se representan como nodos internos.

La ejecución del árbol de consulta consiste en efectuar la operación interna de nodo y reemplazar el operando por la relación que resulta de ejecutar toda la operación.

El optimizador, para elegir el mejor plan de ejecución, evalúa cada costo: compara la cantidad de registros que tiene que recorrer.

Buscar transformaciones de una consulta

En SQL [mostrar plan de ejecución] muestra el árbol

Crear índice: create clustered IndiceXNombre on Cliente (nombre\_cliente)

create clustered index: ordena físicamente los registros por ese índice. Útil para desigualdades y agrupamientos.

Buenas prácticas Particulares: MySQL

Especificar el alias de la tabla delante de cada campo definido en el select, esto le ahorra tiempo al motor de tener que buscar a que tabla pertenece el campo especificado.

El orden de ubicación las tablas en el from deberían ir en lo preferible de menor a mayor según el número de registros, de esta manera reducimos la cantidad de revisiones de registros que realiza el motor al unir las tablas a medida que se agregan.

Buenas prácticas Particulares: SQL Server

Si desea guardar objetos grandes (BLOB) en la BD como, por ejemplo: Imágenes, documentos, etc. No lo haga en la misma tabla de la entidad y cree una nueva tabla con el ID y su campo pesado.

Cada tabla soporta un solo índice Clustered ya que es el ordenamiento físico de la misma. Se debe prestar atención a la creación de este tipo de índices ya que puede tener impacto negativo en la performance. Se recomienda que todas las tablas tengan Clustered index ya que las que no lo tienen son menos performantes.

Se recomiendan utilizar el estándar (YYYYMMDD HH:mm:ss) para la definición de las fechas.

Evitar subconsultas en campos del SELECT.

Utilizar BETWEEN en lugar de IN u OR.

Evitar operaciones aritméticas del lado de las columnas del WHERE y ON del JOIN.

Optimizadores en DBMS: MySQL

MySQL Query Esta herramienta gráfica es provista por por MySQL para crear, ejecutar, y optimizar consultas dirigidas a bases de datos MySQL.

myisamchk : Una utilidad para describir, testear, optimizar y reparar tablas. La forma es: myisamchk -r nombre\_tabla

Puede optimizar una tabla de la misma forma usando el comando SQL OPTIMIZE TABLE . Realiza una reparación de la tabla y un análisis de las claves, y también ordena el árbol de índices para obtener un mejor rendimiento en la búsqueda de claves.

SELECT BENCHMARK(loop\_count,expression): devuelve el tiempo en segundos que consume la expresion, la cantidad de veces requeridas. Todas las funciones MySQL pueden ser optimizadas. BENCHMARK() es una herramienta excelente para buscar donde estan los problemas de una query.

EXPLAIN tbl\_name : es un sinonimo DESCRIBE tbl\_name or SHOW COLUMNS

EXPLAIN SELECT select\_options: MySQL explica cómo procesaría el SELECT, proporcionando información acerca de cómo las tablas se unen y en qué orden.

ANALYZE TABLE tt: permite analizar una tabla, e informa errores sobre esta.

Optimizadores en DBMS: SQL Server

SQL Server Management Studio provee una herramienta gráfica para analizar planes de consulta, utilización de índices y costos.

SQL Server Index Tuning Advisor es una herramienta automática para detectar problema en la configuración de índices (índices que falta o que sobran).

SQL Trace: es una herramienta para analizar las sentencias ejecutadas en forma on-line, ver sus costos, sus planes de ejecución, cadenas de bloqueos, deadlock, etc.

Pasos en MySQL para optimizar una consulta WHERE

1. Elimina los paréntesis innecesarios:

((a AND b) AND c OR (((a AND b) AND (c AND d))))

->(a AND b AND c) OR (a AND b AND c AND d)

2. Simplificación constante:

(a<b AND b=c) AND a=5

-> b>5 AND b=c AND a=5

3. Eliminación continua de condiciones (es necesario por la simplificación constante):

(B>=5 AND B=5) OR (B=6 AND 5=5) OR (B=7 AND 5=6)

-> B=5 OR B=6

4. Las expresiones constantes utilizados por los índices se evalúan sólo una vez.

5. COUNT (\*) sobre una sola tabla sin un WHERE se recupera directamente de la tabla de información para tablas MyISAM y HEAP. Esto también se hace para cualquier expresión no NULL cuando se utiliza con una sola tabla.

6. Detección temprana de expresiones constantes inválidas: MySQL detecta rápidamente que algunos comandos SELECT no son posibles y no devuelve ninguna fila.

7. HAVING se fusiona con WHERE si no se utiliza GROUP BY o funciones de grupo (COUNT (), MIN (), y así sucesivamente).

8. Para cada tabla en un join, se construye un WHERE más simple para obtener una rápida evaluación del WHERE de la tabla y también para omitir registros tan pronto como sea posible.

9. Etc …..

**Vistas materializadas**

Cuando se define una vista, normalmente la BD sólo almacena la consulta que define la vista.

Una vista materializada es una vista cuyo contenido se calcula y se almacena. Constituyen datos redundantes, pero resulta mucho más económico en muchos casos leer el contenido de la vista materializada que calcular el contenido de una vista ejecutando la consulta que la define.

Las vistas materializadas resultan importantes para la mejora del rendimiento de algunas aplicaciones.

Un problema de las vistas materializadas es su mantenimiento. Pueden mantenerse mediante código escrito (triggers o procedimientos almacenados).

¿Qué vistas materializar? Esta decisión debe tomarse teniendo en cuenta la carga de trabajo del sistema

**Implementación de los operadores relacionales para resolver producto cartesiano**

A continuación, explicaremos algunos métodos directos para implementar algunos operadores relacionales, especialmente el join. Estos métodos pertenecen al 3º paso en el procesamiento de consultas, que llamamos “procedimientos de implementación de bajo nivel”

Fuerza bruta

Es lo que podría llamarse el caso “llano” donde se inspeccionan todas las combinaciones posibles de tuplas (cada tupla de R es examinada en conjunción con cada tupla de S). Es el más costoso de todos.

Observemos los costos asociados al enfoque de fuerza bruta (mirando solamente el costo de E/S, aunque en realidad sean importantes otros costos): el enfoque requiere claramente un total de operaciones de lectura de tuplas, pero la cantidad de escrituras será distinta según la cardinalidad de la relación:

* En el caso de que un join de muchos a uno (en particular de clave externa con una clave candidata coincidente), es claro que la cardinalidad del resultado es igual a la cardinalidad (m o n) de R o de S, dependiendo del cual representa el lado de la clave externa del join.
* En el caso de muchos a muchos, si suponemos una distribución uniforme de los valores, la cantidad total de tuplas del join sería , donde dCRoS es la cantidad de valores distintos del atributo de junta C en la relación R o S (el menor de los dos).

Este es el procedimiento del peor caso, ya que asume que la relación S no está indexada ni dispersada sobre el atributo de junta C.

También puede aplicarse con índice o con tabla Hash y no sería tan lento.

Búsqueda con índice

Consideremos el caso en el que existe un índice X, sobre el atributo S.C de la relación interna.

La ventaja de esta técnica sobre la fuerza bruta es que para una tupla dada de la relación externa R, se puede ir “directamente” a las tuplas coincidentes de la relación interna S. La cantidad de lecturas de tuplas de las relaciones R y S es simplemente la cardinalidad del resultado juntado.

Búsqueda con dispersión (tabla Hash)

Es similar a la búsqueda con índice, con excepción de que la ruta de acceso rápida a la relación interna S, sobre el atributo de junta S.C, se hace con dispersión en lugar de un índice. Se accede a los datos a través de una estructura hash.

La tabla Hash tiene el resultado de una función que apunta directamente a la tabla de datos.

Mezcla

La técnica de mezcla supone que las dos relaciones R y S están físicamente guardadas en secuencia por los valores del atributo de junta C. Si este es el caso, las dos relaciones pueden ser revisadas en secuencia física, las dos revisiones pueden ser sincronizadas y la unión completa puede ser realizada en una sola pasada sobre los datos.

El agrupamiento físico de los datos relacionados lógicamente es uno de los factores críticos de desempeño.

En ausencia de tal agrupamiento, a menudo es buena idea ordenar una o ambas relaciones en tiempo de ejecución.

Por más que las tablas sean largas, es una lectura rápida.

Dispersión

Al igual que la técnica de mezcla, la técnica de dispersión requiere de una sola pasada sobre cada una de las dos relaciones.

La primera pasada construye una tabla de dispersión para la relación S sobre los atributos de junta S.C; las entradas de esa tabla contienen el valor del atributo de junta y un apuntador hacia la tupla correspondiente en el disco.

La segunda pasada revisa la relación R y aplica la misma función de dispersión al atributo de junta R.C.

La gran ventaja de esta técnica sobra la de mezcla es que las relaciones R y S no necesitan estar guardadas en ningún orden específico y no es necesario ordenamiento.

Es un método lento porque armar la tabla Hash es muy costoso. Se usa generalmente para no usar fuerza bruta sin índice porque es más lento.

**U4: BD DISTRIBUIDAS**

**BD Distribuidas**

Un sistema de BDD consiste en una colección de sitios, conectados por medio de algún tipo de red de comunicación, en el cual:

* Cada sitio es un sistema de BD completo por derecho propio.
* Los sitios han acordado trabajar juntos, a fin que un usuario de cualquier sitio pueda acceder a los datos desde cualquier lugar de la red, exactamente como si los datos estuvieran guardados en el propio sitio del usuario.

Ventajas y desventajas

* Las empresas ya están distribuidas al menos físicamente.
* Eficiencia de procesamiento.
* Mayor accesibilidad. El fallo en un nodo no se nota.
* Complejidad, en cuanto a SW y administración.
* Costo.
* Seguridad.

Principio fundamental: Ante el usuario, un Sistema de BD Distribuido debe lucir exactamente igual que un sistema que no es distribuido.

**12 Reglas u objetivos**

1. Autonomía local: Los sitios deben ser autónomos. Todas las operaciones de ese sitio deben ser administradas por ese sitio. La seguridad, integridad y representación del almacenamiento de los datos locales permanecen bajo el control y jurisdicción del sitio local.
2. No dependencia de un sitio central: Todos los sitios deben ser tratados como iguales. Esta relacionado con el anterior, pero la no dependencia de un sitio central es necesaria por sí misma, aunque no se logra la autonomía local completa.
3. Operación continua: Deben ofrecer mayor confiabilidad y mayor disponibilidad.
   1. La confiabilidad mejora ya que los sistemas distribuidos pueden continuar operando cuando hay alguna falla en algún sitio individual.
   2. La disponibilidad mejora, en parte por la razón anterior y además por la posibilidad de replicación de datos.
4. Independencia de ubicación: Los usuarios no tienen que saber dónde están almacenados físicamente los datos. Permite que los datos emigren de un sitio a otro sin invalidar programas o actividades.
5. Independencia de fragmentación: La fragmentación es necesaria por razones de rendimiento: los datos pueden estar almacenados en la ubicación donde son usados más frecuentemente para que la mayoría de las operaciones sean locales y se reduzca el tráfico en la red. La fragmentación debe hacerse de manera lógica de modo que sea más óptimo.
6. Independencia de replicación: Las réplicas son necesarias por dos razones como mínimo: pueden significar un mejor rendimiento, y pueden significar mejor disponibilidad. La réplica se debe mantener al mínimo. La principal desventaja es la actualización de datos replicados, ya que es necesario actualizar todas las copias: propagación de la actualización. Los usuarios deben ser capaces de comportarse como si los datos no estuvieran replicados.
7. Procesamiento de consultas distribuidas: La optimización en un sistema distribuido es aún más importante que en uno centralizado. En una consulta habrá muchas formas distintas de satisfacer esa solicitud. Por esto, es crucial la optimización en el procesamiento de BDD.
8. Administración de transacciones distribuidas: Dos aspectos fundamentales son: el control de la recuperación y el control de concurrencia. En los sistemas distribuidos una sola transacción puede involucrar actualizaciones en más de un sitio. Cada sitio tiene un “agente” que es el proceso realizado en una transacción. El sistema necesita saber cuándo dos agentes son parte de la misma transacción, ya que, por ejemplo, no debería permitir que dos agentes que forman parte de la misma transacción caigan en un bloqueo mortal entre ellos.
9. Independencia de HW: Existe una necesidad real de poder integrar los datos en máquinas distintas y presentar al usuario una imagen de sistema único. Es necesario tener la posibilidad de ejecutar el mismo DBMS en diferentes plataformas de HW, y hacer que esas máquinas diferentes participen como socios igualitarios en un sistema distribuido.
10. Independencia de SO: Es necesario ejecutar el mismo DBMS en diferentes plataformas de SO.
11. Independencia de red: Si el sistema tiene que soportar sitios distintos, con HW y SW distinto, es obviamente necesario tener la posibilidad de soportar también una variedad de redes de comunicación distintas.
12. Independencia del DBMS: El soporte para la heterogeneidad es definitivamente necesario. El sistema distribuido ideal debe proporcionar independencia de DBMSs.

**BDD Homogéneas y Heterogéneas**

En las BDD homogéneas todos los sitios tienen idéntico software DBMS, conocen los demás sitios o localizaciones y acuerdan cooperar en el procesamiento de las solicitudes de los usuarios.

En las BDD heterogéneas sitios diferentes pueden utilizar esquemas diferentes y DBMS diferentes. Además, pueden no conocer la existencia de los demás. Esto trae aparejado importantes dificultades para el procesamiento de las consultas y también para el procesamiento de transacciones.

**Almacenamiento distribuido de datos**

Si una relación debe almacenarse en una BD, hay dos enfoques del almacenamiento de esta relación en la BDD:

* Réplica: el sistema conserva réplicas (copias) idénticas de la relación y guarda cada réplica en una localización diferente.
* Fragmentación: el sistema divide la relación en varios fragmentos y guarda cada fragmento en un sitio diferente.

Réplica de datos

Si la relación se replica, se guarda una copia de dicha relación en dos o más sitios. En el caso más extremo, se tiene una réplica completa, en la que se guarda una copia en cada sitio del sistema.

* *Disponibilidad*: si alguno de los sitios que contiene la relación falla, la relación puede hallarse en otro sitio distinto, por lo que, el sistema puede seguir procesando las consultas que impliquen a , a pesar del fallo del sitio.
* *Paralelismo incrementado*: en caso de que la mayoría de los accesos a solo resulten en la lectura de la relación, varios sitios pueden procesar en paralelo las lecturas que impliquen a . La réplica de los datos minimiza el movimiento de los datos entre los sitios.
* *Sobrecarga incrementada durante la actualización*: el sistema debe asegurarse que todas las réplicas de , sean consistentes, en caso contrario pueden producirse cálculos incorrectos. Siempre que se actualiza , hay que propagar la actualización a todos los sitios que contienen réplicas. El resultado es una sobrecarga incremental.

En general la réplica mejora el rendimiento de las operaciones leer y aumenta la disponibilidad de los datos para las transacciones de lectura. Las transacciones de actualización suponen una mayor sobrecarga.

Se puede simplificar la gestión de las réplicas de la relación , escogiendo una localización como copia principal de .

Fragmentación de los datos

Si la relación se fragmenta, se divide en varios fragmentos . Estos fragmentos contienen suficiente información para permitir la reconstrucción de r.

Existen dos esquemas diferentes de fragmentación:

* *Fragmentación Horizontal*: la relación r se divide en varios subconjuntos . Cada tupla de la relación debe pertenecer como mínimo a uno de los fragmentos. Esta fragmentación suele utilizarse para conservar las tuplas en los sitios que más se utilizan, minimizando la transferencia de datos. Se tienen todas las columnas de algunos registros particulares.
* *Fragmentación Vertical*: implica la definición de varios subconjuntos de atributos del esquema . Una forma de asegurar que la relación pueda reconstruirse es incluir los atributos de la clave principal de R en cada fragmento . Se tienen todos los registros, pero sólo algunas columnas.

Almacenamiento

Se pueden aplicar los dos tipos de fragmentación a un solo esquema, y los fragmentos también pueden replicarse.

En general, podemos decir, que los fragmentos pueden replicarse, las réplicas de los fragmentos pueden fragmentarse, etc.

**Transparencia**

Los usuarios de BDD no deben conocer la ubicación física de los datos, ni el modo en que se puede tener acceso a ellos en un sitio concreto.

* Transparencia de la fragmentación: no se exige que conozca la forma en que se fragmento la relación.
* Transparencia de la réplica: los usuarios ven cada objeto de datos como único.
* Transparencia de la ubicación: El sistema de BDD debe poder hallar los datos siempre que la transacción del usuario facilite el identificador de los datos.

Los elementos de datos (relaciones, fragmentos, réplicas) deben tener nombres únicos. En las BDD, hay que tener cuidado para asegurarse de que dos sitios no utilicen el mismo nombre para elementos diferentes.

Una solución es tener un servidor de nombres central. Los inconvenientes son que el servidor de nombres se transforme el cuello de botella, y si queda fuera de servicio, puede que no siga funcionando ningún otro sitio del SBDD.

Otra solución es que cada sitio anteponga su propio identificador de sitio a cualquier nombre que genere. Pero no logra la transparencia de la ubicación.

Para superar este último problema, el SBD puede crear un conjunto de nombres alternativos o alias para elementos de datos.

**Asignación de Fragmentos**

Seguido a la decisión de la fragmentación sigue la asignación de los fragmentos y la cantidad de réplicas a generar.

Parámetro para la generación de réplicas:

* Cantidad de lecturas
* Cantidad de escrituras

Comparación de las alternativas de réplica

|  |  |  |  |
| --- | --- | --- | --- |
|  | Réplica Total | Réplica Parcial | Partición |
| Procesamiento de consultas | Fácil | Difícil | Difícil |
| Control de concurrencia | Moderado | Difícil | Fácil |
| Confiabilidad | Muy alta | Alta | Baja |
| Realidad | Posible aplicación | Realista | Posible aplicación |

**Transacciones Distribuidas**

El acceso a los diferentes elementos de datos en los sistemas distribuidos suele realizarse mediante transacciones, que deben preservar las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad).

Hay dos tipos de transacciones que se deben considerar:

* Transacciones locales: acceden y actualizan en una BD local.
* Transacciones globales: acceden y actualizan datos en varias BD locales.

Asegurar las propiedades ACID en transacciones globales es una tarea complicada, dado que participan en la ejecución varios sitios. El fallo de alguno de los sitios, o el de un enlace de comunicación puede dar lugar a cálculos erróneos.

**Estructura del SGBDD**

Cada sitio tiene su propio gestor local de transacciones que asegura las propiedades ACID de las transacciones que se ejecutan en ese sitio.

Cada gestor de transacciones colabora para ejecutar las transacciones globales.

Cada gestor contiene: el gestor de transacciones, y el coordinador de transacciones que coordina la ejecución de las diferentes transacciones (tanto locales como globales) iniciadas en ese sitio.

Coordinador de transacciones

Es responsable de:

* Inicio de la ejecución de la transacción.
* División de la transacción en varias subtransacciones y distribución de esas subtransacciones a los sitios correspondientes para su ejecución.
* Coordinación de la terminación de la transacción, que puede hacer que la transacción se comprometa en todos los sitios o que se aborte en todos los sitios.

**Modos de fallos en SGBDD**

Los sistemas distribuidos pueden sufrir los mismos tipos de fallos que los sistemas centralizados: errores de software, errores de hardware, fallos de discos, pero hay más fallos propios de entornos distribuidos:

* Fallo de un sitio.
* Pérdida de mensajes entre nodos.
* Fallo de un enlace de comunicación.
* División de la red. (grupos de nodos conectados por separado)

**Protocolos de compromiso**

Si hay que asegurar la atomicidad, todos los sitios en los que se ejecute una transacción T deben coincidir en el resultado final de la ejecución. T debe comprometerse en todos los sitios o abortarse en todos los sitios.

Para asegurar esta propiedad el coordinador de transacciones de T debe ejecutar un protocolo de compromiso.

**Compromiso de dos fases (C2F)**

Al comprometer un dato, se aplica bloqueo.

Considere una transacción iniciada en el sitio , en que el coordinador de transacciones es . Cuando completa su ejecución (cuando todos los sitios donde se ha ejecutado informan a que se ha completado), inicia el protocolo C2F:

* Fase 1: añade al registro al registro histórico y obliga a guardar el registro histórico en un lugar de almacenamiento estable (Log). Entonces envía un mensaje a todos los sitios donde se ha ejecutado . El gestor de transacciones del sitio determina si desea comprometer su parte de . Si la respuesta es negativa, añade un registro al registro histórico y responde enviando a el mensaje de abortar . Si la respuesta es sí, añade un registro al registro histórico y obliga a que el registro histórico se guarde en un almacenamiento estable. El gestor de transacciones contesta con el mensaje .
* Fase 2: Cuando recibe las respuestas al mensaje de preparar de todos los sitios, o cuando ha transcurrido un intervalo de tiempo especificado, puede determinar si la transacción puede comprometerse o abortarse. En función del resultado, se añade al registro histórico un registro o y se obliga a que el registro se guarde en almacenamiento estable. A partir de este momento, el coordinador envía un mensaje o a todos los sitios participantes. Cuando un sitio recibe este mensaje, lo guarda en su registro histórico.

Trato de Fallos

* Fallo de un sitio participante: Si el coordinador detecta que un sitio ha fallado emprende las acciones siguientes. Si falla antes de responder con el mensaje , el coordinador asume que ha respondido con el mensaje . Si el sitio falla después que el coordinador haya recibido del sitio el mensaje , el coordinador ejecuta el resto del protocolo ignorando el fallo del sitio.
* Cuando un sitio se recupera de un fallo debe examinar su registro histórico para determinar el destino de las transacciones que se hallaban en ejecución cuando se produjo el fallo. Si T es una de esas transacciones, se toman en consideración cada uno de los casos posibles. (, )
* Fallo del coordinador: si el coordinador falla durante la ejecución del protocolo de compromiso para la transacción , los sitios deben decidir el destino de . En ciertos casos los sitios participantes no pueden decidir si comprometer o abortar , y deben esperar a la recuperación del coordinador que ha fallado. Esta situación puede hacer que los elementos de datos no estén disponibles, no solo en el sitio que fallo, sino en sitios activos, por bloqueos, ya que queda bloqueada mientras espera la recuperación del sitio coordinador.
* División de la red: cuando una red queda dividida hay dos posibilidades:
  + El coordinador y los sitios participantes siguen en una partición, por lo que el fallo no tiene ningún efecto sobre el protocolo de compromiso.
  + El coordinador y los participantes quedan en varias particiones. Esto se maneja como si los sitios que están en una partición distinta al coordinador hubieran fallado.

**Compromiso de 3 Fases**

Es una extensión del C2F que evita el problema del bloqueo con determinadas suposiciones.

Se supone que no se producen fragmentaciones de la red y que no fallan más de sitios, donde es un numero predeterminado.

El protocolo evita el bloqueo introduciendo una tercera fase adicional en que se implican varios sitios en la decisión de comprometer.

Antes de anotar la decisión de comprometer en su almacenamiento persistente, el coordinador se asegura antes que al menos sitios sepan que pretende comprometer la transacción.

Si el coordinador falla, los sitios seleccionan un nuevo coordinador, este nuevo coordinador verifica el estado del protocolo a partir de los demás sitios, si el coordinador había decidido comprometer, al menos uno de los sitios restantes a los que informo estará funcionando y asegurará que se respete la decisión.

El nuevo coordinador vuelve a iniciar la 3 fase del protocolo, ya sea comprometiendo o abortando.

Este protocolo se usa muy poco, debido a la gran sobrecarga. Tiene la propiedad de no bloquearse, a menos que fallen sitios.

**Sistemas Federados**

Problemática: Una consulta cuya respuesta requiere acceder a diversas BD. Estas BD tienen diferentes arquitecturas, y diferentes DBMS.

Se trata de poder formular una sola consulta y recibir una única respuesta de modo de que en la preparación de la respuesta intervienen datos de varias BD.

Ejemplos:

* Dos empresas, cada una con sus bases de datos, que se fusionan o pasan a formar parte de un mismo holding (unión que puede ser temporaria o no, pero las BD siguen teniendo autonomía).
* Ministerios que quieren compartir sus datos.
* Provincias o territorios autónomos que desean acceder mutuamente a ciertos datos
* Países de un mercado común que necesitan intercambiar datos.

Soluciones: Se debe seleccionar la mejor

* Integración manual: Consultar separadamente cada BD, e integrar manualmente las respuestas.
* Integración de datos: Crear una nueva BD que integre todos los datos de las preexistentes.
* Integración del acceso: Construir un Sistema Federado en el que las bases de datos interoperen.
* Crear un Data Warehouse.

Consultas separadas e integración manual

Las personas que lo realice deben:

* Saber qué bases de datos están accesibles
* Saber qué datos hay en cada BD
* Saber descomponer la consulta en las consultas parciales a cada BD
* Conocer el modelo de cada BD
* Conocer el lenguaje de cada BD
* Saber integrar los resultados parciales para producir el resultado deseado

*Viable sólo muy excepcionalmente.*

Integración de datos

* Diseñar la nueva BD (distribuida?)
* Convertir los programas
* Transferir los datos de las bases de datos preexistentes a la nueva
* Preparar y enseñar los nuevos modos de trabajar de los usuarios

*La gestión autónoma de cada BD se pierde, en aras de una gestión única.*

Acceso integrado

* Superponer un sistema nuevo sobre los DBMSs de las bases de datos preexistentes.
* El nuevo sistema acepta la consulta y devuelve la respuesta, generando internamente las consultas parciales e integrando sus respuestas.
* La existencia de múltiples bases de datos puede ser transparente al usuario.
* Los programas y usuarios preexistentes no se ven afectados.

*Se preserva la autonomía de cada BD.*

Autonomía y Flexibilidad

Una de las principales diferencias entre las soluciones radica en el grado en que se mantiene la autonomía de las bases de datos preexistentes:

* Se pierde totalmente en la integración de datos en una BDD
* Se preserva totalmente en la integración manual
* Se obtienen grados intermedios en bases de datos federadas

Se pueden obtener las mismas conclusiones respecto a la flexibilidad de añadir más Bases de Datos

Cualquiera de las soluciones debe superar dificultades técnicas importantes

En muchos casos es preferible el acceso integrado, por razones de viabilidad, preservación de autonomía, flexibilidad y evolución.

*No se integran los datos, sino el acceso a los datos*.

**Sistemas de BD Federados**

Dos niveles:

* El de las bases de datos preexistentes, que denominaremos bases de datos componentes: NIVEL COMPONENTE
* El del conjunto de bases de datos que interoperan, que llamaremos NIVEL FEDERADO

Los sistemas de bases de datos componentes se FEDERAN para dar lugar a un Sistema de Bases de Datos Federado (FDBS)

FDBS consiste en una colección de Sistemas de Bases de Datos componentes:

* Cooperantes y autónomos
* Interoperando según:
  + Las necesidades de los usuarios de la federación
  + El deseo de los administradores de las bases de datos en participar y compartir sus datos
* Manipulados y controlados de manera coordinada por un software llamado Sistema de Gestión de Bases de Datos Federadas (FDBMS)

Características de los SBDF

* Autonomía y heterogeneidad.
* El sistema es distribuido.
* No hay un esquema conceptual global único, común a toda la federación.
* Las federaciones pueden formarse y desaparecer.
* Sistemas de bases de datos pueden entrar y salir de una federación.
* Un sistema componente puede ser distribuido o a su vez otro FDBS.
* Un sistema de BD puede ser componente de varias federaciones al mismo tiempo.

Autonomía

Los SBD estaban normalmente controlados de forma separada e independiente. Se comparten datos únicamente si se retiene el control. La autonomía de cada componente puede ser de: diseño, comunicaciones, ejecución, asociación.

Diseño:

* Principal fuente de heterogeneidad
* Capacidad para elegir su propio diseño en todos sentidos:
  + Universo de discurso: qué
  + Conceptualización: qué conceptos
  + Representación (modelo, lenguaje) y nombres: cómo
  + Comportamiento (operaciones, restricciones)
  + Implementación: hard, soft, SGBD, esquema interno, etc.

Comunicación:

Capacidad de un SGBD para decidir:

* Si acepta mensajes de otros sistemas componentes
* Cómo y cuándo responder a las solicitudes de otros sistemas

Ejecución:

Capacidad de cada BD componente para:

* Decidir el orden en que son ejecutadas las operaciones
* Ejecutar operaciones sin interferencia de operaciones externas
* Abortar cualquier operación que no cumpla con las restricciones propias
* Que las operaciones a nivel componente no se afecten lógicamente con la participación en un FDBS
* No tener que informar a un sistema externo el orden de ejecución

Los SGBD componentes tratan a las operaciones externas de la misma forma que a las operaciones propias: no siempre pueden distinguirlas.

Asociación:

Capacidad de un SGBD para decidir

* Con quién compartir
* Qué compartir
* Cómo compartir
* Si asociarse / desasociarse de una federación

Puede considerarse parte de la autonomía de diseño.

**Semejanzas y diferencias entre un SBDF y SBDD**

Semejanzas:

* Hay datos en varias instalaciones
* Las instalaciones están interconectadas: sistema distribuido
* Hay dos niveles
* Se formula una pregunta y se obtiene una respuesta consistente.

Diferencias:

* De diseño (top-down vs. bottom-up). Las DBB van de lo más general a los más específico. En las BDF es al revés.
* De terminología (global/local – federado/componente)
* De autonomía. En las BDF el diseño es uno.
* De transparencias. En BDF se tiene que saber qué comparte cada uno de las BD
* De arquitectura. BBD nodos de igual jerarquía.

En FDBS la “distribución” es una consecuencia de la autonomía.

**Construcción de un SBDF**

El proceso de negociación:

Hay dos casos:

* Para formar una federación nueva
* Para incorporar una BD componente a un FDBS existente

El administrador de la BD componente cede acceso a datos

* A cambio de poder acceder a otros datos
* Siguiendo órdenes de rango superior

La BD pierde autonomía: debe obedecer protocolos

* Respecto a modificar su esquema nativo
* Respecto a abandonar el FDBS

Si el FDBS es fuertemente acoplado, la negociación es más compleja

La complejidad de los sistemas de seguridad de las BD plantea dificultades

Integración de esquemas:

Proceso de integración:

* Seleccionar un conjunto de esquemas de exportación a integrar
* Integrarlos para producir un esquema federado
* Generar las correspondencias entre esquemas (directorio)
* Desarrollar el procesador de construcción:
  + Operaciones sobre el esq. Federado generan operaciones sobre los esq. Exportación y viceversa.

A partir de aquí se derivan los esquemas externos de acuerdo con las aplicaciones que van a interactuar con los mismos

**Motivación BD Distribuidas**

Las organizaciones necesitan acceder a los datos que se encuentra en diferentes sitios. Los avances en las tecnologías de comunicación que permiten un flujo constante entre diferentes sitios. La creciente globalización de las empresas y la necesidad de compartir información con clientes y proveedores.

Antes había problemas en las redes de comunicaciones por lo que la BD podía quedar inaccesible. Ahora con la mejora de las redes de comunicación se prefiere cliente-servidor con BD centralizada y procesamiento distribuido. Ahora se utiliza réplica en un mismo data center, porque en distribuido trae problemas.

Ventajas

* Las empresas ya están distribuidas al menos físicamente y ven como algo natural el enfoque de BDD que refleja la estructura organizacional.
* El uso más frecuente de los datos almacenados localmente, brinda una mayor eficiencia en las transacciones.
* Crecimiento proporcional, cuando se suman nuevas sucursales o sedes de las organizaciones, las BDD proporcionan una solución con mínimo impacto.
* Se reduce la sobrecarga de las comunicaciones, ya que mayormente, una alta proporción de las transacciones se resuelve de manera local.
* Mejora el rendimiento, teniendo un alto grado de paralelismo, la carga es distribuida entre los diferentes nodos/sitios.
* Mayor confiabilidad y disponibilidad, al tener redundancia de datos, lo que brinda una alta disponibilidad de los sistemas. Esto requiere además mayores controles.

**Comparación entre BDC y BDD**

|  |  |  |
| --- | --- | --- |
|  | **BD CENTRALIZADAS** | **BD DISTRIBUIDAS** |
| **Definición** | Los sistemas de DBC son aquellos que se ejecutan en un único Software de Administración de BD.  Son accedidos desde diferente terminales o sistemas, pero todas las transacciones se resuelven sobre ese único Software de administración. | Una BDD es un conjunto de múltiples BD lógicamente relacionadas, las cuales se encuentran distribuidas en diferentes espacios lógicos e interconectados por una red de comunicaciones. Tienen la capacidad de realizar procesamiento autónomo, esto permite realizar operaciones locales o distribuidas. Un SBDD es un sistema en el cual múltiples sitios de BD están ligados por un sistema de comunicaciones de tal forma que, un usuario en cualquier sitio puede acceder los datos en cualquier parte de la red exactamente como si estos fueran accedidos de forma local. |
| **Ventajas** | Se evita la redundancia: En sistemas que no usan BDC, cada aplicación tiene sus propios archivos privados o se encuentran en diferentes localidades. Esto origina enorme redundancia en los datos almacenados, así como desperdicio del espacio de almacenamiento.  Se evita la inconsistencia: Ya que, si un hecho específico se representa por una sola entrada, la no concordancia de datos no puede ocurrir. Pueden hacerse cumplir las normas establecidas. Con un control central de la BD, el Administrador de BD (DBA) puede garantizar que se cumplan todas las formas aplicables a la representación de los datos.  Pueden aplicarse restricciones de seguridad. El DBA puede:   * Asegurar que el único medio de acceder la BD sea a través de los canales establecidos. * Definir controles de autorización para que se apliquen cada vez que se intente el acceso a datos sensibles. | Autonomía local: un sitio puede controlar los datos que le pertenecen.  Disponibilidad: un fallo en una parte del sistema solo afectará a un fragmento, en lugar de a toda la BD.  Rendimiento: los datos generalmente se ubican cerca del sitio con mayor demanda, también los sistemas trabajan en paralelo, lo cual permite balancear la carga en los servidores.  Economía: es más barato crear una red de muchas computadoras pequeñas, que tener una sola computadora muy poderosa.  Modularidad: se pueden modificar, agregar o quitar sistemas de la BD distribuida sin afectar a los demás sistemas (módulos). |
| **Desventajas** | * Los mainframes (computadora central) no ofrecen mejor proporción precio/rendimiento que los microprocesadores de los sistemas distribuidos. * Cuando un sistema de BDC falla, se pierde toda la disponibilidad de procesamiento y sobre todo de la información confiada al sistema. * Las cargas de trabajo no se pueden difundir entre diferentes computadoras, ya que los trabajos siempre se ejecutarán en la misma máquina. | * Costo de desarrollo del SW: Es complejo mantener la coordinación entre nodos. * Mayor probabilidad de errores: Como los nodos del sistema funcionan en paralelo, es más difícil asegurar el funcionamiento correcto de los algoritmos, así como de los procedimientos de recuperación de fallos del sistema. * Mayor sobrecarga de procesamiento: El intercambio de mensajes y ejecución de algoritmos para el mantenimiento de la coordinación entre nodos supone una sobrecarga que no se da en los Sistemas de BDC. |

Estructura cliente-servidor: es en parte centralizado y en parte distribuido. La BD está en un solo servidor pero el procesamiento se puede hacer en diferentes servidores ya que la lógica de negocio está en una capa por arriba de la BD (aplicación).

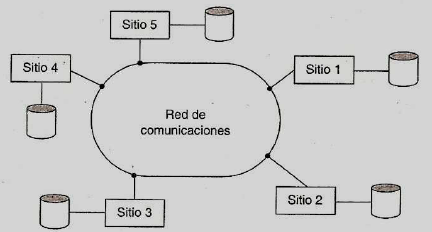
**Tareas de Administración de una BDD**

Actividades típicas de toda BD: Seguridad, Integridad, Optimización, etc.

Actividades especificas de una BDD:

* 1. Descripción del esquema global.
  2. Descripción de la fragmentación/replicación.
  3. Descripción de las localizaciones.
  4. Estadísticas de la BD. (para optimizar consultas)

**Arquitectura de una BDD**



**Pasos a seguir para diseñar una BDD**

1. Diseño del "*esquema conceptual*" el cual describe la BD integrada (todos los datos que son utilizados por las aplicaciones que tienen acceso a las BD).
2. Diseño "*físico*" de la BD: mapear el esquema conceptual a las áreas de almacenamiento y determinar los métodos de acceso.
3. Diseño de la *fragmentación* lógica: determinar la forma en que las relaciones globales se subdividen en fragmentos horizontales, verticales o mixtos.
4. Diseño de la *asignación de los fragmentos*: determinar la forma en que los fragmentos se mapean a las imágenes físicas.

**Procesamiento distribuido de consultas**

* NODO1: EMPLEADO: 1.000.000 bytes
* NODO2: DEPARTAMENTO: 3500 bytes

Caso 1:

*“Por cada empleado, obtener el nombre del empleado y el nombre del departamento al que pertenece”*

: ΠNombre,Apellido,NombreDPto(EMPLEADO ⋈ DEPARTAMENTO)

▪ La consulta se lanza desde el nodo3 (nodo respuesta) que no tiene datos implicados en la consulta.

▪ El resultado de ésta consulta constará de 10.000 tuplas. Cada tupla resultante será de una longitud de 40 bytes. El tamaño del resultado será por tanto de 400.000 bytes.

▪ Existen tres alternativas para resolver la consulta:

* Primera alternativa: Transferir, tanto la relación EMPLEADO, como la relación DEPARTAMENTO al nodo respuesta (nodo3) y realizar allí mismo la operación de join. En éste caso se transfieren: 1.000.000 + 3.500 = 1.003.500 bytes.
* Segunda alternativa: Transferir la relación EMPLEADO al nodo2, ejecutar el join en este nodo y enviar el resultado al nodo3. Esto implicaría transferir: 1.000.000 + 400.000 (resultado) = 1.400.000 bytes
* Tercera alternativa: Transferir la relación DEPARTAMENTO al nodo1, ejecutar el join en este nodo y enviar el resultado al nodo3. En este caso, los bytes transferidos serán: 3.500 + 400.000 (resultado) = 403.500 bytes.

Caso 2:

*“Para cada departamento, obtener el nombre del departamento y el de su director”*

La consulta se lanza desde el nodo3. El resultado de ésta consulta constará de 100 tuplas (4.000 bytes).

* Opción 1: transferimos las relaciones DEPARTAMENTO y EMPLEADO al nodo3. Se transfieren: 3.500 + 1.000.000 = 1.003.500 bytes.
* Opción 2: transferimos la relación EMPLEADO al nodo2 y enviamos el resultado del join al nodo3. Se transfieren: 1.000.000 + 4.000 = 1.004.000 bytes.
* Opción 3: transferimos la relación DEPARTAMENTO al nodo1 y enviamos el resultado del join al nodo3. Se transfieren: 3.500 + 4.000 = 7.500 bytes.

**Procesamiento distribuido de consultas utilizando semijoin**

Reducción del número de columnas antes de transferir a otro nodo.

* Se envía la columna con la que se va a realizar el join de una relación R al nodo donde se encuentra la otra relación, allí se realiza el join con la otra relación S.
* Se envían las columnas implicadas en el resultado al nodo inicial y se vuelve a realizar el join con R.
* Sólo se transfieren las columnas de R que intervienen en la realización del join en una dirección y el subconjunto de columnas de S resultantes en la otra.

**U5: BD ORIENTADA A OBJETOS**

Problemática

Los modelos de bases de datos tradicionales han sido capaces de satisfacer con éxito las necesidades, en cuanto a bases de datos, de las aplicaciones de gestión tradicionales. Sin embargo, presentan algunas deficiencias cuando se trata de aplicaciones más complejas o sofisticadas como, por ejemplo, el diseño y fabricación en ingeniería (CAD/CAM, CIM), los experimentos científicos, los sistemas de información geográfica o los sistemas multimedia. Los requerimientos y las características de estas nuevas aplicaciones difieren en gran medida de las típicas aplicaciones de gestión: la estructura de los objetos es más compleja, las transacciones son de larga duración, se necesitan nuevos tipos de datos para almacenar imágenes y textos, y hace falta definir operaciones no estándar, específicas para cada aplicación.

Desarrollo de aplicaciones OO

* Lenguajes OO poseen aceptación en la práctica.
* La programación en el paradigma OO sobre DBMS relacionales requiere de un mapeo entre los paradigmas o materialización de objetos a partir de la BD o desmaterialización hacia la BD o manejo de referencias entre objetos.

Las BDOO se crearon para tratar de satisfacer las necesidades de estas nuevas aplicaciones. La orientación a objetos ofrece flexibilidad para manejar algunos de estos requisitos y no está limitada por los tipos de datos y los lenguajes de consulta de los sistemas de bases de datos tradicionales. Una característica clave de las bases de datos orientadas a objetos es la potencia que proporcionan al diseñador al permitirle especificar tanto la estructura de objetos complejos, como las operaciones que se pueden aplicar sobre dichos objetos.

Ejemplos de ODBMS

* ObjectStore (Adquirida por Progress)
* Objectivity/DB (Objectivity) http://www.objectivity.com/
* ONTOS (Ontologic)
* VERSANT (Versant Object Technology) http://www.versant.com/
* GemStone (GemStone) http://www.gemstone.com/
* Jasmine (CA) http://www.cai.com/
* DB4Object
* Eloquera

**ODMG (Object Database Management Group)**

Durante los últimos años se han creado muchos prototipos experimentales de sistemas de BDOO y también muchos sistemas comerciales. Conforme éstos fueron apareciendo, surgió la necesidad de establecer un modelo estándar y un lenguaje. Para ello, los fabricantes de los SGBD OO formaron un grupo denominado ODMG (Object Database Management Group), que propuso el estandar ODMG-93 y que ha ido evolucionando hasta el ODMG 3.0. El uso de estándares proporciona portabilidad, permitiendo que una aplicación se pueda ejecutar sobre sistemas distintos con mínimas modificaciones. Los estándares también proporcionan interoperabilidad, permitiendo que una aplicación pueda acceder a varios sistemas diferentes. Y una tercera ventaja de los estándares es que permiten que los usuarios puedan comparar entre distintos sistemas comerciales, dependiendo de qué partes del estándar proporcionan.

Formas Básicas de proveer OO

* ODBMS: son SGBD basados completamente en el modelo OO.
* ORDBMS: son SGBD relacionales que permiten almacenar objetos en sus relaciones (tablas). Ej.: Oracle
* Software para transformación OO-Relacional.

**Conceptos básicos de OO**

Tradicionalmente, los datos y los procedimientos se han almacenado separadamente: los datos y sus relaciones en la BD y los procedimientos en los programas de aplicación. La OO, combina los procedimientos de una entidad con sus datos. (Encapsulamiento)

Las entidades son unidades autocontenidas que se pueden reutilizar con relativa facilidad. En lugar de ligar el comportamiento de una entidad a un programa de aplicación, el comportamiento es parte de la entidad en sí, por lo que en cualquier lugar en el que se utilice la entidad, se comporta de un modo predecible y conocido.

El modelo OO también soporta relaciones de muchos a muchos, siendo el primer modelo que lo permite.

Objetos

* Es un elemento autocontenido utilizado por el programa. Los valores que almacena un objeto se denominan atributos, variables o propiedades. Los objetos pueden realizar acciones, que se denominan métodos, servicios, funciones, procedimientos u operaciones.
* Los objetos tienen un gran sentido de la privacidad, por lo que sólo dan información sobre sí mismos a través de los métodos que poseen para compartir su información. También ocultan la implementación de sus procedimientos, aunque es muy sencillo pedirles que los ejecuten. Los usuarios y los programas de aplicación no pueden ver qué hay dentro de los métodos, sólo pueden ver los resultados de ejecutarlos. (Encapsulamiento de datos).
* Cada objeto presenta una interface pública al resto de objetos que pueden utilizarlo. Una de las mayores ventajas del encapsulamiento es que mientras que la interface pública sea la misma, se puede cambiar la implementación de los métodos sin que sea necesario informar al resto de objetos que los utilizan. Para pedir datos a un objeto o que éste realice una acción se le debe enviar un mensaje.
* Un programa OO es un conjunto de objetos que tienen atributos y métodos. Los objetos interactúan enviándose mensajes. La clave es averiguar qué objetos necesita el programa y cuáles deben ser sus atributos y sus métodos.
* OID: identidad única de cada objeto independiente almacenado en la BD. El OID es inmutable, es decir, el valor no cambia. Cada OID es único para cada objeto, si el objeto es removido se lleva el OID con él y ningún otro objeto lo puede usar.
* El objeto se identifica con un triple (i, c, v)
  + i = OID
  + c = constructor de tipo
  + v = valor corriente que tiene el objeto
  + El modelo puede incluir varios constructores de tipo:
    - átomo, tupla, set, lista, bag, array
    - set/bag: conjunto de objetos sin orden
    - lista: conjunto de objetos con orden sin tamaño definido
    - array: conjunto de objetos con orden con tamaño definido
    - tupla: tipo estructurado (struct de C ó C++)

Clase

* Es un patrón o plantilla en la que se basan objetos que son similares (abstracción del objeto). Cuando un programa crea un objeto de una clase, proporciona datos para sus variables y el objeto puede entonces utilizar los métodos que se han escrito para la clase. Todos los objetos creados a partir de la misma clase comparten los mismos procedimientos para sus métodos, también tienen los mismos tipos para sus datos, pero los valores pueden diferir.
* Una clase también es un tipo de datos. De hecho, una clase es una implementación de lo que se conoce como un tipo abstracto de datos. El que una clase sea también un tipo de datos significa que una clase se puede utilizar como tipo de datos de un atributo.

Tipos de métodos

Hay varios tipos de métodos que son comunes a la mayoría de las clases:

* *Constructor*: Tiene el mismo nombre que la clase. Se ejecuta cuando se crea un objeto de una clase. Por lo tanto, un constructor contiene instrucciones para inicializar las variables de un objeto.
* *Destructor*: Se utiliza para destruir un objeto. No todos los lenguajes OO poseen destructores.
* *Accesor*: Devuelve el valor de un atributo privado de otro objeto. Así es cómo los objetos externos pueden acceder a los datos encapsulados.
* *Mutador*: Almacena un nuevo valor en un atributo. De este modo es cómo objetos externos pueden modificar los datos encapsulados.

Herencia

* + - Simple: un supertipo puede ser heredado por uno o más subtipos.
    - Múltiple: un subtipo puede heredar de más de un supertipo.
    - Selectiva: un subtipo hereda solo una parte del supertipo (cláusula EXCEPT).

Polimorfismo

* + - Sobrecarga de funciones y operadores.

Versiones y configuraciones

* + - En algunas aplicaciones se necesitan generar nuevas versiones de algunos objetos sin descartar los existentes (diseño). Muchos ODBMS tienen algunos sistemas que facilitan la generación de versiones del objeto.

**Persistencia de Objetos**

*Cuestión*: ¿Cómo proveer persistencia a una aplicación?

*Problemas en la implementación de persistencia*:

* + - ¿Cómo representar objetos que pueden ser removidos y objetos que deben ser almacenados?
    - ¿Cómo acceder a objetos eficientemente?
    - ¿Dónde almacenar el código de una aplicación?
    - ¿Cómo excluir objetos?

Objeto persistente: Objeto que sobrevive a la ejecución del proceso que lo creó.

Objetotransiente: Objeto que deja de existir cuando el proceso que lo creó termina su ejecución.

LPOO: Concebido para manipular datos transientes.

ODBMS: Concebido para manipular datos persistentes.

**Persistencia explicita e inferida**

Persistencia explícita: el programador explícitamente informa al sistema cuáles objetos son persistentes por medio de un nombre (mecanismo de nominación).

Persistencia inferida: el sistema infiere cuáles objetos son persistentes, por el hecho de estar asociados a otros objetos persistentes. También denominada persistencia por alcanzabilidad, si un objeto es persistente, todo objeto al cual el mismo referencia también es persistente.

**Jerarquía de tipo - Herencia**

* El mecanismo de herencia responde a la pregunta: “es un”.
* Se definen nuevos tipos a partir de tipos predefinidos. Se define un subtipo a partir de un supertipo.
* En jerarquía de Generalización/Especialización, los subtipos especializan al supertipo.
* Se heredan los atributos y métodos: se reutiliza del código y sólo se implementan los nuevos atributos y métodos

*Extensión*: colección de objetos del mismo tipo.

*Restricción*: todos los objetos de una extensión que pertenece a un subtipo deben ser miembros de la extensión que pertenece al supertipo.

**Objetos no estructurados**

Se utilizan para almacenar objetos grandes: objetos bitmap, textos largos, videos.

Los objetos no estructurados son complejos porque requieren gran cantidad de almacenamiento, que puede recuperarse por porciones. También son complejos porque no es posible generar una condición de selección directas sobre estos objetos basado en sus valores.

Se manipulan a través del comportamiento que se define al objeto.

Se dice que los ODBMS es un sistema que extiende los tipos, se pueden crear librerías de nuevos tipos que permiten la generación de una aplicación compleja particular, el DBMS debe proveer los mecanismos de almacenamiento y recuperación.

**Objetos complejos estructurados**

Por ejemplo, una tupla que es lo más parecido a una relación o tabla que su vez puede estar compuesta por otras tuplas o sets (listas, arreglos), y así sucesivamente.

Las maneras de referenciar los objetos son:

* + - *Composición*: los subobjetos son encapsulados dentro del objeto complejo y son considerados parte de él.
    - *Referencia*: son objetos independientes referenciados por el objeto complejo.

En composición, los objetos que lo componen no necesitan OID, son accedidos a través de la interface del objeto. Si se elimina el objeto que lo contiene éstos también son eliminados.

Un objeto referenciado es independiente, tiene su propio OID, puede ser referenciado por más de un objeto y no se elimina automáticamente si el objeto complejo es eliminado.

**ODMG**

Es el estándar de ODBMS. El objetivo es integrar en forma transparente las funcionalidades de BD en un lenguaje de programación OO.

Características:

* + - El usuario no necesita aprender DML independientemente del LP que utiliza.
    - Carga/descarga de objetos implícita.
    - Lenguaje de consulta a bases de objetos. (OQL)

**Componentes de la arquitectura ODMG**

Modelo de objetos: basado en el modelo de objetos de OMG con extensiones para BDs.

Lenguaje de definición de objetos (ODL):

* Equivalente a DDL
* Lenguajes:
  + ODL (Object Definition Language): definición de objetos.
  + OIF (Object InterchangeFormat): intercambio de objetos entre OODBMS y el lenguaje de programación.

OQL (Object Query Language): Lenguaje declarativo para consultar una base de datos basado en SQL.

Enlaces (bindings) con LPOO

* C++ OML (lenguaje de manipulación de objetos)
  + C++ ODL versión de ODL en sintaxis C++
  + Mecanismos para ejecutar sentencias OQL.
  + Mecanismos para ejecutar operaciones sobre la base de datos y manipular transacciones.
* Smalltalk
* Java

**ODMG - Modelo de objetos**

Conceptos soportados por una BD ODMG. Indica las características de los objetos ODMG: cómo se relacionan, cómo se nominan e identifican.

Un modelo específico se construye utilizando ODL (independiente del lenguaje o no).

No todo LPOO implementa necesariamente todos los conceptos del modelo de objetos ODMG.

**Estado y Comportamiento de los objetos**

Estado: Definido por valores de un conjunto de propiedades que pueden ser atributos o relaciones.

Comportamiento: Definido por un conjunto de operaciones (métodos).

**Modelando estados**

Atributos: El valor de un atributo puede ser un literal o un OID. Un atributo es distinto de una estructura de datos. Los atributos pueden ser implementados por estructuras de datos o por métodos que calculan su valor.

Relaciones: Se modelan relaciones binarias solamente. Se implementan entre objetos, no entre literales. Se definen explícitamente y se declaran de a pares. Los caminos de acceso pueden ser ordenado o desordenados dependiendo el tipo de colección que se especifica.

ODBMS es el responsable por mantener la integridad referencial, si el objeto es borrado, todo camino que accede

al objeto también es borrado. Asegura que las aplicaciones no puedan desreferenciar caminos de acceso a objetos no existentes.

Los atributos de un objeto referencian otro objeto, sin caminos de acceso inversos o integridad referencial: “relaciones unidireccionales”.

**Modelando comportamientos**

Operaciones

Firma de las operaciones: nombre, nombre y tipo de cada argumento, tipo de valores retornados, nombre de las excepciones que se pueden “lanzar”.

Una operación debe estar definida sobre un tipo simple. El nombre de una operación debe ser único dentro de la definición de un tipo: no existe polimorfismo estático (sobrecarga nombres función).

Se pueden sobrecargar operaciones sobre distintos tipos. Las operaciones a emplear (en la sobrecarga de operaciones) se resuelven basado en el tipo de objeto más específico definido en el primer argumento de la llamada.

**Herencia**

Herencia de comportamiento: Se define por medio del símbolo “:”. El supertipo siempre tiene que ser una interface. Provista por el mecanismo de tipo y subtipo (tipificación). Admite herencia múltiple. El modelo restringe que las clases pueden heredar de clases e interfaces y las interfaces sólo pueden heredar de interfaces.

Herencia de estado y comportamiento: Se emplea la palabra clave “EXTENDS”. Se emplea para heredar estado y comportamiento entre clases. El supertipo y el subtipo siempre tiene que ser una clase. No admite herencia múltiple.

**Extensión de un tipo (“extend”)**

* Se puede declarar un extent para cualquier tipo definido via una declaración de clase (class).
* Es un nombre que va a contener todos los objetos persistentes de esa clase.
* La extensión de un tipo, es el conjunto de todas las instancias de ese tipo que existen en la BD.
* Equivalente al concepto de tabla en BDs relacionales.
* La extensión es una lista mantenida automáticamente por el ODBMS cuando un objeto de ese tipo es creado, el mismo se inserta en la extensión del tipo.
* Cuando un objeto es destruido, el mismo se elimina de la extensión del tipo.

**Clave**

* Una clase con una extensión puede contener una o más claves cuyos valores son construidos para ser únicos dentro de cada extent.
* Es un concepto útil para un DBMS. El modelo OO no contiene el concepto de clave.
* Si la clave es compuesta se debe especificar entre paréntesis.

**Nombre de Objeto**

* Es posible asociar uno (o más) nombres a un objeto.
* Operación bind del Objeto Database en ODMG.
* El nombre es un identificador provisto por el usuario (el OID es creado por el ODBMS).
* Existe una función que retorna el objeto identificado por un nombre dado.
* Se utiliza para dar nombres a objetos “raíz”, que sirven como puntos de entrada para navegar por la BD.

**Resumen del Modelo de Objetos**

* Diferencia objetos (con identidad) y literales.
* Literales: atómicos, colecciones, estructurados.
* Clases con atributos, métodos, subtipos, extensiones.
* Relaciones uni/bidireccionales, integridad referencial mantenida por el DBMS para las bidireccionales.
* Objetos estructurados, colección de objetos.
* Objetos persistentes y transientes.
* Se pueden asignar nombres a los objetos.
* Repositorio de metadatos (las veces de DD).
* Manejo de excepciones.
* Bloqueo tradicional para el manejo de concurrencia.

**ODMG - OQL**

Una BD puede ser accedida de diferentes formas: navegacionalmente (el programa navega de un objeto a otro utilizando las referencias entre objetos) o asociativamente (una consulta es especificada a través de una expresión. El OODBMS hace accesibles a los objetos que resuelven la consulta al usuario).

**OQL (Object Query Language)**

▪ Lenguaje de consultas al OODBMS.

▪ Originario de la BDOO O2.

▪ Basado en el modelo de objetos de ODMG e inspirado en SQL-92 con extensiones de orientación a objetos (objetos complejos, identidad de objeto, expresiones de camino, polimorfismo, llamada a métodos, “late-binding”).

▪ Provee primitivas para manipular colecciones (conjuntos, listas, etc).

▪ Lenguaje de consultas exclusivamente: actualización por medio de operaciones definidas en el LPOO.

▪ Basado en el sistema de tipos del LPOO: las consultas de OQL pueden ser embebidas dentro del LPOO y pueden invocar métodos programados en el LPOO.

Entradas y resultados

Objetos consultados: colecciones

* + - Objetos con nombre (puntos de entrada en la BD)
    - Extensiones de clase (“extent”)

**Manifiesto Malcom Atkinson**

En 1989 se hizo el manifiesto de BDOO. Tiene 12 características obligatorias y 4 opcionales.

Una BD es OO cuando cumple con los 12 puntos del manifiesto. Las BDOO pueden ser relacionales si cumplen con los 12 puntos.

Características obligatorias:

* Deben soportarse objetos complejos.
* Deben soportarse mecanismos de identidad de los objetos.
* Debe soportarse la encapsulación.
* Deben soportarse los tipos o clases.
* Los tipos o clases deben ser capaces de heredar de sus ancestros.
* Debe soportarse el enlace dinámico.
* El DML debe ser computacionalmente complejo.
* Debe proporcionarse persistencia a los datos.
* El SGBD debe ser capaz de gestionar bases de datos de muy gran tamaño.
* El SGBD debe soportar a usuarios concurrentes.
* El SGBD debe ser capaz de recuperarse de fallos hardware y software.
* El SGBD debe proporcionar una forma simple de consultar los datos.

Características opcionales: herencia múltiple, comprobación de tipos e inferencia de tipos, sistema de BDD.

Ventajas de BDOO

* Mayor capacidad de modelado. El modelado de datos orientado a objetos permite modelar el ‘mundo real’ de una manera mucho más fiel. Esto se debe a:
  + Un objeto permite encapsular tanto un estado como un comportamiento
  + Un objeto puede almacenar todas las relaciones que tenga con otros objetos
  + Los objetos pueden agruparse para formar objetos complejos (herencia).
* Ampliabilidad. Esto se debe a:
  + Se pueden construir nuevos tipos de datos a partir de los ya existentes.
  + Agrupación de propiedades comunes de diversas clases e incluirlas en una superclase, lo que reduce la redundancia.
  + Reusabilidad de clases, lo que repercute en una mayor facilidad de mantenimiento y un menor tiempo de desarrollo.
* Lenguaje de consulta más expresivo. El acceso navegacional desde un objeto al siguiente es la forma más común de acceso a datos en un SGBDOO. Mientras que SQL utiliza el acceso asociativo. El acceso navegacional es más adecuado para gestionar operaciones como los despieces, consultas recursivas, etc.
* Adecuación a las aplicaciones avanzadas de base de datos. Hay muchas áreas en las que los SGBD tradicionales no han tenido excesivo éxito como el CAD, CASE, OIS, sistemas multimedia, etc. en los que las capacidades de modelado de los SGBDOO han hecho que esos sistemas sí resulten efectivos para este tipo de aplicaciones.
* Mayores prestaciones. Los SGBDOO proporcionan mejoras significativas de rendimiento con respecto a los SGBD relacionales. Aunque hay autores que han argumentado que los bancos de prueba usados están dirigidos a aplicaciones de ingeniería donde los SGBDOO son más adecuados.

Inconvenientes de BDOO

* Carencia de un modelo de datos universal. No hay ningún modelo de datos que esté universalmente aceptado para los SGBDOO y la mayoría de los modelos carecen una base teórica.
* Carencia de experiencia. Todavía no se dispone del nivel de experiencia del que se dispone para los sistemas tradicionales.
* Competencia. Con respecto a los SGBDR y los SGBDOR. Estos productos tienen una experiencia de uso considerable. SQL es un estándar aprobado y ODBC es un estándar de facto. Además, el modelo relacional tiene una sólida base teórica y los productos relacionales disponen de muchas herramientas de soporte que sirven tanto para desarrolladores como para usuarios finales.
* La optimización de consultas compromete la encapsulación. La optimización de consultas requiere una compresión de la implementación de los objetos, para poder acceder a la base de datos de manera eficiente. Sin embargo, esto compromete el concepto de encapsulación.
* El modelo de objetos aún no tiene una teoría matemática coherente que le sirva de base.

**Ejemplo: BDOR – Tecnología BDOR de Oracle**

El término Base de Datos Objeto Relacional (BDOR) se usa para describir una base de datos que ha evolucionado desde el modelo relacional hacia otra más amplia que incorpora conceptos del paradigma orientado a objetos. Por tanto, un SGBDOR contiene ambas tecnologías: relacional y de objetos.

Una idea básica de las BDOR es que el usuario pueda crear sus propios tipos de datos, para ser utilizados en aquella tecnología que permita la implementación de tipos de datos predefinidos. Además, las BDOR permiten crear métodos para esos tipos de datos. Con ello, este tipo de SGBD hace posible la creación de funciones miembro usando tipos de datos definidos por el usuario, lo que proporciona flexibilidad y seguridad.

Estos sistemas gestionan tipos de datos complejos con un esfuerzo mínimo y albergan parte de la aplicación en el servidor de base de datos. Permiten almacenar datos complejos de una aplicación dentro de la BDOR sin necesidad de forzar los tipos de datos tradicionales.

Debido a los requerimientos de las nuevas aplicaciones, el sistema de gestión de bases de datos relacionales de Oracle, desde versión 8.i, ha sido significativamente extendido con conceptos del modelo de bases de datos orientadas a objetos. De esta manera, aunque las estructuras de datos que se utilizan para almacenar la información siguen siendo tablas, los usuarios pueden utilizar muchos de los mecanismos de orientación a objetos para definir y acceder a los datos.

Oracle proporciona mecanismos para que el usuario pueda definir sus propios tipos de datos, cuya estructura puede ser compleja, y se permite la asignación de un tipo complejo (dominio complejo) a una columna de una tabla. Además, se reconoce el concepto de objetos, de tal manera que un objeto tiene un tipo, se almacena en cierta fila de cierta tabla y tiene un identificador único (OID). Estos identificadores se pueden utilizar para referenciar a otros objetos y así representar relaciones de asociación y de agregación.

Oracle también proporciona mecanismos para asociar métodos a tipos, y constructores para diseñar tipos de datos multivaluados (colecciones) y tablas anidadas.

Tipos de Datos definidos por el usuario

Un tipo de dato define una estructura y un comportamiento común para un conjunto de datos de las aplicaciones. Los usuarios de Oracle pueden definir sus propios tipos de datos mediante dos categorías: tipos de objetos (object type) y tipos para colecciones (collection type). Para construir los tipos de usuario se utilizan los tipos básicos provistos por el sistema y otros tipos de usuario previamente definidos.

Un tipo de objeto representa una entidad del mundo real y se compone de los siguientes elementos:

* Su nombre que sirve para identificar el tipo de los objetos.
* Sus atributos que modelan la estructura y los valores de los datos de ese tipo. Cada atributo puede ser de un tipo de datos básico o de un tipo de usuario.
* Sus métodos (procedimientos o funciones) escritos en lenguaje PL/SQL (almacenados en la BDOR), o escritos en C (almacenados externamente).

Los tipos de objetos actúan como plantillas para los objetos de cada tipo. A continuación, vemos un ejemplo de cómo definir el tipo de dato Direccion\_T en el lenguaje de definición de datos de Oracle, y cómo utilizar este tipo de dato para definir el tipo de dato de los objetos de la clase de Cliente\_T.

Métodos

La especificación de un método se hace junto a la creación de su tipo, y debe llevar siempre asociada una directiva de compilación (PRAGMA RESTRICT\_REFERENCES), para evitar que los métodos manipulen la base de datos o las variables del paquete PL/SQL. Tienen el siguiente significado:

* WNDS: no se permite al método modificar las tablas de la base de datos
* WNPS: no se permite al método modificar las variables del paquete PL/SQL
* RNDS: no se permite al método leer las tablas de la base de datos
* RNPS: no se permite al método leer las variables del paquete PL/SQL

Los métodos se pueden ejecutar sobre los objetos de su mismo tipo. Si x es una variable PL/SQL que almacena objetos del tipo Cliente\_T, entonces x.edad() calcula la edad del cliente almacenado en x.

Constructores

En Oracle, todos los tipos de objetos tienen asociado por defecto un método que construye nuevos objetos de ese tipo de acuerdo a la especificación del tipo. El nombre del método coincide con el nombre del tipo, y sus parámetros son los atributos del tipo.

Métodos de comparación

Para comparar los objetos de cierto tipo es necesario indicar a Oracle cuál es el criterio de comparación. Para ello, hay que escoger entre un método MAP u ORDER, debiéndose definir al menos uno de estos métodos por cada tipo de objeto que necesite ser comparado. La diferencia entre ambos es la siguiente:

* Un método MAP sirve para indicar cuál de los atributos del tipo se utilizará para ordenar los objetos del tipo, y por tanto se puede utilizar para comparar los objetos de ese tipo por medio de los operadores de comparación aritméticos (<, >).
* Un método ORDER utiliza los atributos del objeto sobre el que se ejecuta para realizar un cálculo y compararlo con otro objeto del mismo tipo que toma como argumento de entrada. Este método devolverá un valor negativo si el parámetro de entrada es mayor que el atributo, un valor positivo si ocurre lo contrario y un cero si ambos son iguales.

Si un tipo de objeto no tiene definido ninguno de estos métodos, Oracle es incapaz de deducir cuándo un objeto es mayor o menor que otro. Sin embargo, sí puede determinar cuándo dos objetos del mismo tipo son iguales. Para ello, el sistema compara el valor de los atributos de los objetos uno a uno:

* Si todos los atributos son no nulos e iguales, Oracle indica que ambos objetos son iguales.
* Si alguno de los atributos no nulos es distinto en los dos objetos, entonces Oracle dice que son diferentes.
* En otro caso, Oracle dice que no puede comparar ambos objetos.

Tablas de Objetos

Una vez definidos los tipos, éstos pueden utilizarse para definir nuevos tipos, tablas que almacenen objetos de esos tipos, o para definir el tipo de los atributos de una tabla. Una tabla de objetos es una clase especial de tabla que almacena un objeto en cada fila y que facilita el acceso a los atributos de esos objetos como si fueran columnas de la tabla. Se puede definir una clave en la tabla.

La cláusula VALUE permite visualizar el valor de un objeto.

Referencias entre objetos

Los identificadores únicos asignados por Oracle a los objetos que se almacenan en una tabla, permiten que éstos puedan ser referenciados desde los atributos de otros objetos o desde las columnas de tablas. El tipo de datos proporcionado por Oracle para soportar esta facilidad se denomina REF. Un atributo de tipo REF almacena una referencia a un objeto del tipo definido, e implementa una relación de asociación entre los dos tipos de objetos. Estas referencias se pueden utilizar para acceder a los objetos referenciados y para modificarlos; sin embargo, no es posible operar sobre ellas directamente. Para asignar o actualizar una referencia se debe utilizar siempre REF o NULL.

Cuando se define una columna de un tipo a REF, es posible restringir su dominio a los objetos que se almacenen en cierta tabla. Si la referencia no se asocia a una tabla, sino que sólo se restringe a un tipo de objeto, se podrá actualizar a una referencia a un objeto del tipo adecuado con independencia de la tabla donde se almacene. En este caso su almacenamiento requerirá más espacio y su acceso será menos eficiente. El siguiente ejemplo define un atributo de tipo REF y restringe su dominio a los objetos de cierta tabla.

Cuando se borran objetos de la BD, puede ocurrir que otros objetos que referencien a los borrados queden en estado inconsistente. Estas referencias se denominan dangling references, y Oracle proporciona el predicado llamado IS DANGLING que permite comprobar cuándo sucede esto.

Tipos de datos colección

Para poder implementar relaciones 1:N, Oracle permite definir tipos colección. Un dato de tipo colección está formado por un número indefinido de elementos, todos del mismo tipo. De esta manera, es posible almacenar en un atributo un conjunto de tuplas en forma de array (VARRAY), o en forma de tabla anidada.

Al igual que los tipos objeto, los tipos colección también tienen por defecto unas funciones constructoras de colecciones cuyo nombre coincide con el del tipo.

En Oracle es posible diferenciar entre un valor nulo y una colección vacía. Para construir una colección sin elementos se puede utilizar la función constructora del tipo seguida por dos paréntesis sin elementos dentro.

El tipo VARRAY

Un array es un conjunto ordenado de elementos del mismo tipo. Cada elemento tiene asociado un índice que indica su posición dentro del array. Oracle permite que los VARRAY sean de longitud variable, aunque es necesario especificar un tamaño máximo cuando se declara el tipo VARRAY. Las siguientes declaraciones crean un tipo para una lista ordenada de precios, y un valor para dicho tipo.

Se puede utilizar el tipo VARRAY para definir el tipo de dato de una columna de una tabla relacional, definir el tipo de dato de un atributo de un tipo de objeto o para definir una variable PL/SQL, un parámetro, o el tipo que devuelve una función.

Cuando se declara un tipo VARRAY no se produce ninguna reserva de espacio. Si el espacio que requiere lo permite, se almacena junto con el resto de columnas de su tabla, pero si es demasiado largo (más de 4000 bytes) se almacena aparte de la tabla como un BLOB.

La principal limitación del tipo VARRAY es que en las consultas es imposible poner condiciones sobre los elementos almacenados dentro. Desde una consulta SQL, los valores de un VARRAY solamente pueden ser accedidos y recuperados como un bloque. Es decir, no se puede acceder individualmente a los elementos de un VARRAY. Sin embargo, desde un programa PL/SQL si que es posible definir un bucle que itere sobre los elementos de un VARRAY.

Tablas anidadas

Una tabla anidada es un conjunto de elementos del mismo tipo sin ningún orden predefinido. Estas tablas solamente pueden tener una columna que puede ser de un tipo de datos básico de Oracle, o de un tipo de objeto definido por el usuario. En este último caso, la tabla anidada también puede ser considerada como una tabla con tantas columnas como atributos tenga el tipo de objeto.

Inserción y Acceso a los datos

*Alias*: En una base de datos con tipos y objetos, lo más recomendable es utilizar siempre alias para los nombres de las tablas. El alias de una tabla debe ser único en el contexto de la consulta. Los alias sirven para acceder al contenido de la tabla, pero hay que saber utilizarlos adecuadamente en las tablas que almacenan objetos.

*Inserción de Referencias*: La inserción de objetos con referencias implica la utilización del operador REF para poder insertar la referencia en el atributo adecuado. La siguiente sentencia inserta una orden de pedido. El acceso a un objeto desde una referencia REF requiere primero referenciar al objeto. Para realizar esta operación, Oracle proporciona el operador DEREF. No obstante, utilizando la notación de punto también se consigue referenciar a un objeto de forma implícita.

*Llamadas a métodos*: Para invocar un método hay que utilizar su nombre y unos paréntesis que encierren sus argumentos de entrada. Si el método no tiene argumentos, se especifican los paréntesis, aunque estén vacíos.

Inserción en tablas anidadas: Además del constructor del tipo de colección disponible por defecto, la inserción de elementos dentro de una tabla anidada puede hacerse siguiendo estas dos etapas:

1. Crear el objeto con la tabla anidada y dejar vacío el campo que contiene las tuplas anidadas.

2. Comenzar a insertar tuplas en la columna correspondiente de la tupla seleccionada por una subconsulta.

Esta técnica es especialmente útil si dentro de una tabla anidada se guardan referencias a otros objetos.

**Mapeo Objeto Relacional (ORM)**

* Los ORM son herramientas de software que permiten trabajar con los datos persistidos en nuestras bases de datos relacionales como si ellos fueran parte de una base de datos orientada a objetos (virtual).
* En aplicaciones estándar, realizadas sobre bases de datos relacionales, la función del ORM es transformar un registro en objeto y viceversa, en pos de realizar operaciones de consulta y persistencia directamente sobre los objetos.
* Existen variados componentes que se pueden utilizar a tal fin en tecnología .NET (muchos de los cuales son portaciones de componentes existentes en JAVA).
* El más utilizado y el que posee un mayor soporte en la actualidad es NHIbernate, aunque también contamos con otros muy buenos como ORM.net oWilson ORM Mapper.
* Existen, además, herramientas que "encapsulan" un ORM e incorporan funcionalidad adicional como ActiveRecord.net, que utiliza NHibernate para persistir, pero además facilita su uso notablemente evitándonos la realización de archivos de mapeo (necesarios para relacionar nuestras tablas con nuestras clases) e incorpora validaciones diversas sobre nuestras clases.

¿Cuándo usar ORM?

Consideremos usar un ORM si:

* Tenemos la posibilidad de diseñar la base de datos, ya que, de esa manera, podremos seguir prácticas que facilitarán la interacción ORM-DB.
* Queremos abstraernos del motor de base de datos (SQL Server, Oracle) y poder cambiarlo sin demasiado inconveniente).

Y evitemos utilizarlo cuando:

* Recibamos bases de datos que posean características tales como tablas con varios campos como clave principal o sin claves primarias y/o foráneas, ya que eso podría dificultar el mapeo y las consultas.
* Necesitemos ejecutar procesos BATCH con millones de registros

**U6: NUEVAS TENDENCIAS EN BD**

**3ra. Generación de BD**

“Proporciona capacidades de gestión de datos, permitiendo que grandes cantidades de datos persistentes sean compartidos por muchos usuarios. También proporcionan gestión de objetos, permitiendo tipos de datos muchos más complejos, objetos multimedia, datos derivados, encapsulamiento de la semántica de los datos, así como otras nuevas capacidades. Algunos proporcionan incluso gestión de conocimiento, soportando un gran número de reglas complejas para inferencia automática de información y mantener las restricciones de integridad entre datos” Cattell (1991)

**BD Activas**

**Motivación**

Las bases de datos convencionales se consideran “muertas” o pasivas, en el sentido de que es el usuario o el programa de aplicación quien decide qué hacer a la base de datos y ella no reacciona a las acciones ejecutadas sobre ella.

* *¿Cómo se puede motivar la necesidad de que una base de datos sea activa?*

Supongamos una aplicación de inventario para una fábrica de productos que se apoya en una base de datos de los productos, su cantidad en stock (existencia) y que varía de acuerdo con la venta de productos. La fábrica tiene la descripción de cada producto, del cual se tiene una cantidad en existencia; a medida que se venden los productos se altera la cantidad en existencia. Lo deseable es que cada producto tenga un rango de la cantidad en existencia, definido por un nivel mínimo y un nivel máximo de cantidad.

Si la cantidad en existencia de un producto se sale de su rango, lo ideal es que se detecte esta situación inmediatamente y se tomen medidas al respecto.

* *¿Cuáles son las alternativas de solución para este problema?*

1. En cada transacción que altera el valor de la cantidad en existencia, se coloca la verificación de los límites y se toman las medidas cuando se violen los límites. Es un desastre en mantenimiento y constituye una mala práctica de ingeniería de software.

2. Se construye una transacción especial que, periódicamente revisa la base de datos, verificando el rango de la cantidad en existencia de cada producto con la cantidad real. ¿Cuándo se hace la verificación? ¿con cuál periodicidad? Si se hace muy a menudo, puede incidir negativamente en el rendimiento y puede ser muy ineficiente. Si se hace con poca frecuencia, puede que no se detecten las condiciones a tiempo y no se tomen las acciones oportunamente.

* Comportamiento Activo:Cuando sucede algo, hacer algo. CUANDO + QUE

**Base de Datos Activa**

El paradigma de bases de datos activas planteado por Morgenstern en 1983, describe la noción de que una base de datos sea activa, como una metáfora de su comportamiento, el cual se concentra en: la dinámica de la interacción con los usuarios unido a la “inteligencia” de la base de datos para lidiar con las consecuencias e implicaciones de esa interacción.

Una base de datos activa, es aquella base de datos capaz de detectar situaciones de interés y de actuar en consecuencia. Alguien debe especificar las situaciones a detectar y las acciones a llevar a cabo en esas situaciones. El mecanismo que se utiliza para especificar estos aspectos se parece a las reglas de producción utilizadas en el área de inteligencia artificial.

En el área de bases de datos las reglas que se utilizan para especificar estas situaciones y sus acciones se llaman reglas del tipo evento-condición-acción o reglas que siguen el paradigma de evento-condición-acción.

El formato genérico de estas reglas es:

ON evento

IF condicion

THEN accion

El lenguaje de reglas de este tipo debe tener componentes para especificar eventos, especificar condiciones y especificar acciones.

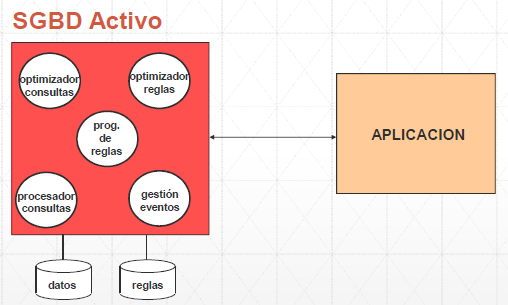
**Semántica de la especificación de reglas**

* Granularidad del procesamiento de las reglas: Es importante saber si una regla se dispara una vez por cada tupla “tocada” o una sola vez por todas las tuplas “tocadas”. También hay otras opciones, por ejemplo al final de la transacción que se está ejecutando cuando se lanzó la regla.
* Anidamiento de reglas y terminación: Otro aspecto importante es si se puede especificar una sola regla por evento o más de una. En el caso en el que se permita más de una regla por evento, en cuál orden se ejecutan esas reglas y cuándo se termina la ejecución, son otros aspectos a considerar.
* Concurrencia con las transacciones: Es necesario determinar si las reglas se van a ejecutar como parte de la transacción donde se disparen o si se van a ejecutar como transacciones aparte. Esto es fundamental por la propiedad de atomicidad que se debe garantizar para las transacciones de una base de datos.

**Aplicaciones de BD activas**

Una primera clasificación de las aplicaciones lo establece el uso de las reglas para labores internas del DBMS o para labores externas, las cuales son especificadas por el usuario y permiten realizar labores específicas dependientes del dominio del problema.

* Internas: Soportar las características clásicas del manejo o administración de las bases de datos. Ejemplos de estas aplicaciones son:
* Control de integridad. (Restricciones implícitas y explícitas.)
* Mantenimiento de vistas y datos derivados, los cuales pueden existir virtualmente o ser materializados.
* Administración de copias de los datos (duplicación).
* Seguridad. Recuperación ante fallas.



* Externas: Estas aplicaciones contienen conocimiento de la aplicación, expresado en la forma de reglas, a las cuales comúnmente se les llama reglas del negocio. Con respecto al control de integridad las restricciones que se pueden establecer con las reglas activas son:
* Restricciones estáticas: se evalúan sobre un estado de la base de datos, un ejemplo de las cuales son las restricciones de dominio.
* Restricciones dinámicas: se evalúan sobre la transición de un estado a otro, por ejemplo: el sueldo de un empleado solo puede aumentar.

**Labores externas en una BDA**

Independientemente de si las restricciones son estáticas o dinámicas, dependiendo de quién las específica, se pueden dividir en:

* Restricciones “built-in”: son fijas y especificadas con cláusulas del DDL, por ejemplo: referential integrity (foreign keys, REFERENCES) y claves primarias (PRIMARY KEY).
* Genéricas: especificadas por el usuario, por ejemplo con la definición de CONSTRAINTS; algunos ejemplos de éstos son: NOT NULL, UNIQUE y CHECK.

**Estándar de SQL y BDA**

El primer estándar de SQL donde se consideró la noción de ser activa de una base de datos fue en SQL:99 (es decir, en SQL 3); pero el estándar que salió en 1992 no se incluyó este aspecto porque ya era muy extenso el estándar y quienes lo desarrollaron pensaron, equivocadamente, que los implementadores de DBMS no iban a implementar ese aspecto en los manejadores por un tiempo. Sin embargo, los desarrolladores si implementaron este aspecto, pero como no había estándar, cada uno realizó su implementación independientemente, obteniendo así una gran diversidad de implementaciones de triggers, específicamente. La descripción de triggers y assertions si se encuentra en SQL:99, por lo tanto, ahora se quiere tratar de uniformizar todas las implementaciones existentes.

En síntesis, los aspectos de base de datos activas en el estándar SQL:99 se concentran en los siguientes aspectos: constraints, triggers y assertions.

**Constraints**

Son especificaciones del SQL que se aplican a columnas o tuplas de una tabla. Algunos tipos especiales de constraints son: UNIQUE, NOT NULL, REFERENCES, CHECK; éste último permite especificar una amplia gama de reglas, como, por ejemplo, rangos de valores y listas de valores, entre otros. Es conveniente que cada constraint tenga un nombre, pues cuando el constraint es violado el sistema indica su nombre y se puede saber exactamente qué falló.

La verificación de las restricciones y el hacer que se cumplan se puede hacer de manera inmediata (constraint check time IMMEDIATE) durante la ejecución de la transacción o de forma diferida (constraint check time DEFERRED), es decir, al tiempo de comprometerse de la transacción (commit).

Los constraints pueden ser violados solo por una o más tuplas de una tabla y no por la tabla en sí misma, es decir, que una tabla vacía cumple con todos los constraints, una tabla vacía no viola restricción alguna.

**Assertions**

Es un tipo de restricción especial que se puede especificar en SQL sin que deba estar asociada a una tabla en particular, como es el caso de los constraints. Generalmente se utilizan para describir restricciones que afectan a más de una tabla.

Como los constraints sólo se pueden establecer sobre tuplas de una tabla, los assertions son útiles cuando es necesario especificar una condición general de la base de datos que no se puede asociar a una tabla específica de la base de datos. Por esta característica de poder especificar un assertion para toda la base de datos y no para una tabla en particular, también se les llama standalone constraints. Esto tiene grandes ventajas a nivel conceptual, pero las hace difíciles de implementar. La reacción es abortar.

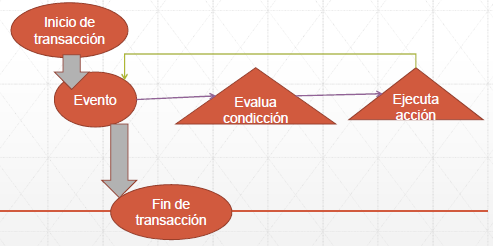
**Triggers**

Los triggers en SQL:99 son parecidos a los constraints, pero proveen mayor flexibilidad pues el usuario puede especificar un conjunto de acciones complejas a ejecutarse cuando el trigger se dispare. Los constraints actúan cuando se viola lo que ellos indican, pero el usuario no puede especificar acción alguna con esa violación, sino que el DBMS simplemente envía un mensaje de error e impide que se realice la operación que produjo la violación.

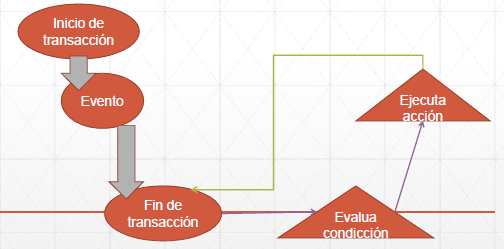
**Modos de Acoplamiento**

¿Cuándo se evalúa la condición?

▪ Modo de acoplamiento Inmediato:



▪ Modo de acoplamiento diferido:



**Manifiesto de los SGBDA**

Características de los SGBDA

* Un SGBDA es un SGBD.
* Un SGBDA tiene un modelo de reglas ECA.
* Un SGBDA debe soportar la gestión de reglas y la evolución de la base de reglas.

Características de ejecución de reglas ECA

* Un SGBDA tiene un modelo de ejecución.
* Un SGBDA debe ofrecer diferentes modelos de acoplamiento.

**BD Temporales**

Motivación

El manejo de fechas en SQL es limitado, fundamentalmente cuando se requieren datos que cambian con el tiempo.

Ejemplo: agregar una fecha inicio y fin de vigencia de registros.

* Se mantiene historia de los datos.
* Consume más almacenamiento porque hay insert, no update ni delete.
* La búsqueda es complicada y las consultas tiene peor desempeño.

**Bases de datos temporales**

TDB: Toda aplicación de BD que requiere de algún aspecto del tiempo para organizar su información. Es una BD que incorpora el modelo de datos en el tiempo. La dimensión temporal tiene mucha relevancia.

Muchas aplicaciones requieren del tiempo:

* Salud: se debe mantener la historia clínica de los pacientes
* Seguros: se debe llevar la historia de los reclamos y accidentes y la información de las pólizas y el tiempo en que son efectivas
* Los sistemas de reservas de hoteles, aerolíneas (transporte en general), alquiler de autos
* Bases de datos científicas: datos de los experimentos incluyendo las fechas en que se midieron los datos, etc.
* En un sistema de ventas, se puede dar temporalidad únicamente a una tabla Precio.

**Representación del tiempo y calendario**

* Definir unidad de tiempo: Para BDT, el tiempo es considerado como una secuencia ordenada de puntos de alguna granularidad que es determinada por la aplicación, por ejemplo, una aplicación nunca requiere una unidad de menos de 1 seg. cada punto implica 1 seg.duración investigadores prefieren hablar de cronos. Fecha\_desde se suele usar como clave primaria porque no hay dos registros en un mismo momento.

El problema de la mínima unidad de tiempo es que depende de lo que se elija dos eventos pueden considerarse simultáneos cuando no lo son.

* Definir el calendario: existen varios: Gregoriano, Chino, Islámico, etc., con diferentes punto de referencia pero, por lo general, 60 segundos, 60 minutos, 24 hs., 7 días…
* Definir el tipos de datos: DATE(Fecha), TIME(hora, min., seg.), TIMESTAMP(Combinación Fecha/Tiempo), INTERVAL(duración de tiempo relativa(10 días, 250 minutos) PERIOD(duración de tiempo con un punto de comienzo fijo, 10 días de duración desde el 1 de Enero de 1999, al 10 de Enero de 1999 inclusive).

**Información del evento vs Información de duración**

Determinar si los sucesos son puntos (momentos únicos) o un evento durable (período).

Información de cuando el evento ocurre o cuando ciertos hechos se consideran verdaderos:

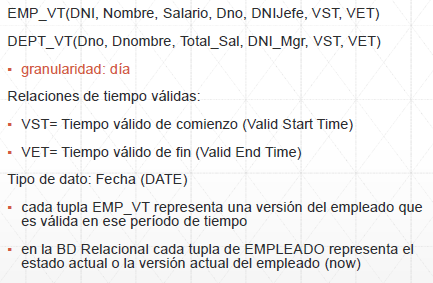
* Eventos puntuales (sucesos) están asociados en la BD con un punto de tiempo de alguna granularidad: Depósito en un banco tiene un TIMESTAMP, los montos de venta de un mes están asociados a ese mes (Puntos de diferentes granularidades)
* Eventos durables (sucesos) están asociados con un período de tiempo en la BD. El período de tiempo que un empleado trabajó en la compañía. El período de tiempo se representa por su punto de comienzo y fin. Algunas veces el período se representa por el conjunto de todos los puntos de tiempo entre ellos de una dada granularidad.

**Dimensiones del tiempo**

Significados que puede tener un evento asociado a un tiempo puntual o un período de tiempo:

* que sea verdadero en el mundo real al tiempo en que el evento ocurrió o el período de tiempo que duró el evento TIEMPO VÁLIDO → BASE de DATOS de TIEMPOS VALIDOS
* válido en el sistema → el tiempo asociado es el que se refiere al momento en que la información fue registrada TIEMPO de TRANSACCION → BASE de DATOS de TIEMPOS de TRANSACCION
* Se tienen otras interpretaciones, pero estas dos son las más comunes.
* Algunas aplicaciones pueden necesitar sólo una de las dimensiones, si necesitan ambas: BASES de DATOS BITEMPORALES.

**El tiempo en BD Relacionales**



Los datos no se actualizan, sino que se genera una nueva versión (se cambia el VET) indicando que esa versión de los datos es una “historia cerrada”.

Actualización Preactiva: se aplica en la BD antes del cambio efectivo en el mundo real.

Actualización Retroactiva: se aplica en la BD después del cambio efectivo en el mundo real – Simultánea: al mismo tiempo.

Relaciones de tiempo validas:

* La eliminación en las BD temporales es cerrar la versión actual de una tupla por medio de darle una valor de tiempo válido a VET.
* La inserción se corresponde con la primera versión de la tupla en la tabla.
* Notar que en una relación de tiempo válida las claves no-temporales no son más únicas en cada tupla, entonces la nueva clave la constituyen la combinación de la clave no-temporal y el tiempo válido de comienzo VST. Clave primaria (DNI, VST)
* No deben existir períodos de tiempo con alguna intersección para dos versiones de tupla que representan a la misma entidad.
* Es importante que la clave primaria no-temporal no cambie con el tiempo para poder relacionar todas las versiones de la misma entidad.
* Las relaciones de tiempo válida tratan de representar la historia del mundo real, dado que las bases de datos aplican los cambios de manera preactiva o retroactiva, pueden no existir registros del estado actual de la BD, si esto es importante, se debe trabajar con relaciones de tiempo transaccionales.

Relaciones de tiempo transaccionales

* Cada vez que hay un cambio en la BD (insert, update, delete) se registra el TIMESTAMP de la transacción que hizo el cambio.
* Es útil cuando los cambios se aplican simultáneamente en la mayoría de los casos: control de stock en tiempo real, transacciones bancarias
* En lugar de act se pone hc (hasta que cambie) la información es correcta hasta que otra transacción la cambie
* Base de Datos rollback el usuario puede hacer un rollback del estado actual de la BD hasta cualquier punto en el pasado (el rollback transaccional es lógico, no físico)

**BD Espaciales: Sistema GIS**

Saber: tipos de datos modo de representación, componentes, etapa de creación, ventajas y desventajas

Introducción

Antecedentes:

* El inicio del desarrollo de los SIG comienza en los años 60, pero relacionados con diversos campos de investigación.
* A principios de los 80, su uso se hizo más frecuente gracias al desarrollo de una tecnología informática adecuada, que fomentó la aparición de productos SIG en el mercado.
* Hoy en día, se están produciendo fuertes inversiones en el desarrollo de bases de datos de información geográfica y en sistemas SIG.
* El uso de los Sistemas de Información Geográfica (SIG) ha aumentado enormemente en las últimas décadas pasando del total desconocimiento a la práctica cotidiana.
* Factores que han fomentado el desarrollo de los SIG:
* El abaratamiento de los equipos informáticos.
* Uso de la información geográfica desde múltiples ámbitos para ayudar a la toma de decisiones.
* La sociedad de la información que demanda cada vez más datos de cualquier ámbito.
* En resumen, están de total actualidad y son muchas las áreas en las que intervienen.

**¿Qué es un SIG?**

Es una tecnología de manejo de información geográfica.

**Arquitectura de un SIG**

Un SIG está formado por cinco componentes:

HW: Equipos informáticos donde opera el SIG.

SW: Son los programas. Proveen las funciones y las herramientas necesarias para almacenar, analizar y desplegar la información geográfica. Los principales componentes de los son: Herramientas para la entrada y manipulación de la información geográfica, DBMS, Herramientas para búsquedas geográficas, análisis y visualización, GUI

Datos geográficos: Es la parte más importante.

La realidad geográfica es muy compleja por tres motivos: la diversidad de las partes que la forman, hay un gran número de relaciones, mundo en continuo cambio.

La información geográfica es el elemento diferenciador de un SIG. La información geográfica tiene tres atributos:

* Espacial: delimitación espacial de cada uno de los objetos geográficos.
* Temático: datos asociados a una localización.
* Temporal: describir los cambios ocurridos en el transcurso del tiempo.
* *Base de datos geográfica* (o espacial): Es el conjunto de datos geográficos relacionados entre sí. Puede utilizar los manejadores de base de datos más comunes para manejar la información geográfica. El SIG integra datos espaciales con otros de datos: Cartografía (dato espacial) + Datos (BD temáticas).
* *Tipo de datos de representación*: En general la información geográfica es representada por tres tipos de entidades espaciales: puntos, líneas y áreas o polígonos.
* Puntos: Se utilizan para representar aquellas entidades que son demasiado pequeñas para ser mostradas como áreas o líneas. Por ejemplo, un punto puede representar un cable telegráfico o una ciudad dependiendo de la escala.
* Líneas: Es básicamente un conjunto de puntos ordenados, utilizados para representar entidades que son demasiado pequeñas como para ser representadas como áreas (como ejemplo un río cuyo ancho no amerita esta representación).
* Áreas o Polígonos: Figuras planas con límites de líneas. Representan campos, áreas administrativas, ciudades.

El problema de definir con qué tipo de entidad representamos a determinado elemento, no es una cuestión menor: ¿un edificio, lo representamos como un punto o un área? ¿Un camino como una línea o un área?

Recurso humano: Personal que opera, desarrolla y administra el sistema. Establece los planes para aplicarlo en del mundo real. Resuelve los problemas de E/S de datos, diseña el modelo de BD, etc.

Procedimientos: Un SIG operará adecuadamente si está basado en un buen diseño, con unas reglas claras de actividad de negocio.

**Función de los componentes de un SIG**

Las principales funciones que se llevan a cabo en un SIG son:

* Captura de la información: normalmente mediante de digitalización, procesamiento de satélite, fotografías, videos y aerofotogramétricos.
* Análisis: se realiza tanto con los datos gráficos como no gráficos:
  + Contigüidad de objetos sobre un área determinada.
  + Coincidencia en la superposición de objetos sobre un mapa.

**La información de un SIG**

* Existen varias formas de modelar la realidad geográfica que se diferencian en los principios de modelado que usan, los medios y elementos para representar los objetos reales y en los objetivos que persiguen.
* Para construir un modelo hay que identificar los objetos de estudio y clasificarlos en función de su naturaleza. En geografía se distinguen dos grandes grupos de objetos:
  + Discretos: pueden ser naturales o artificiales pero tienen una manifestación física con límites reconocibles y diferenciables de otros individuos.
  + Continuos: prácticamente todos aquellos son de origen natural y su delimitación es fruto de definiciones. Sus valores son medidas de una característica de una localización espacial determinada.
* Para agrupar o clasificar los objetos en clases o categorías, es necesario hacer un análisis de las necesidades de los usuarios y del modelo de datos del que se dispone en el problema concreto. De esa manera será más fácil extraer las características del objeto que lo situarán en una categoría determinada.

**Modelos de datos de un SIG** (forma de representar los datos)

* El modelo de datos elegido es clave a la hora de implementar un SIG ya que de él dependerán los tipos de análisis que se harán, el modo de visualización de los datos, etc.
* El modelo de datos es el conjunto de reglas a utilizar para formar representaciones del territorio en un entorno digital y discreto.
* Los datos espaciales han de codificarse como entidades discretas para poder ser almacenados en ordenadores, siendo los puntos, líneas y superficies los elementos básicos de representación.
* La información que finalmente se maneja en SIG, es la de objetos concretos de la superficie terrestre que están perfectamente ubicados bajo un sistema convencional de coordenadas. Estos objetos tienen una dimensión física y una posición medible en el espacio relativo a la superficie terrestre.
* Todo objeto tiene asociados unos atributos que pueden ser:
  + Gráficos. Son las representaciones de los objetos geográficos asociados con ubicaciones específicas en el mundo real, y se usan para representarlos puntos, líneas o áreas.
  + Alfanuméricos. Se corresponden con las descripciones o características de los objetos geográficos a los que se refieren.
* Estos atributos suelen estar relacionados.
* Existen otros dos tipos de relaciones:
  + La relación posicional que informa de dónde está el elemento respecto de un sistema de coordenadas establecido.
  + La relación topológica que informa de la relación que tiene el elemento con otros elementos de su entorno geográfico próximo.
* Los objetos se agrupan con otros que tienen características comunes formando así un mapa temático que puede considerarse con una unidad básica de almacenamiento.
* Algunos de los modelos de datos de información geográfica son los siguientes: Diseño asistido, Diseño gráfico, Tratamiento de imágenes, Georrelacional, Raster, TIN (Red de triángulos irregulares), Red, Geobase de datos. La mayoría de ellos utilizan como modo de representación de la información geográfica, sistemas vectoriales o raster.
* Los SIG raster se basan en considerar que existen relaciones de vecindad entre los objetos geográficos. El funcionamiento consiste en dividir la zona de estudio en una malla regular de pequeñas celdas (pixeles) y atribuir un valor numérico a cada celda como representación de su valor temático. Cuanto más pequeña sea la malla, la descripción será más precisa pero la captura de la información y su procesamiento serán más costosos. El uso de este tipo de SIG es adecuado para objetos geográficos con límites difusos. Ej.: imagen.jpg
* Los SIG vectoriales usan vectores definidos por pares de coordenadas relativas a algún sistema cartográfico para representar los objetos geográficos. Esta forma de representación es adecuada cuando los objetos geográficos con los que se trabaja, tienen límites bien establecidos, como pueden ser fincas, carreteras, etc. Ej.: imagen vectorial

**Recogida y organización de la información en un SIG**

Necesidad de disponer de datos adecuados en cantidad, calidad y forma.

Alto coste de preparación de los datos de un SIG: captura, edición, estructuración, reestructuración.

La calidad del valor de los resultados dependerá de la calidad de los datos. (“Si sus datos son falsos, pero su lógica es perfecta, entonces sus contenidos serán inevitablemente falsos. Por tanto, si comete errores con su lógica, tiene al menos una probabilidad aleatoria de llegar a la conclusión correcta.” (Teorema de Christie-Davis.))

**Tipos de fuentes de datos**

Cada proyecto utiliza distintas fuentes de datos que a su vez se encuentran en formatos variados. Una clasificación simple respecto a las fuentes y formatos de los mismos sería: Primarias/Secundarias, Digitales/No digitales.

Datos primarios: Acordes con los objetivos de los proyectos. Son costosos de obtener. Más control sobre los mismos (unidades de medida, los procesos de medición…)

Datos secundarios: Producidos por otros agentes (normalmente con otras finalidades). El uso de la información geográfica publicada necesita del conocimiento de las características de esos datos, de “metadatos”.

**Captura de datos**

* Importación de información espacial directamente de archivos digitales.
* Creación de datos:
  + Tableta digitalizadora: Proporciona como resultado un conjunto de pares de coordenadas que pueden representar puntos, líneas y polígonos.
  + Escáner: Genera una matriz de valores que representa la desigual reflectancia de pequeñas partes de la imagen original.

Tratamiento de imagen digital:

* Edición manual o semiautomática:
  + Supresión de información innecesaria.
  + Completar las entidades (cerrar una región, reducir el grosor de las líneas, etc.).
* Dotar imagen de geometría del proyecto. (funciones de transformación).

**Etapas del diseño de una base de datos para una aplicación SIG**

1. Modelo conceptual.

2. Definición de objetos y relaciones.

3. Selección de la representación geográfica.

4. Modelo lógico de la base de datos.

5. Organización y estructura de la base de datos.

**Uso y aplicación de los sistemas de información geográfica**

* La utilidad principal de un Sistema de Información Geográfica radica en su capacidad para construir modelos o representaciones del mundo real a partir bases de datos espaciales.
* Algunas de sus aplicaciones principales:
  + Cartografía automatizada: Construcción y mantenimiento de planos digitales de cartografía.
  + Infraestructura: Almacenamiento de información alfanumérica de servicios relacionados con las distintas representaciones gráficas de los mismos para el desarrollo, mantenimiento y administración de redes de electricidad, gas, agua, teléfono, alcantarillado...
  + Gestión territorial: Creación de aplicaciones SIG dirigidas a la gestión de entidades territoriales.
  + Medio ambiente: Sistemas SIG que facilitan la evaluación del impacto ambiental en la ejecución de proyectos.
  + Equipamiento social: Implementación de aplicaciones GIS dirigidas a la gestión de servicios de impacto social, tales como servicios sanitarios, centros escolares.
  + Recursos Mineros, Ingeniería de tránsito, Demografía, GeoMarketing, Banca, Planimetría, Cartografía digital 3D.
* Los SIG en los negocios se pueden considerar como una herramienta de apoyo en la toma de decisiones. (Identificación de patrón espacial en sus datos.)
  + Las aplicaciones tácticas proporcionan información requerida para la toma de decisiones.
  + Los SIG apoyarán la combinación de datos espaciales: por ejemplo, información sobre mercado potencial y dónde se localizan los competidores.
  + Las aplicaciones estratégicas proporcionan la información ad\_hoc que se necesita para tomar decisiones estratégicas.

**BD Deductivas o lógicas**

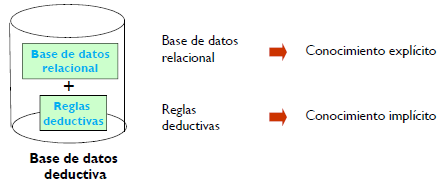
**Motivación BD Deductivas**

En el afán de ofrecer una respuesta a las necesidades planteadas por los usuarios y por las aplicaciones avanzadas, en donde se necesitan herramientas semánticamente más ricas que las provistas por las Bases de Datos Relacionales, aparecen recientes aplicaciones de los sistemas de bases de datos que consiste en ofrecer recursos para definir Reglas Deductivas que permitan deducir, inferir u obtener información nueva a partir de los datos almacenados.

La meta de estas aplicaciones es incorporar a las Bases de Datos Relacionales los beneficios de la lógica como instrumento para la formalización integrada de los aspectos estáticos y dinámicos del modelado de aplicaciones.

**BD Deductivas**

Las Bases de Datos Deductivas extienden la capacidad expresiva de las bases de datos relacionales incluyendo un conjunto de reglas que permiten definir conocimiento implícito.



Existen diversas clases de BDDs y para cada una de ellas existe una semántica bien definida. Las BDDs son muy usadas en las áreas de: inteligencia artificial, sistemas expertos, representación del conocimiento, tecnología de agentes, sistemas de información, integración de datos, meteorología, etc.

Existe una importante relación entre BDDs y programación lógica. Una BDD es, en esencia, un programa lógico; mapeo de relaciones base hacia hechos, y reglas que son usadas para definir nuevas relaciones en términos de las relaciones base y el procesamiento de consultas.

**Representación de la información**

Conjunto de aserciones, o hechos, conocidos como la base de datos extensional (EDB). Los hechos se especifican de manera similar a como se especifican las relaciones, excepto que no es necesario incluir los nombres de los atributos. Recordemos que una tupla en una relación describe algún hecho del mundo real cuyo significado queda determinado en parte por los nombres de los atributos. En una BDD, el significado del valor del atributo en una tupla queda determinado exclusivamente por su posición dentro de la tupla.

Conjunto de reglas (axiomas), referidos como la base de datos intencional (IDB). Las reglas se parecen un poco a las vistas relacionales. Especifican relaciones virtuales que no están almacenadas realmente, pero que se pueden formar a partir de los hechos aplicando mecanismos de inferencia basados en las especificaciones de las reglas. La principal diferencia entre las reglas y las vistas es que en las primeras puede haber recursión y por tanto pueden producir vistas que no es posible definir en términos de las vistas relacionales estándar.

Un sistema de inferencia tiene hechos (conocimiento explícito) y reglas (conocimiento implícito) que forman información.

La EDB se encuentra generalmente almacenada en una base de datos relacional, y constituye el conocimiento básico de las BDDs. Las reglas permiten obtener o deducir nuevo conocimiento a partir de la EDB, conocimiento que no se encuentra almacenado directamente en la EDB. De esta manera, las BDDs permiten inferir nuevo conocimiento, el cual puede ser usado para la toma de decisiones importantes del negocio.

Los hechos terminan con punto.

La coma significa AND. El punto y coma es OR.

Las mayúsculas son variables. Las minúsculas son constantes.

Ejemplo horóscopo:

*Hechos*: horoscopo(Signo,DiaIni,MesIni,DiaFin,MesFin)

horoscopo(aries,21,3,21,4).

horoscopo(tauro,21,4,21,5).

…

*Reglas*:

signo(D,M,X) : - horóscopo(X,Di,Mi,Df,Mf) , Di <= D , Mi = M.

signo(D,M,X) : - horóscopo(X,Di,Mi,Df,Mf) , Df >= D , Mf = M.

signo(Dia,Mes,Signo) : - horoscopo(Signo,D1,M1,D2,M2) , ((Mes=M1,Dia>=D1) ; (Mes=M2,Dia=<D2)).

Las consultas se hacen verdaderas con la primera regla o hecho que encuentra, pero devuelve todo con lo que se hace verdadero. El orden es muy importante en la lógica.

Si hay consultas con variables, se devuelve la constante con la que se hace verdadero. Si no hay variables, sólo true o false.

**Manejo de BDD**

En un sistema de BDDs por lo regular se usa un lenguaje declarativo para especificar reglas. Con lenguaje declarativo se quiere decir un lenguaje que define lo que un programa desea lograr, en vez de especificar los detalles de cómo lograrlo.

Una máquina de inferencia (o mecanismo de deducción) dentro del sistema puede deducir hechos nuevos a partir de la base de datos interpretando las reglas. El modelo empleado en las BDDs está íntimamente relacionado con el modelo de datos relacional, y sobre todo con el formalismo del cálculo relacional. También está relacionado con el campo de la programación lógica y el lenguaje Prolog. Los trabajos sobre BDDs basados en lógica han utilizado Prolog.

Con un subconjunto de Prolog llamado Datalog se definen reglas declarativamente junto con un conjunto de relaciones existentes que se tratan como literales en el lenguaje. Aunque la estructura gramatical se parece a la de Prolog, su semántica operativa (esto es, la forma como debe ejecutarse un programa en Datalog) queda abierta.

**Limitaciones del modelo relacional (sql92)**

Definición de vistas: Limitaciones en la definición de vistas recursivas

Actualización: Limitaciones en la actualización de las vistas

SGBD relacionales: Ausencia de procedimientos para la evaluación de consultas recursivas

**Base de datos deductivas**

Los sistemas de gestión de bases de datos deductivas deben superar las limitaciones de los sistemas relacionales. Los problemas anteriores se resuelven con la lógica.

Semántica de una BDD

Definir el conocimiento existente en la base de datos.

¿qué es cierto en la BDD?

Semántica declarativa: conocimiento en la BDD

Semántica operacional: procedimiento para obtener el conocimiento.

Ventajas

* Tener la capacidad de expresar consultas por medio de reglas lógicas.
* Permitir consultas recursivas y algoritmos eficientes para su evaluación.
* Contar con negaciones estratificadas.
* Soportar objetos y conjuntos complejos.
* Contar con métodos de optimización que garanticen la traducción de especificaciones dentro de planes eficientes de acceso.
* Como característica fundamental de una Base de Datos Deductiva es la posibilidad de inferir información a partir de los datos almacenados, es imperativo modelar la base de datos como un conjunto de fórmulas lógicas, las cuales permiten inferir otras fórmulas nuevas.

Desventajas

* La explotación de las reglas de deducción en una BDD plantea algunos problemas:
* Encontrar criterios que permitan, para una ley dada; decidir su utilización como regla de deducción o como regla de coherencia.
* Replantear correctamente, en un contexto deductivo, las convenciones habituales en una base de datos (representaciones de informaciones negativas, eficacia de las respuestas a las interrogaciones, cierre del dominio).
* Desarrollar procedimientos eficaces de deducción. La posibilidad de caer en bucles infinitos es un problema muy importante.

Marco Formal: Lógica de 1er orden (Programación Lógica)

Esquema de BDD:

(L, RI): - L es un lenguaje de 1er orden

- RI es un conjunto de f.b.f de L (restricciones de integridad)

BDD: (programa lógico)

{A: A es un átomo base} (hechos)

{ A ← L1 ^ L2 ^ ... ^ Ln : A es un átomo y Li es un literal} (reglas)

**Data Warehouses - Apoyo a la toma de Decisiones - Recuperación de Información valiosa**

**“Nuevos Tiempos” en las organizaciones**

Tres fuerzas impulsan a las compañías a producir un cambio.

Las tres C:

Clientes: asumen el mando. Cambia la fuerza dominante en la relación Vendedor-Cliente. Antes los clientes se adaptaban. La oferta era poca. Hoy la empresa se adapta al cliente. La oferta es mucha. Los Clientes exigen personalización. Mayor diversificación. La tecnología es la gran impulsora del cambio.

Competencia: se intensifica. Se diversifica sobre bases competitivas totalmente distintas. No existen territorios protegidos de competencia extranjera. Un competidor eficiente sube el umbral competitivo para los demás. Nuevos competidores. La tecnología cambia la naturaleza de la competencia.

Cambio: se vuelve constante. General y permanente. La rapidez del cambio tecnológico promueve la innovación. Cortos ciclos de vida de los productos. Disminuye el tiempo disponible para innovar.

**¿Qué sucede?**

* La división del trabajo y la especialización de las tareas estratificaron los procesos separando aquellas actividades relacionadas a la producción de aquellas asociadas a la gestión.
* Podemos categorizar entonces a los procesos en algún punto en un continuo entre dos extremos:
* Los completamente estructurados.
* Los completamente no estructurados.
* Paradójicamente, mientras la evolución de las organizaciones se mueve hacia ambientes menos estructurados, los sistemas han evolucionado en general dentro de los ambientes estructurados.
* El volumen de información empresarial se encuentra en un crecimiento constante. Muchos datos están dejando de ser estudiados debido a la incapacidad de los sistemas actuales para procesarlos.
* Se han generado y se generan muchos datos. El desafío es encontrar la información que hay oculta y darle un significado.
* La tecnología hoy dejo de ser una facilitadora para ser una posibilitadora de la administración de la información de los procesos.
* La tecnología de la información está transformando la forma de hacer negocios y los negocios mismos.
* Las grandes innovaciones de la humanidad fueron tecnologías genéricas. Han producido éxito en la medida que fueron aplicadas exitosamente. Requirieron de siglos para ofrecer su máxima utilidad.
* Hoy indudablemente LA innovación pasa por la información en todas sus formas.
* Juegan entonces un papel preponderante los desarrollos que se hagan para aplicar esta innovación.
* Seguramente no va a tomar siglos obtener los máximos beneficios de estas tecnologías, pero todavía no sabemos el impacto final que va a tener.
* Se trata de estar atento y al día.

**"Data Warehouses" - Motivación**

En la mayor parte de las organizaciones productivas y de servicios las bases de datos operacionales han sido diseñadas para cubrir las necesidades de procesos eminentemente transaccionales.

Los sistemas transaccionales que operan sobre las bases de datos realizan consultas o "queries" sencillas, que pueden ser eficientemente respondidas por los administradores de bases de datos.

Desafortunadamente, no importa cuán eficientemente se hayan diseñado las bases de datos de una organización, generalmente no satisfacen las necesidades de información de los sistemas de soporte de decisión (SSD) y los sistemas de soporte para ejecutivos (ESS).

Una de las razones fundamentales de esta aseveración es que las bases de datos tradicionales no poseen información histórica y la estabilidad de datos que es necesaria para llevar a cabo el tipo de análisis que realizan quienes toman decisiones.

Ejemplos de Consultas de Nivel Estratégico y Táctico

* ¿Cómo afectó las ventas del producto X la promoción navideña, en comparación con la misma promoción del año anterior? ¿Fue la performance distinta en diferentes regiones?
* ¿Cómo influyó en la tasa de accidentes industriales el nuevo plan de capacitación de operarios? ¿El impacto fue diferente en las distintas plantas de la empresa?
* ¿Cuál es la relación entre la cantidad de material "scrap" generado y el número de diferentes productos manufacturados en un mes de operación? ¿Existe relación entre la cantidad de "scrap" generado y el rendimiento económico de un determinado proceso productivo?

Por qué las BD tradicionales son inapropiadas para soportar este tipo de consultas?

* Es necesario disponer de información histórica.
* Se requiere combinar datos extraídos de bases de datos diferentes, a menudo implementadas sobre distintos administradores tales como Oracle,Access, Informix, SQL Server, etc.
* Las bases de datos operacionales fueron concebidas para satisfacer requerimientos de seguridad y eficiencia de sistemas transaccionales y si se las emplea para soportar las actividades de sistemas de soporte de decisión, no sólo no proveerán respuestas satisfactorias, sino que además degradarán la performance de los sistemas transaccionales a los que naturalmente deben soportar.

¿Cómo apoyar la actividad de los Sistemas de Soporte de Decisión?

Históricamente, los sistemas de soporte de decisión se emplearon en forma muy limitada, ya que no se disponía de la información necesaria en un tiempo razonable.

Para salvar este inconveniente se propuso "congelar" determinados extractos para tareas de análisis. Estos extractos contenían información acerca de un conjunto de entidades seleccionadas en un determinado punto en el tiempo.

Mientras el empleo de una serie de extractos (fotos de la base de datos en el tiempo) resultaba más eficiente que utilizar la base de datos transaccional directamente, no proveía una solución integral y carecía de la flexibilidad necesaria para responder a preguntas "adhoc".

**Data Warehouses (DW)**

* Se concibieron para soportar las necesidades de información de los SSD y ESS, es decir dar apoyo a los niveles gerenciales medios y altos de una empresa.
* Una DW es un sistema administrador de bases de datos que existe separadamente de los DBMS que soportan las actividades operacionales, pero que se integra con los mismos.
* Los datos almacenados en una DW son no volátiles, o son mucho menos volátiles que en una base de datos tradicional, y en consecuencia permiten la realización de una variedad de análisis. Por ejemplo, mientras en una base de datos operacional se almacena el precio actual de una materia prima, en una DW se mantendrá información histórica del valor de este atributo de las materias primas.
* Las DW son bases de datos relacionales que soportan una amplia variedad de consultas, que pueden ser formuladas en diferentes formatos.
* Contienen cientos de tablas cuyo diseño se ha optimizado para las consultas típicas.
* Para su creación los datos son tomados de una variedad de fuentes de datos operacionales. Una vez transferidos, estos datos son filtrados para asegurar que los mismos tienen significado, que son consistentes y que poseen el nivel de exactitud adecuado. Luego son cargados en tablas relacionales que soportan una variedad de análisis y consultas y cuya estructura se ha optimizado para satisfacer aquéllas que se espera ocurran con mayor frecuencia.
* Base de Datos para dar soporte a todos los procesos de decisión de la organización.

**Características**

Orientación a temas

Es organizado en torno a TEMAS principales

Excluye datos no usados para tomar decisiones

Los PROCESOS DE NEGOCIO no son considerados en el diseño del DW.

Integración

Fuentes de Datos del DW

Datos Internos: Entorno operacional de la organización, Legacy Systems, DB operacionales, spreadsheet, etc.

Datos Externos

Relación con el tiempo

Fotografía del Dato:

* Horizonte tiempo - 5-10 años
* Clave con atributos de tiempo
* Una vez tomada la fotografía, el registro no se cambia.
* Se guardan datos ya procesados (edad, no fecha).
* Nunca se borran los datos.

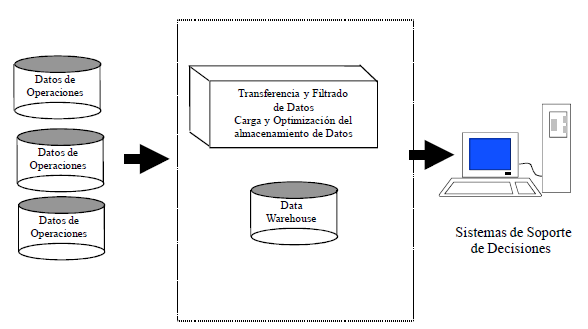
No volátil

El dato es cargado en el DW y accedido, pero una vez realizada la fotografía del dato, el dato en el DW no se modifica.

Consecuencias: Manejo de datos mucho más simple

* Elimina todo el proceso de actualización de datos en-línea: backup y recuperación, transacción e integridad, detección y solución de deadlock.
* Es posible optimizar el acceso a datos

**Representación Esquemática de la Creación y Operación de una Data Warehouse**



No todos los datos de las BD se pasan al DW (para optimizar consultas).

**Creación de una Data Warehouse**

La creación de una DW es un proceso difícil y altamente consumidor de tiempo, por lo que sus costos son generalmente elevados.

Mientras los procesos de transferencia de datos son sencillos, los procesos de identificación de datos relevantes, su "mezclado" y correcto filtrado son sumamente difíciles.

Con el propósito de hacer la base de datos más eficiente, los diseñadores enfrentan decisiones tales como sobre qué campos indexar los datos y cuál es el formato de indexado que es más conveniente desde el punto de vista del uso que se hará de los mismos.

Para la construcción de DWs se realizan numerosas entrevistas que tienen el propósito de definir las necesidades de todos los usuarios y de contemplar todos los posibles usos que podrían darse a los datos.

Una DW se construye en forma incremental.

El diseño de una DW no debe ser estático.

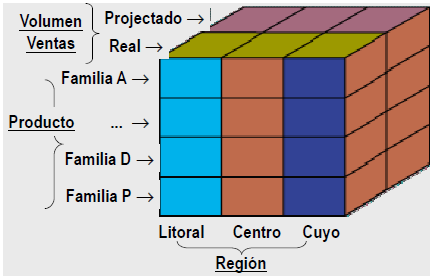
A medida que la cantidad de información se multiplica en el tiempo, los diseñadores de la DW puede que necesiten definir nuevas sintaxis, así como otros formatos de consulta que sean más rápidos y sencillos de utilizar.

De manera similar, a medida que aumenta la carga de información es menester implementar nuevos mecanismos de unión de tablas relacionales y de búsqueda de la información relevante.

Esta última actividad es también conocida como “data mining” y se lleva a cabo con la ayuda de una serie de tecnologías, entre las que merecen citarse: Técnicas de visualización, OLAP (On Line Analytical Processing,), Agentes Inteligentes, entre otras.

Las técnicas OLAP se emplean para realizar un modelado multidimensional de los datos, soportando la modelización y análisis de grandes volúmenes de datos desde diferentes perspectivas.

**Modelo de Análisis Multidimensional**



**Data Mining y Agentes Inteligentes**

A pesar que las DW proveen acceso a información que va a ayudar a los tomadores de decisiones a comprender su ambiente de trabajo, la gran variedad de oportunidades que brindan puede ser contraproducente.

Puede que los usuarios no experimentados sean sobrepasados por las capacidades de la DW y requieran de herramientas especializadas que los ayude a encontrar la información que ciertamente es de valor.

Los agentes inteligentes pueden ayudar a hallar los datos que mejor se ajustan a un determinado patrón, a encontrar la mejor alternativa entre un conjunto de posibilidades, a identificar los casos que se ajustan a un determinado criterio, etc. También pueden analizar grandes volúmenes de información para hallar patrones y descubrir tendencias.

Los agentes inteligentes son "software" que llevan a cabo tareas específicas, en lugar de los usuarios, actuando a menudo en forma autónoma, es decir realizando la mayoría de sus actividades sin intervención humana.

No son nuevos, ya que desde hace mucho tiempo se los ha empleado para el monitoreo de las capacidades y del uso que se hace de la CPU, así como para clasificar y priorizar la correspondencia electrónica recibida. Actualmente, los más populares son los agentes vinculados a Internet

Sin embargo, los agentes inteligentes modernos tienen nuevas y mejores capacidades para bucear en los datos ya que en su definición se han incorporado una variedad de herramientas de Inteligencia Artificial (Redes Neurales, Herramientas para Reconocimiento de Patrones, Razonamiento basado en casos).

**Agentes Inteligentes**

Los agentes inteligentes trabajan en diferentes dominios del conocimiento y con diferentes aplicaciones informáticas (Ayudantes de correo electrónico, personalización de páginas Web, etc.). Existen agentes inteligentes que operan en conjunción con una DW y su empleo debe ser programado. Pueden ser invocados de diferentes formas:

* de manera eventual, cuando el usuario lo considera apropiado.
* periódicamente, de acuerdo a una determinada agenda o a un "schedule" de trabajo
* en base a la ocurrencia de eventos predeterminados, tales como el alcance un determinado valor por parte de un indicador, la aplicación de una determinación promoción, etc.

Muchos agentes inteligentes, basados en diferentes metodologías (estadísticas, redes neuronales, técnicas de componentes principales, etc.), proveen una variedad de opciones para "scanear" la DW y buscar información. Por ejemplo, los usuarios pueden definir filtros basándose en diferentes aspectos, tales como: criterios de calificación, percentiles, información ordinal o basada en "rankings" o jerarquías.

**Posibles Resultados de Actividades de “Data-Mining”**

Asociaciones: Se corresponden con eventos vinculados a una situación o a un criterio. Por ejemplo, artículos que la gente suele comprar simultáneamente, o cursos optativos que los alumnos suelen tomar simultá-neamente.

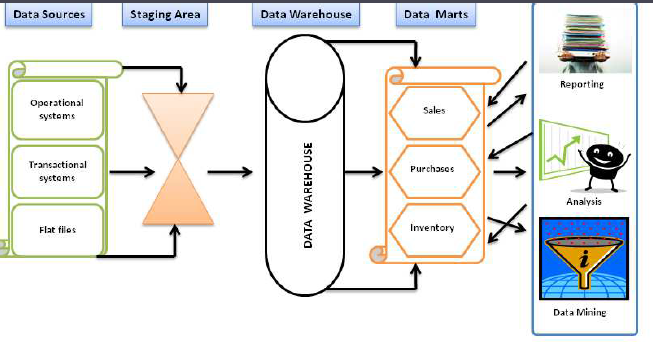
Secuencias: Se corresponden con una serie de eventos que tienen lugar secuencialmente en el tiempo. Por ejemplo, la secuencia de cursos optativos que los buenos alumnos de una carrera toman a lo largo de sus estudios, en un esquema de cursado con créditos; secuencias de eventos que ocurren en un ambiente industrial.

Clasificaciones: Se trabaja sobre un grupo de "individuos" muestrales que ya han sido clasificados (por ejemplo, personas con un ingreso per cápita superior a un determinado monto, o inferior a un valor, alumnos con un promedio de notas superior a un valor umbral, etc.) y se trata de identificar patrones de comportamiento en ellos, es decir valores comunes en determinados atributos.

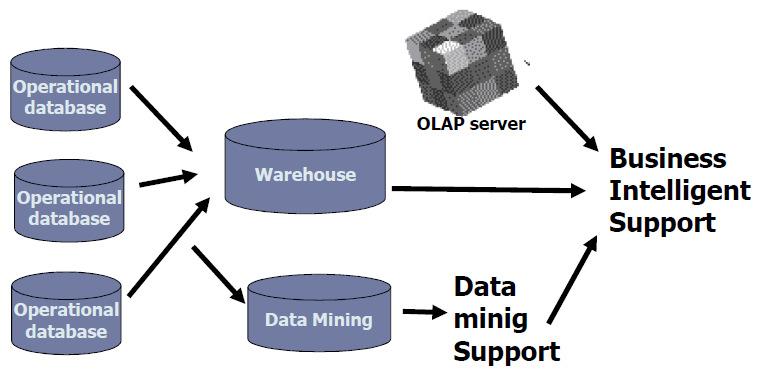
Agrupaciones o “Clusters”: Un proceso similar al anterior es el de "clustering", en el que se trata de inferir reglas sobre grupos, que se distinguen de otros grupos. La diferencia entre clasificación y "clustering" se basa en el hecho que, en el primer caso, las categorías se conocen de antemano (Por ejemplo, las personas de ingresos altos, medios y bajos), mientras en que en el caso de "clustering" los agentes inteligentes también identifican los diferentes grupos, es decir deben definir un agrupamiento y encontrar reglas de comportamiento para cada uno de los grupos identificados.

Pronósticos: Es la más conocida y la que se ha realizado por más tiempo con diferentes tipos de herramientas.

**Arquitectura de un DW**



**Soporte Inteligente para Negocios (BI)**



**Criterios de Evaluación de Productos para “Data-Mining”**

* ¿Qué enfoque emplea la herramienta para modelar los datos?
* ¿Qué tipo de problemas permite abordar?
* ¿Qué “hardware” y sistema operativo demanda?
* ¿Requiere un extracto de la DW o puede leer directamente de ella? ¿En caso afirmativo, sobre qué DBMS debe estar soportada para que pueda hacer la lectura de información?
* ¿Qué tipo de interfaz de usuario posee para realizar el ingreso y la interpretación de datos? ¿Es fácil de utilizar y comprender?
* ¿Cuál es el máximo número de variables y filas que soporta?
* ¿Cuán sensible a “ruidos” es la herramienta? ¿A qué tipo de “información corrupta” es particularmente sensible? ¿En qué extensión?
* ¿Cuál es el tiempo de respuesta promedio requerido para obtener resultados útiles?
* ¿Qué tipo de resultados puede entregar? (Asociaciones, Secuencias, Clasificaciones, pronósticos, etc.)
* ¿Cuán claros son los resultados?
* ¿Qué tipo de documentación está disponible?
* ¿Posee ayuda “on-line”?
* ¿Qué soporte a usuarios provee?

Es posible apreciar que muchos de los criterios de evaluación son los que se emplean cuando se analizan diferentes herramientas informáticas provistas por empresas de software.