**U5: BD ORIENTADA A OBJETOS**

Problemática

Los modelos de BD tradicionales han sido capaces de satisfacer las necesidades en cuanto a BD de las aplicaciones de gestión tradicionales. Sin embargo, presentan algunas deficiencias cuando se trata de aplicaciones más complejas o sofisticadas (diseño y fabricación en ingeniería, CAD/CAM, CIM, experimentos científicos, sistemas de información geográfica o 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

* Los Lenguajes OO poseen aceptación en la práctica.
* La programación en el paradigma OO sobre DBMS relacionales requiere de un mapeo o manejo de referencias entre objetos.

Las BDOO se crearon para tratar de satisfacer las necesidades de estas nuevas aplicaciones. La OO 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 BD tradicionales. Característica clave de BDOO: 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, Objectivity/DB, ONTOS, VERSANT, GemStone, Jasmine, DB4Object, Eloquera.

**ODMG (Object Database Management Group)**

Surgió la necesidad de establecer un modelo estándar de sistemas de BDOO y un lenguaje. Los fabricantes de los SGBD OO formaron un grupo denominado ODMG (*Object Database Management Group*), que propuso el estándar **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. También proporcionan interoperabilidad, permitiendo que una aplicación pueda acceder a varios sistemas diferentes. Además, los estándares 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 (mapeadores).

**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. El comportamiento es parte de la entidad en sí, por lo que en cualquier lugar en el que se la utilice, 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.

Objeto

* 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.
* Gran sentido de la privacidad: 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.
* OID (Object Identifier): 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 eliminado, 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.
    - Bag: conjunto de objetos sin orden.
    - Set: conjunto de objetos sin orden y sin objetos repetidos.
    - Lista: conjunto de objetos con orden sin tamaño definido.
    - Array: conjunto de objetos con orden con tamaño definido.
    - Tupla: tipo estructurado (struct).

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.
* 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. Esto permite 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. 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. Así, objetos externos pueden acceder a los datos encapsulados.
* *Mutador*: Almacena un nuevo valor en un atributo. Asi, 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 necesita generar nuevas versiones de algunos objetos sin descartar los existentes (diseño). Muchos ODBMS tienen sistemas que facilitan la generación de versiones del objeto.

**Persistencia de 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 (o por alcanzabilidad): el sistema infiere cuáles objetos son persistentes, por el hecho de estar asociados a otros objetos persistentes. Si un objeto es persistente, todo objeto al cual referencie también lo es.

**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.

Son complejos porque requieren gran cantidad de almacenamiento, que puede recuperarse por porciones. También lo son porque no es posible generar una condición de selección directa sobre estos objetos basada en sus valores.

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

**Objetos complejos estructurados**

Por ejemplo, una tupla que su vez puede estar compuesta por otras tuplas o sets (listas, arreglos), y así sucesivamente.

Maneras de referenciar objetos:

* + - *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 (Object Data Management Group)**

Es el estándar de ODBMS. Objetivo: 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 BD basado en SQL.

Enlaces (bindings) con LPOO: Smalltalk. Java. C++ OML (lenguaje de manipulación de objetos): C++ ODL, mecanismos para ejecutar sentencias OQL, mecanismos para ejecutar operaciones sobre la BD y manipular transacciones.

**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 ordenados o desordenados, dependiendo el tipo de colección que se especifica.

ODBMS es el responsable de mantener la integridad referencial: si el objeto es borrado, todo camino que accede al objeto también lo es. 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 de nombres de 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 (“extent”)**

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

**ODMG - OQL**

Una BD puede ser accedida de diferentes formas:

* Navegacionalmente: el programa navega de un objeto a otro utilizando las referencias entre objetos.
* 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.
* Provee primitivas para manipular colecciones.
* 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 13 características obligatorias y 4 opcionales. Una BD es OO cuando cumple con los 13 puntos del manifiesto. Las BD relacionales pueden ser BDOO si cumplen con los 13 puntos.

Características obligatorias:

1. Deben soportarse objetos complejos.
2. Deben soportarse mecanismos de identidad de los objetos.
3. Debe soportarse la encapsulación.
4. Deben soportarse los tipos o clases.
5. Los tipos o clases deben ser capaces de heredar de sus ancestros.
6. Debe soportarse el enlace dinámico.
7. El DML debe ser computacionalmente complejo.
8. Debe proporcionarse persistencia a los datos.
9. Debe ser capaz de gestionar BD de muy gran tamaño.
10. Debe soportar usuarios concurrentes.
11. Debe ser capaz de recuperarse de fallos hardware y software.
12. Debe proporcionar una forma simple de consultar los datos.
13. Debe soportar consultas ad hoc./El conjunto de todos los tipos de datos debe ser ampliable.

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

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 que
  + 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.
  + Reducción de redundancia mediante la agrupación de propiedades comunes a diversas clases en una superclase.
  + Reusabilidad de clases: 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 BD. Hay muchas áreas en las que los SGBD tradicionales no han tenido excesivo éxito, pero las capacidades de modelado de los SGBDOO sí resultan efectivas.
* Mayores prestaciones. Los SGBDOO proporcionan mejoras significativas de rendimiento con respecto a los SGBD relacionales.

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 que se tiene en 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. El modelo relacional tiene una sólida base teórica y muchas herramientas de soporte.
* 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 BD de manera eficiente.
* El modelo de objetos aún no tiene una teoría matemática coherente que le sirva de base.

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

El término BD Objeto Relacional (BDOR) se usa para describir una BD 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 BD. 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 BD relacionales de Oracle, desde versión 8.i, ha sido significativamente extendido con conceptos del modelo de BDOO. 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.

**Mapeo Objeto Relacional (ORM)**

* Los ORM son herramientas de SW que permiten trabajar con los datos persistidos en BD relacionales como si fueran parte de una BDOO (virtual).
* En aplicaciones estándar realizadas sobre BD relacionales, la función del ORM es transformar un registro en objeto y viceversa, con el fin de realizar operaciones de consulta y persistencia directamente sobre los objetos.
* Existen varios componentes que se pueden utilizar a tal fin en tecnología .NET.
* 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.

¿Cuándo usar ORM?

Usar un ORM si:

* Se tiene la posibilidad de diseñar la BD. Así, podremos seguir prácticas que facilitarán la interacción ORM-DB.
* Se desea abstraerse del DBMS y poder cambiarlo sin demasiados inconvenientes.

Evitar utilizarlo cuando:

* Se reciban BD 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.
* Se necesiten 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 BD convencionales se consideran pasivas en el sentido de que es el usuario o el programa de aplicación quien decide qué hacer en la BD y ella no reacciona a las acciones ejecutadas sobre ella. Las BD activas buscan reaccionar ante ciertos eventos. Comportamiento Activo:Cuando sucede algo, hacer algo.

**Definición BDA**

Una BDA es aquella BD 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 BD, las reglas que se utilizan para especificar estas situaciones y sus acciones se llaman reglas que siguen el paradigma de evento-condición-acción.

El formato genérico de estas reglas es: ON evento IF condición THEN acción.

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

**Semántica de la especificación de reglas**

* Granularidad del procesamiento de las reglas: Determinar si una regla se dispara una vez por cada tupla “tocada” o una sola vez por todas las tuplas “tocadas” o al final de la transacción que se está ejecutando cuando se lanzó la regla.
* Anidamiento de reglas y terminación: Determinar si se puede especificar una sola regla por evento o más de una. Si se permite más de una regla por evento, en cuál orden se ejecutan y cuándo se termina la ejecución.
* Concurrencia con las transacciones: Determinar si las reglas se van a ejecutar como parte de la transacción donde se disparen o como transacciones aparte. Esto es fundamental por la propiedad de atomicidad que se debe garantizar.

**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 BD. Ejemplos: control de integridad, mantenimiento de vistas y datos derivados, administración de copias de los datos, seguridad, etc.
* Externas: Estas aplicaciones contienen conocimiento de la aplicación, expresado en la forma de 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 BD. Por ejemplo, las restricciones de dominio.
* *Restricciones dinámicas*: se evalúan sobre la transición de un estado a otro. Por ejemplo, el sueldo de un empleado sólo puede aumentar.

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.

**Estándar de SQL y BDA**

El primer estándar de SQL donde se consideró la noción activa de una BD fue en SQL:99 (SQL 3). El estándar que salió en 1992 no 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 sí implementaron este aspecto, pero como no había estándar, cada uno realizó su implementación independientemente, obteniendo así una gran diversidad de implementaciones. La descripción de constraints, triggers y assertions si se encuentra en SQL:99, por lo tanto, ahora se quiere tratar de uniformizar todas las implementaciones existentes.

**Constraints**

Son especificaciones del SQL que se aplican a columnas o tuplas de una tabla. Algunos tipos especiales son: UNIQUE, NOT NULL, REFERENCES, CHECK. El CHECK permite especificar una amplia gama de reglas, como rangos de valores y listas de valores, entre otros. Es conveniente que cada constraint tenga un nombre porque 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 durante la ejecución de la transacción (constraint check time IMMEDIATE) 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.

**Assertions**

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

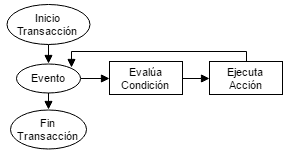
Los assertions son útiles cuando es necesario especificar una condición general de la BD que no se puede asociar a una tabla específica de la BD. Esto tiene grandes ventajas a nivel conceptual, pero las hace difíciles de implementar. La reacción ante un assertion siempre es abortar.

**Triggers**

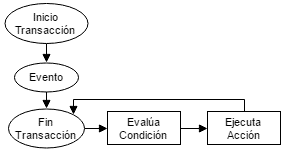
Los triggers proveen mayor flexibilidad que los constraints ya que el usuario puede especificar un conjunto de acciones complejas a ejecutarse cuando el trigger se dispare. Los constraints actúan cuando se viola lo que ellos indican, pero el usuario no puede especificar acción alguna con esa violación,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

**Modos de Acoplamiento**

Modo de acoplamiento Inmediato: Se evalúa la condición y se ejecuta la acción cuando ocurre el evento.



Modo de acoplamiento diferido: Se evalúa la condición y se ejecuta la acción cuando finaliza la transacción.



**Manifiesto de los SGBDA**

* Un SGBDA tiene un modelo de reglas ECA (evento-condición-acción).
* Un SGBDA debe soportar la gestión de reglas y la evolución de la base de reglas.
* Un SGBDA tiene un modelo de ejecución de reglas ECA.
* 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.

**Definición BDT**

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

Permite mantener historia de los datos, pero consume más almacenamiento porque los datos no se modifican ni eliminan (crece en forma exponencial). Las consultas en estas BD son complicadas y tienen peor desempeño.

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
* BD 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 una 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. de duración). Los investigadores prefieren hablar de cronos. La “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 con diferentes puntos de referencia, pero, por lo general, 60 segundos, 60 minutos, 24 hs., 7 días, etc.
* Definir el tipo de dato: 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 eventos o sucesos son puntos (momentos únicos) o un evento durable (período). Las BDT guardan información de cuando ocurre un evento o cuando ciertos hechos se consideran verdaderos:

* Eventos puntuales: están asociados en la BD con un punto de tiempo de alguna granularidad (ej.: timestamp).
* Eventos durables: están asociados con un período de tiempo en la BD. Se representa con un 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**

Algunas aplicaciones pueden necesitar sólo una de las dimensiones, si necesitan ambas son BD Bitemporales.

* Tiempo válido: que sea verdadero en el mundo real.
* Tiempo de transacción: momento en que la información fue registrada.

Relaciones de tiempo válidas

VST = Tiempo válido de comienzo (Valid Start Time)

VET = Tiempo válido de fin (Valid End Time)

Los datos no se actualizan ni se eliminan, sino que se genera una nueva versión (se cambia el VET) indicando que esa versión de tupla está cerrada. Cada tupla de una tabla representa una versión que es válida en ese período de tiempo.

* *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.

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.

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 válidas tratan de representar la historia del mundo real, dado que las BD 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

TST = Tiempo transaccional de comienzo (Transaction Start Time)

TET = Tiempo transaccional de fin (Transaction End Time)

Cada vez que hay un cambio en la BD 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.

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: Sistemas de Información Geográfica (SIG)**

**Introducción**

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 BD de información geográfica y en sistemas SIG. El uso de los SIG ha aumentado enormemente en las últimas décadas pasando del total desconocimiento a la práctica cotidiana.

Los factores que han fomentado el desarrollo de los SIG son: el abaratamiento de los equipos informáticos, el uso de la información geográfica desde múltiples ámbitos para ayudar a la toma de decisiones y la sociedad de la información que demanda cada vez más datos de cualquier ámbito.

**Definición SIG**

Es una tecnología de manejo de información geográfica. La esencia de un SIG está constituida por una BD geográfica.

Es una colección de datos acerca de objetos localizados en una determinada área de interés en la superficie de la tierra, organizados en una forma tal que puede servir eficientemente a una o varias aplicaciones.

Una BD geográfica requiere de un conjunto de procedimientos que permitan hacer un mantenimiento de ella, tanto desde el punto de vista de su documentación como de su administración.

La eficiencia está determinada por los diferentes tipos de datos almacenados en diferentes estructuras. El vínculo entre las diferentes estructuras se obtiene mediante el campo clave que contiene el número identificador de los elementos.

**Arquitectura de un SIG**

* HW: Equipos informáticos donde opera el SIG.
* SW: Programas que proveen funciones y herramientas necesarias para almacenar, analizar y desplegar la información geográfica. Los principales componentes son: herramientas para la entrada y manipulación de la información geográfica, DBMS, Herramientas para búsquedas geográficas, análisis y visualización, y GUI.
* Datos geográficos: Es la parte más importante. La realidad geográfica es muy compleja debido a la diversidad de las partes que la forman, al mundo en continuo cambio y a que hay un gran número de relaciones.

Atributos de la información geográfica:

* *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.

BD geográfica o espacial: Es el conjunto de datos geográficos relacionados entre sí. Puede utilizar los manejadores de BD más comunes para manejar la información geográfica. El SIG integra datos espaciales (cartografía) con otros datos (BD temáticas).

Tipo de datos de representación:

* *Puntos*: entidades que son demasiado pequeñas para ser mostradas como áreas o líneas. (un cable telegráfico o una ciudad, dependiendo de la escala).
* *Líneas*: conjunto de puntos ordenados, utilizados para representar entidades que son demasiado pequeñas como para ser representadas como áreas (río).
* *Áreas o Polígonos*: figuras planas con límites de líneas. (campos, áreas administrativas, ciudades).
* Recurso humano: Personal que opera, desarrolla y administra el sistema. Establece los planes para aplicarlo en el 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.

**Funciones de un SIG**

Captura de la información: normalmente mediante 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.

**Modelado de 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 que 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 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.

El modelo de datos es el conjunto de reglas a utilizar para formar representaciones del territorio en un entorno digital y discreto. 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.

Los datos espaciales han de codificarse como entidades discretas para poder ser almacenados en ordenadores mediante puntos, líneas o áreas. 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 suelen estar relacionados, los cuales son:

* + Gráficos: Son las representaciones de los objetos geográficos asociados con ubicaciones específicas en el mundo real, usando puntos, líneas o áreas.
  + Alfanuméricos: Se corresponden con las descripciones o características de los objetos geográficos a los que se refieren.

Existen dos tipos de relaciones:

* + La relación posicional, que informa dónde está el elemento respecto de un sistema de coordenadas establecido.
  + La relación topológica, que informa 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.

Los modelos de datos de información geográfica utilizan como modo de representación de la información geográfica sistemas vectoriales o raster:

* 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.
* 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.

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

Necesidad de disponer de datos adecuados en cantidad, calidad y forma. La preparación de los datos de un SIG tiene alto coste en cuanto a captura, edición, estructuración y reestructuración. La calidad del valor de los resultados dependerá de la calidad de los datos.

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, 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 BD para una aplicación SIG

1. Modelo conceptual.
2. Definición de objetos y relaciones.
3. Selección de la representación geográfica.
4. Modelo lógico de la BD.
5. Organización y estructura de la BD.

**Uso y aplicación de los SIG**

La utilidad principal de un SIG radica en su capacidad para construir modelos o representaciones del mundo real a partir BD espaciales. Algunas de sus aplicaciones principales son:

* + 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.
  + Oracle Spatial, PostgreSQL + Postgis (módulo que añade soporte de objetos geográficos, convirtiéndola en una BD espacial).

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)

Madurez de los SIG: (Factores)

1. Grado con que las funcionalidades ofrecidas por los GIS corresponden al tipo de operaciones que se les exigen. Vacío entre las necesidades del usuario y lo que los SW GIS pueden ofrecer.
2. Facilidad de uso: Se han añadido distintos tipos de herramientas a los SW GIS que permiten a los usuarios construir su propia interfaz especializada.
3. Inversiones en software y hardware. Su coste no es un gran obstáculo.
4. La educación y el conocimiento. Aumento de la involucración de las personas y las disciplinas con los GIS.
5. Los datos. Problemas relacionados a disponibilidad, coste, estándares, exactitud y las obligaciones legales. Inversiones altas. El coste de los datos es ahora el factor más firme que dificulta el uso de información geográfica.

**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 BD Relacionales, aparecen recientes aplicaciones de los sistemas de BD que consisten 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 BD Relacionales los beneficios de la lógica como instrumento para la formalización integrada de los aspectos estáticos y dinámicos del modelado de aplicaciones.

**Definición BDD**

Las BDD extienden la capacidad expresiva de las BD relacionales, incluyendo un conjunto de reglas que permiten definir conocimiento implícito.

Una BDD está compuesta por una BD relacional, que es el conocimiento explícito, y por reglas deductivas que son el conocimiento implícito.

Existen diversas clases de BDD y para cada una de ellas existe una semántica bien definida. Las BDD 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 BDD 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 hechos: conocidos como la BD 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.

nombre\_hecho(datos).

* Conjunto de reglas: referidos como la BD intencional (IDB). Son axiomas, es decir, reglas que se toman como verdad. 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.

regla(Datos) : - hecho() , condiciones.

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

La EDB se encuentra generalmente almacenada en una BD relacional, y constituye el conocimiento básico de las BDD. 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 BDD permiten inferir nuevo conocimiento, el cual puede ser usado para la toma de decisiones importantes del negocio.

**Semántica de una BDD**

* Definir el conocimiento existente en la BD.
* Semántica declarativa: conocimiento en la BDD. Hechos.
* Semántica operacional: procedimiento para obtener el conocimiento. Reglas.

Ejemplo:

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

La “,” es AND. El “;” es OR. Las mayúsculas son variables, las minúsculas son constantes. Los hechos y reglas terminan con punto.

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

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

**Manejo de BDD**

En un sistema de BDD 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 BD interpretando las reglas. El modelo empleado en las BDD 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 BDD 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 (la forma como debe ejecutarse un programa en Datalog) queda abierta.

**Limitaciones del modelo relacional**

EL modelo relacional SQL92 tiene limitaciones en la definición de vistas recursivas y en la actualización de las vistas. Además, el SGBD relacionales tiene ausencia de procedimientos para la evaluación de consultas recursivas.

Los DBMS de BDD van a superar estas limitaciones de los sistemas relacionales mediante la lógica.

**Ventajas BDD**

* 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.
* Métodos de optimización que garantizan la traducción de especificaciones dentro de planes eficientes de acceso.
* Característica fundamental: inferir información a partir de datos almacenados. Es imperativo modelar la BD como un conjunto de fórmulas lógicas, las cuales permiten inferir otras fórmulas nuevas.

**Desventajas BDD**

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 BD (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.

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

**Motivación Data Warehouses**

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

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

Desafortunadamente, no importa cuán eficientemente se hayan diseñado las BD 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 BD tradicionales no poseen información histórica ni la estabilidad de datos que es necesaria para llevar a cabo el tipo de análisis que realizan quienes toman decisiones.

Las BD tradicionales no son apropiadas para soportar consultas de nivel estratégico y táctico, tales como:

* ¿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?

Para responder a esas cuestiones es necesario disponer de información histórica. Se requiere combinar datos extraídos de BD diferentes, a menudo implementadas sobre distintos administradores.

Las BD 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 BD en el tiempo) resultaba más eficiente que utilizar la BD transaccional directamente, no proveía una solución integral y carecía de la flexibilidad necesaria para responder a los problemas.

**Definición (DW)**

Se concibieron para soportar las necesidades de información de los SSD y ESS, es decir, para dar apoyo a los niveles gerenciales medios y altos de una empresa.

Un DW es un sistema administrador de BD 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 BD tradicional, y en consecuencia permiten la realización de una variedad de análisis. Por ejemplo, mientras en una BD operacional se almacena el precio actual de una materia prima, en un DW se mantendrá información histórica del valor de este atributo de las materias primas.

Los DW son BD 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.

Esta BD nació para dar soporte a todos los procesos de decisión de las organizaciones.

**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.

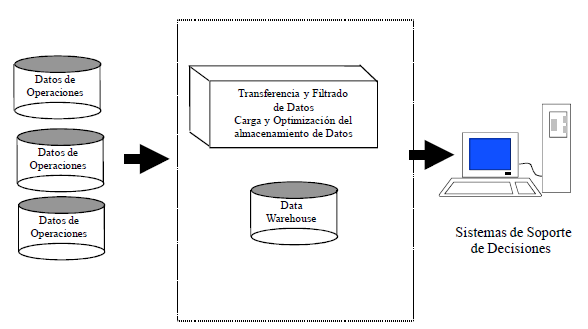
Relación con el tiempo: Fotografía del Dato:

* 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).

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. Nunca se borran los datos. La consecuencia de la no volatilidad es el 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: No se pasan todos los datos de las BD al DW.

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



**Creación de una DW**

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 BD 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. Para la construcción de DW 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.

Un 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 del 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 necesario 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.

**Data Mining y Agentes Inteligentes**

A pesar de que los 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 del 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 SW 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.

Existen agentes inteligentes que operan en conjunción con un 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; o en base a la ocurrencia de eventos predeterminados.

Muchos agentes inteligentes, basados en diferentes metodologías (estadísticas, redes neuronales, técnicas de componentes principales, etc.), proveen una variedad de opciones para "scanear" el 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 simultáneamente.

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: Pronosticar eventos. Es la más conocida y la que se ha realizado por más tiempo.