Base de Datos II

Unidad N° 6: Nuevas tendencias en BD

Objetivos

- Conceptos de Base de Datos Activas.
- Conceptos de BD Temporales.

3ra. Generación de BD

*"Proporciona capacidades de **gestión de datos** al igual que sus predecesoras, 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.

BD Activas - Motivación

¿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.
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.

- La solución 1 es un desastre en mantenimiento y constituye una mala práctica de ingeniería de software.
- Con respecto a la solución 2, ¿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 + QUE
- Ejemplos:
  - Gestión de Stock:
    - Cuando Item < 10 entonces Solicitar nuevo Item al Proveedor.
  - Productos Perecederos:
    - Cuando fecha_caducidad – today < 7 entonces reducir el precio del producto en un 10 %

Base de Datos Activa

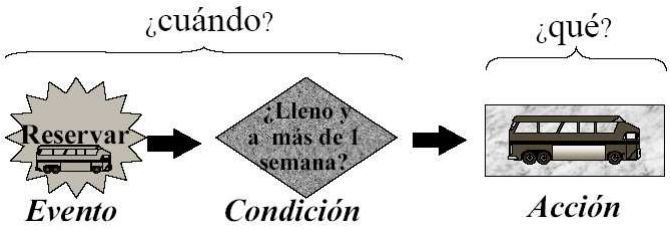
El paradigma de bases de datos activas planteado por Morgenstern en 1983, describe la noción de que una base de datos sea activa, como una metáfora de su comportamiento, el cual se concentra en:

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.

Bases de Datos Activa

- 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.

Bases de Datos Activas



Bases de Datos Activas

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

1. 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.

2. 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.

3. 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.

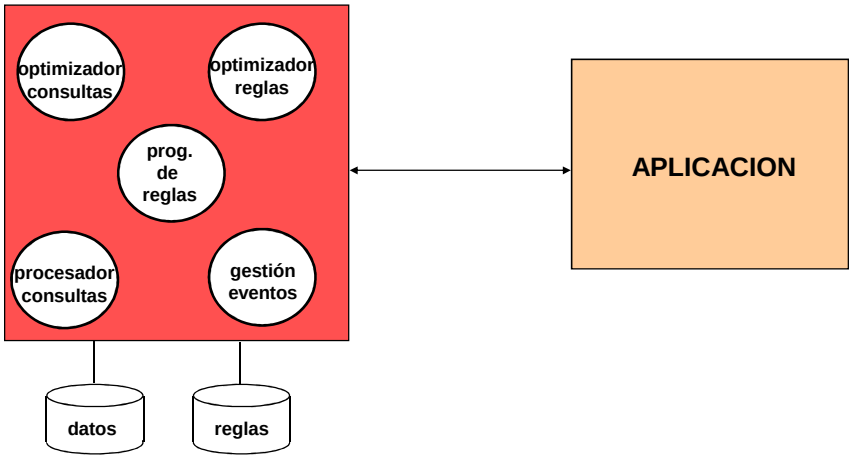
Aplicaciones de BD activas

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.

SGBD Activo



Aplicaciones de BD activas

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 especifica, 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), o
- 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.

Ejemplo:
```
CREATE ASSERTION un_presidente CHECK(
(SELECT COUNT(*) FROM emp WHERE
puesto='PRESIDENTE') < 2);
```

Assertions

```
CREATE ASSERTION as_salarios_similares
CHECK( ( (SELECT MAX(SAL) FROM EMP)-
(SELECT MIN(SAL) FROM EMP) ) <1500);

CREATE ASSERTION deptos_pequenos
CHECK (NOT EXISTS
(SELECT DEPTNO FROM EMP
GROUP BY DEPTNO
HAVING COUNT(*)>20)) ;
```

- La reacción siempre 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.

Componentes de un triggers

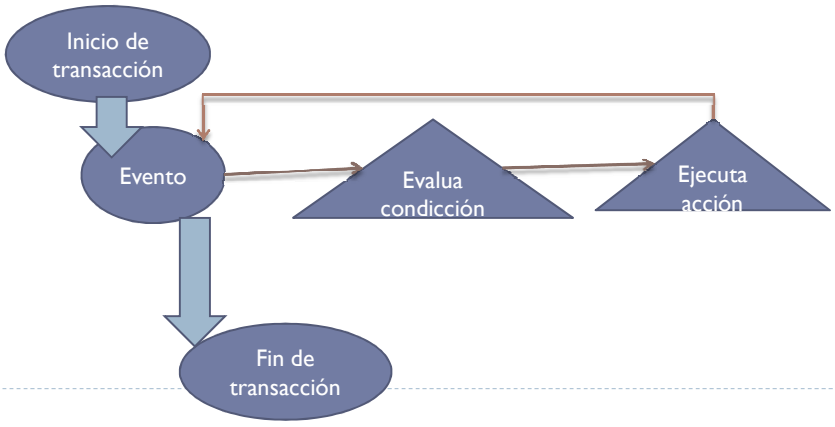
- Nombre, debe ser único en el esquema donde se define.
- Tabla sujeto, es el nombre de la tabla cuyos datos, al ser alterados, van a activar al trigger.
- Tiempo de la acción, le dice al DBMS cuándo ejecutar el cuerpo del trigger, lo cual puede ser antes (BEFORE) o después (AFTER) que ocurra el evento del trigger.
- Evento, indica cuál es el comando de actualización sobre la tabla, que va a activar el trigger, este evento puede ser uno o más de los siguientes: INSERT, DELETE y UPDATE.
- Lista de atributos, sólo se pueden especificar si el evento es un UPDATE.
- Aliases, tanto los valores de la tupla antes y después de la actualización, como las tablas pueden tener nombres que fingen de sinónimos de los nombres originales.
- Cuerpo del trigger, comandos de SQL que se ejecutan cuando el trigger se dispara.
- Sello de tiempo, indica cuándo fue creado el trigger.

Los triggers se definen con un comando de CREATE TRIGGER y para borrarlos se utiliza el comando DROP TRIGGER.

Modos de Acoplamiento

¿Cuándo se evalúa la condición?

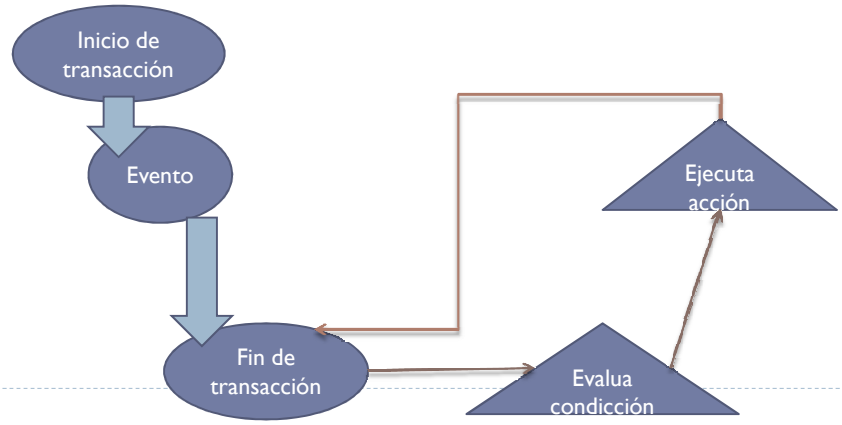
- Modo de acoplamiento Inmediato:



Modos de Acoplamiento

¿Cuándo se evalúa la condición?

- 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

Base de Datos II

Nuevas tendencias en BD – BD Temporales

BD Temporales - Motivación

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

Por ejemplo saber ¿Cuál es el sueldo actual de Bob?

Nombre	Salario	Titulo	F_Nac	Start_Date	Stop_Date
Bob	70000	Decano	01-01-70	01-12-99	
Bob	70000	Adjunto	01-01-70	01-11-97	30-11-99
Bob	70000	Asistente	01-01-70	01-01-96	31-10-97
Bob	60000	Asistente	01-01-70	01-01-95	31-12-95

```
SELECT salario
FROM Empleado
WHERE nombre="Bob"
AND Start_Date <= today() AND
today <=Stop_Date
```

Problemas en BD No Temporales

- ¿Como sacar la historia del salario en SQL ?
- ¿Como mezclar los tiempos en que el salario fue el mismo con SQL?
- Hay que emplear otras elementos de programación para sacar el resultado.

```
CREATE TABLE Temp(salario, start, stop)
AS SELECT salario, Start_Date, Stop_Date)
FROM Empleado
WHERE nombre="Bob"
repetir
UPDATE Temp T1
SET T1.stop =( SELECT MAX(T2.stop)
FROM Temp T2
WHERE T1.salario=T2.salario AND T1.start<T2.start
AND T1.stop>=T2.start AND T1.stop < T2.stop )
WHERE EXISTS ( SELECT *
FROM Temp T2
WHERE T1.salario=T2.salario
AND T1.start < T2.start AND T1.stop <= T2.stop
OR T1.start >=T2.start AND T1.stop < T2.stop )
hasta que las tuplas no se actualizan mas
```

Bases de datos temporales

- TDB: Toda aplicación de Bases de Datos que requiere de algún aspecto del tiempo para organizar su información.
- 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.

Representación del tiempo y calendario

- 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.
- 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.
- Calendario (existen varios: Gregoriano, Chino, Islámico, etc.) con diferentes punto de referencia pero, por lo general, 60 segundos, 60 minutos, 24 hs., 7 días…
- 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

BDT 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

Tiempo en BD Relacionales

EMP_VT(DNI, Nombre, Salario, Dno, DNIJefe, VST, VET)

DEPT_VT(Dno, Dnombre, Total_Sal, DNI_Mgr, VST, VET)

- granularidad: día

Relaciones de tiempo válidas:

- VST= Tiempo válido de comienzo (Valid Start Time)
- VET= Tiempo válido de fin (Valid End Time)

Tipo de dato: Fecha (DATE)

- cada tupla EMP_VT representa una versión del empleado que es válida en ese período de tiempo
- en la BD Relacional cada tupla de EMPLEADO representa el estado actual o la versión actual del empleado (now)

El tiempo en BD Relacionales
Relaciones de tiempo validas

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

El tiempo en BD Relacionales
Relaciones de tiempo validas

- Eliminación en las BD temporales es cerrar la versión actual de una tupla por medio de darle una valor de tiempo válido a VET.
- 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

El tiempo en BD Relaciones
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

  EMP_TT(DNI, Nombre, Salario, Dno, DNIJefe, TST, TET)

  DEPT_TT(Dno, Dnombre, Total_Sal, DNI_Mgr, TST, TET)
- 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)

Objetivos

- ✓ Conceptos de Base de Datos Activas.
- ✓ Conceptos de BD Temporales.