Bases de datos avanzadas

Universidad de Los Andes

Postgrado en Computación

Prof. Isabel M. Besembel Carrera

Unidad III. Sesión 22 y 23. Bases de Datos Activas.

UNIVERSIDAD DE LOS ANDES

Introducción …

- Las BD convencionales son **pasivas**, solo ejecutan consultas a petición de un usuario o de un programa de aplicación.
- Una BD **activa** es una BD que tiene la capacidad de monitorear situaciones de interés y cuando ellas ocurren disparan una respuesta adecuada.
- El comportamiento deseado se expresa por medio de reglas de producción o reglas evento-condición-acción (ECA), las cuales pueden ser definidas y almacenadas en la BD.

on evento	**Una regla se *dispara* cuando el evento ocurre**
if condición	**se *considera* cuando su condición se evalúa**
then acción	**y se *efectúa* cuando la acción se ejecuta**

- Las reglas serán disparadas por eventos de la BD como: un estado particular de la BD y cambios de estado de la misma.

UNIVERSIDAD DE LOS ANDES

Introducción

- Ellas pueden:
  - ✓ asegurar la integridad de datos,
  - ✓ implementar gatillos y alertadores,
  - ✓ mantener datos derivados,
  - ✓ asegurar las restricciones de acceso,
  - ✓ implementar las políticas de control de versiones,
  - ✓ mantener las estadísticas de acceso para la optimización de consultas,
  - ✓ etc.

UNIVERSIDAD DE LOS ANDES

Modelos de reglas

- En los SMBDR como Ariel, Postgres y Starburst, las reglas se definen como metadatos en el esquema.
- En SMBDOO como HiPAC, ellas son tratadas como objetos de primera clase que son instancias del tipo **Rule**.
- Este tipo tiene como estructura evento, condiciones y acciones, y como operaciones incluye: fire, enable y disable.

UNIVERSIDAD DE LOS ANDES

Especificación de los eventos

- Los más comunes son modificaciones de la BD, como: inserciones, eliminaciones y modificaciones.
- Relacional

  define rule MonitorNuevosEmpleados

  on insert to empleado

  if ....

  then ...

- Orientado por objetos

  define rule ChequearAumentos

  on empleado.aumentoSalario()

  if ....

  then ...

UNIVERSIDAD DE LOS ANDES

Lenguajes de reglas

- Algunos lenguajes permiten la definición de reglas basadas en recuperaciones, el disparo basado en eventos temporales y eventos compuestos.
- En SQL2 se permiten aserciones sobre las tablas que pueden ser disparadas según los eventos:
  - **before commit,**
  - **after insert,**
  - **after delete,**
  - **after update**.
- En SQL3 se introducen **gatillos** (triggers) además de las aserciones, denominados: **before y after**.
- En HiPAC, los eventos son operaciones genéricas en la BD (retrieve, insert, delete, update) u operaciones específicas de los tipos.

UNIVERSIDAD DE LOS ANDES

Especificación de las condiciones

- La parte condición de la regla especifica un predicado o una consulta en la BD.
- Algunas permiten condiciones sobre las modificaciones de la BD referidas a datos modificados o al estado predecesor del disparo del evento, este condición se denomina de transición.
- En SQL2, la condición se satisface si el predicado es falso.
- En HiPAC, la condición es un conjunto de predicados o consultas, que si son satisfechos y las consultas no vacías, la condición es verdadera.

UNIVERSIDAD DE LOS ANDES

Especificación de las acciones

- La parte de las acciones especifica las operaciones que se deben hacer cuando la regla es disparada y la condición satisfecha.
- En SQL2, la acción es implícita, si la condición se satisface la acción es abortar la Ti.
- En SQL3, la acción puede contener deshacer la Ti o puede estar basada en los datos de disparo.

  define rule nuevosEmpleados

  on insert to empleado

  then delete empleado e **where** e.nombre = empleado.nombre

- En Postgres, las acciones de las reglas pueden contener la palabra **instead** indicando que la acción puede ejecutarse en vez de la operación de disparo.

UNIVERSIDAD DE LOS ANDES

Enlace de las partes evento-condición-acción

- Cada regla tiene un mecanismo que la asocia a las tuplas insertadas, eliminadas o modificadas que ella referencia en sus condiciones y acciones.

  create trigger DptoEliminación

  before delete on departamento

  when departamento.presupuesto < 100.000.oo

  delete empleado **where** empleado.dptoNro = departamento.dptoNro

- En SQL2, SQL3 y Postgres se incluyen las palabras claves **old** y **new** para referenciar viejos y nuevos valores y ellos pueden usarse en la condición y/o en la acción.

UNIVERSIDAD DE LOS ANDES

Enlace de las partes evento-condición-acción

- En Ariel se usa la palabra **previous** para los valores viejos.

  define rule nuevosEmpleados

  on append to empleado

  then delete empleado **where** empleado.nombre = **new.**nombre

- En Starburst los cambios pueden ser referenciados en la condición y/o en la acción usando tablas de transición que son tablas lógicas.
- Una tabla de transición **inserted** contiene las tuplas insertadas al disparar la regla a tiempo de ejecución, asimismo para aquellas **deleted**, **new-updated** y **old-updated**.

  define rule promGrande

  on update to empleado.salario

  if (select avg(salario) **from new-updated) >** 100

  then rollback

UNIVERSIDAD DE LOS ANDES

Ordenación de las reglas

- SQL2 y SQL3 no permiten más de una regla asociada a un evento, por lo que no hay conflictos en la ejecución de las mismas.
- En la mayoría de los sistemas la escogencia de cual regla ejecutar, si hay más de una que puede ser disparada, se hace arbitrariamente.
- Algunos proveen mecanismos para influenciar el orden de ejecución.
- En Postgres hay prioridades numéricas y jerarquías de excepción pero definidas en teoría (no implementadas).
- Starburst, las reglas están parcialmente ordenadas en base a prioridades.
- En Ariel tienen prioridades numéricas entre -1000 y 1000 asignando por defecto el valor 0.
- HiPAC, todas las reglas disparadas se hacen en forma concurrente, pero incluye un mecanismo para influenciar su serialidad.

UNIVERSIDAD DE LOS ANDES

Organización de las reglas

- Algunos sistemas incluyen mecanismos para activar y desactivar selectivamente las reglas, para controlar el número de reglas que el sistema debe monitorear.
- En los sistemas relacionales no existen estos mecanismos.
- HiPAC, las reglas forman una jerarquía y pueden incluirse en colecciones, las cuales pueden activarse (enable) y desactivarse (disable).

UNIVERSIDAD DE LOS ANDES

Semántica de la ejecución de reglas

- A diferencia de los sistemas de IA, las BD activas procesan las reglas de forma integrada con las actividades normales de las BD, y son estas actividades las que disparan las reglas, ya que es ineficiente conocer en cada ciclo todas las reglas que se pueden disparar.
- Se considera la granularidad del procesamiento de reglas, las operaciones trabajan sobre un conjunto de tuplas o de objetos y se puede escoger el disparo luego de cada modificación de una tupla u objeto, o al final de una Ti.
- Si hay más de una regla disparada por un evento, algunos lenguajes tienen una ejecución secuencial de ellas y otros una concurrente, en ambos casos una regla puede disparar otra regla, y así sucesivamente, por lo cual la terminación de este anidamiento es un problema

UNIVERSIDAD DE LOS ANDES

ARIEL

- Las reglas se disparan por transiciones (procesamiento orientado al conjunto)
- La mínima granularidad es el conjunto de operaciones a nivel de tupla.
- La máxima granularidad es la Ti.
- El procesamiento de las reglas es secuencial e invocado automáticamente al final de las Ti y forma parte de la Ti que contiene la transición.
- Usa una máquina de inferencia cíclica similar a la usada en los sistemas de reglas en IA.
- Cada regla tiene asociada una prioridad numérica que no es única.

UNIVERSIDAD DE LOS ANDES

ARIEL

- El mecanismo de resolución de conflictos en caso de varias reglas disparadas es:
  1. Tomar la(s) regla(s) con la mayor prioridad
  2. Si hay enlace, tomar la(s) regla(s) que se correspondan con los cambios más recientes.
  3. Si aún hay enlace, tomar la(s) regla(s) cuyas condiciones sean más selectivas.
  4. Si aún hay enlace, tomar una regla arbitrariamente
- Se considera el *efecto neto* de las modificaciones en la transición.
  - ✓ Si una tupla es actualizada varias veces en la transición, el efecto neto es una sola modificación.
  - ✓ Si la tupla es actualizada y luego eliminada, el efecto neto es la eliminación.
  - ✓ Si la tupla es insertada y luego actualizada, el efecto neto es la inserción de la tupla actualizada y si es insertada y luego eliminada, el efecto neto es ningún cambio.

UNIVERSIDAD DE LOS ANDES

Postgres

- El procesamiento de las reglas se invoca inmediatamente que ha ocurrido la modificación (procesamiento orientado a la tupla), es recursivo y sincrónico.
- **El algoritmo de procesamiento es:**
  1. Algún usuario o regla modifica una tupla
  2. La modificación dispara las reglas R1, R2, ..., Rn
  3. Para cada regla Ri ejecutar la acción de Ri.
- No hay mecanismo de resolución de conflictos y
- Las reglas se ejecutan en un orden arbitrario.

UNIVERSIDAD DE LOS ANDES

Postgres

- Si hay un deshacer (rollback) en la acción de una regla, el procesamiento de termina y la Ti es abortada.

```
define rule colocarSalario
on insert to empleado
then begin
   salarioInicial :=  select avg(salario) from empleado -10;
   update empleado(salario = salarioInicial) where empleado.id = new.id
end
```

- Se dispara la regla por cada tupla insertada, por ello el salario de los nuevos varia.
- En Ariel, la regla se dispara una sola vez y el salario de los nuevos es el mismo

UNIVERSIDAD DE LOS ANDES

Starburst

- El procesamiento de las reglas es automático al final de cada Ti (ejecución diferida).
- La granularidad mínima es una operación de la BD y la máxima la Ti.
- Considera el efecto neto de un conjunto de modificaciones a partir del último disparo de la regla
  - ✓ Ejem. Insertar t1 y luego eliminarla.
- En cada ciclo, la máquina de inferencia determina un conjunto de reglas disparables (conflict set) sin eliminar aquellas cuya condición es falsa, es decir no evalúa la regla hasta haberla seleccionado para ejecución.

UNIVERSIDAD DE LOS ANDES

Starburst

- El usuario puede invocar una regla fuera del mecanismo normal de Ti.
  - ✓ **process rules** invoca el mismo algoritmo de procesamiento que para las Ti.
  - ✓ **process ruleset** S invoca también el mismo algoritmo anterior, pero para el subconjunto S.
  - ✓ **process rule** R similar a las anteriores, pero solamente para la regla R.
- Si hay que deshacer la Ti como parte de la acción de la regla disparada, entonces se aborta la Ti y se termina la ejecución.
- **Las reglas ECA:**
  - ✓ **Eventos** son las primitivas de modificación de datos en SQL,
  - ✓ **Condiciones** son predicados lógicos en SQL sobre el estado de la BD y
  - ✓ **Acciones** son consultas en SQL

UNIVERSIDAD DE LOS ANDES

Starburst

- Las reglas pueden ser agrupadas y selectivamente ser activadas o desactivadas por un usuario.

  <reglaStarburst > ::= **CREATE RULE** <nomReglas> **ON** <tabla>

  WHEN INSERTED|DELETED|UPDATED [<nomColumnas>**]**

  [IF <predicadoSQL> **]**

  THEN <instrucciónSQL>

  [PRECEDES <nomReglas> **] [FOLLOWS** <nomReglas> **]**

- La relación de precedencia entre las reglas debe ser acíclica.

UNIVERSIDAD DE LOS ANDES

Starburst

- **Algoritmo de procesamiento de reglas:**

  Mientras el conjunto conflicto no esté vacío

  1. Seleccione una regla R del conjunto con la más alta prioridad
  2. Evalúe su condición
  3. Si la condición de R es verdadera, se ejecuta su acción

- **Otros comandos:**

  DEACTIVATE RULE <nomRegla> **ON** <tabla>

  ACTIVATE RULE <nomRegla> **ON** <tabla>

  DROP RULE <nomRegla> **ON** <tabla>

  CREATE RULESET <nomConj>

  ALTER RULESET <nomConj>

  DROP RULESET <nomConj> **[ADDRULES** <nomReglas>**] [DELRULES** <nomReglas>**]**

UNIVERSIDAD DE LOS ANDES

SQL2 y SQL3

- Procesamiento de aserciones a nivel de tupla o de conjunto de tuplas.
- La escogencia se hace a tiempo de definición mediante la especificación de la cláusula **for each row** para usarlo a nivel de tupla, si se omite se usa a nivel de conjunto.
- Las aserciones se pueden definir asociadas a los comandos de la BD o con un evento especial denominado **before commit**.
- Se pueden referir los estados viejos (old) o nuevos (new)
- El procesamiento de reglas es estrictamente secuencial, asegurando así su terminación

UNIVERSIDAD DE LOS ANDES

HiPAC

- *Modos de acoplamiento:* Detemina como las reglas se relacionan con las Ti.
- Modo E-C: Como se ejecutan las condiciones de las reglas con respecto al evento.
- Modo C-A: Cuando se ejecutará la acción con respecto a la condición de la regla.
  - ✓ Inmediato: De ejecución inmediata.
  - ✓ Diferido: Se ejecuta al final de la Ti.
  - ✓ Desacoplado: Se ejecutará en una Ti separada.

UNIVERSIDAD DE LOS ANDES

HiPAC

Modo E-C	Modo C-A		
	Inmediato	Diferido	Desacoplado
Inmediato	Chequeo de la condición y ejecución de la acción luego del evento	Chequeo de la condición luego del evento y ejecución de la acción al final de la Ti	Chequeo de la condición luego del evento y ejecución de la acción en una Ti separada
Diferido	No permitido	Chequeo de la condición y ejecución de la acción al final de la Ti	Chequeo de la condición al final de la Ti y ejecución de la acción en Ti separada
Desacoplado	Chequeo de la condición y ejecución de la acción en Ti separada	No permitido	Chequeo de la condición en Ti separada y ejecución de la acción en otra Ti separada

UNIVERSIDAD DE LOS ANDES

HiPAC

- Ejecuta todas las reglas disparadas en forma concurrente.
- Usa una extensión del modelo de Ti anidadas como modelo de ejecución del procesamiento de reglas e incluye los modos de acoplamiento.

Algoritmo de procesamiento:

1.- Algún evento de usuario o interno dispara las reglas R1, R2, ..., Rn

2.- Para cada regla Ri planificar una Ti para

a. evaluar la condición de Ri

b. si la condición es verdadera, planifique una Ti para su acción.

- La ejecución de las Ti en la parte 2a se hace basada en los modos E-C y las de la parte 2b en los modos C-A.

UNIVERSIDAD DE LOS ANDES

HiPAC

- Un modo inmediato hace que se produzca de manera inmediata una subtransacción anidada.
- Un modo diferido hace que se produzca una subtransacción anidada al final de la Ti actual, donde ella pasa (commit).
- Un modo desacoplado hace que se produzca una Ti separada, sin anidamiento.
- Las reglas pueden tener un orden relativo con el cual se puede influenciar la serialización de las subtransacción anidadas

UNIVERSIDAD DE LOS ANDES

Recuperación de errores

- Este aspecto no ha sido estudiado lo suficiente en HiPAC.
- Una regla puede generar un error en su ejecución, por diferentes razones: que los datos leídos hayan sido eliminados, sus permisos de acceso hayan sido revocados, porque hay un inter-bloqueo, porque hay un error del sistema, etc.
- La mayoría maneja estos errores por medio de la anulación de la Ti.
- Cualquier evento interno se puede recuperar con los mecanismos normales, pues son operaciones de la BD.
- En caso de eventos externos o temporales, ellos se deben declarar como recuperables, para su anotación en el diario.

UNIVERSIDAD DE LOS ANDES

Implementación

- Una BD activa debe proveer los mecanismos para:
  - ✓ detección de eventos (HiPAC, detector de eventos)
  - ✓ disparo de reglas (HiPAC, manejador de reglas, trazado de ejecución, etc)
  - ✓ prueba de sus condiciones
  - ✓ ejecución de sus acciones
  - ✓ desarrollo de reglas en las aplicaciones
- El procesamiento de las reglas puede llevar a lazos infinitos.
- Mecanismos para detectarlos

UNIVERSIDAD DE LOS ANDES

Oracle

- **Granularidad**: a nivel de tupla (orientada a la instancia) o de instrucción (al conjunto).
- El disparador se considera y se efectúa **antes** y **después** de la operación disparable.
- **Combinaciones**:
  - ✓ tupla-disparador-antes
  - ✓ instrucción-disparador-antes
  - ✓ tupla-disparador-después
  - ✓ instrucción-disparador-después
- **Sintaxis**:

CREATE TRIGGER <nomDisparador>
{BEFORE|AFTER} INSERT|DELETE|UPDATE [OF <nomColumnas> **]**
ON <tabla>
[[REFERENCING OLD AS <tuplaVieja> **| NEW AS** <tuplaNueva> **]**
FOR EACH ROW [WHEN (<condición>**)]]** <bloqueSQL/LPAN>

UNIVERSIDAD DE LOS ANDES

Oracle

Algoritmo de procesamiento de las reglas:

1. Ejecute los disparadores instrucción-disparador-antes
2. Para cada tupla de la tabla
   a. Ejecute los disparadores tupla-disparador-antes
   b. Efectúe las modificaciones de la tupla y las tuplas asociadas por la integridad referencial verificando las aserciones
   c. Ejecute los disparadores tupla-disparador-después
3. Efectúe las modificaciones asociadas a la instrucción por integridad referencial verificando las aserciones.
4. Ejecute los disparadores instrucción-disparador-después

UNIVERSIDAD DE LOS ANDES

Oracle

- El número máximo de disparadores en cascada es 32.
- Cuando el sistema alcanza este número, que indica la posibilidad de no finalización, se suspende la ejecución y se da una condición de excepción.
- Si una excepción es alcanzada u ocurre un error, entonces todos los cambios efectuados en la BD, por la instrucción SQL original y todas las acciones disparadas subsecuentes son deshechas (rollback).
- Por esto, Oracle soporta operaciones de deshacer a nivel de instrucciones en vez de a nivel transaccional.

UNIVERSIDAD DE LOS ANDES

DB2

- Manejo de las reglas parecido a Oracle.

Sintaxis:

CREATE TRIGGER <nomDisparador>

{BEFORE|AFTER} INSERT|DELETE|UPDATE [OF <nomColumnas> **]**

ON <tabla>

[REFERENCING OLD AS <tuplaVieja> **| NEW AS** <tuplaNueva> |

OLD_TABLE AS <tablaVieja> **| NEW_TABLE AS** <tablaNueva> **]**

FOR EACH { ROW | STATEMENT }

WHEN (<condiciónSQL>**)** <bloqueSQL/LPAN>

- Los disparadores no pueden desarrollar comandos transaccionales, pero pueden tener errores que causan que las operaciones a nivel de instrucción sean deshechas.

UNIVERSIDAD DE LOS ANDES

DB2

- Los disparadores BEFORE se usan para detectar errores o condiciones a la entrada, generalmente sobre valores nuevos (NEW) y ellos no pueden usar UPDATE, DELETE o INSERT.
- Los disparadores AFTER se evalúan y su acción posiblemente se efectúa después del evento modificación.
- El estado anterior de una tabla T es

  (T - NEW_TABLE) UNION OLD_TABLE
- Varios disparadores pueden monitorear el mismo evento, en base al orden total impuesto por DB2, que toma en cuenta el tiempo de creación del disparador.

UNIVERSIDAD DE LOS ANDES

DB2

- Si una instrucción de modificación S en la acción A causa el evento E, entonces:

Algoritmo de procesamiento de las reglas:

1. Suspender A y guardar su estado en el área de trabajo en la pila
2. Calcular los valores de transición (OLD, NEW) relativos al evento E
3. Considerar y ejecutar todos los disparadores-antes relativos a E, posiblemente cambiando los valores de transición NEW.
4. Aplicar los valores de transición NEW a la BD haciendo el cambio de estado de E.
5. Considerar y ejecutar todos los disparadores-después relativos a E. Si cualquiera de sus acciones Ai activa otros disparadores, iniciar este proceso recursivo para Ai.
6. Sacar de la pila el área de trabajo de A y continuar su evaluación

UNIVERSIDAD DE LOS ANDES

DB2

- El paso 4 del procedimiento anterior se modifica si se viola alguna restricción en SQL-92.

4. Aplicar los valores de transición NEW a la BD haciendo el cambio de estado de E. Para cada restricción de integridad violada, considerar la acción Aj que la compensa

a. Calcular los valores de transición (OLD, NEW) relativos a Aj

b. Ejecute los disparadores-antes relativos a Aj cambiando posiblemente los valores de transición NEW

c. Aplicar los valores de transición NEW a la BD haciendo el cambio de estado asociado a Aj

d. Meter en la cola de los disparadores suspendidos todos los disparadores-después relativos a Aj

- Este manejo de los disparadores es el aceptado en SQL3.

UNIVERSIDAD DE LOS ANDES

DB2

CREATE TRIGGER AuditarProveedor

AFTER UPDATE

ON Parte

REFERENCING OLD_TABLE AS TV

FOR EACH STATEMENT

INSERT INTO Auditar **(**usuario, fechaActual**, SELECT COUNT(*) FROM** TV **)**

UNIVERSIDAD DE LOS ANDES

Chimera

- Es un lenguaje OO con disparadores orientados a conjuntos que usa el Oid para identificar los objetos afectados por los eventos, con varios modos de procesarlos.

Instrucciones:

select (X **where** Empleado(X), X.nombre = 'Juan')

create(Empleado, ['Juan', 500.000,00], X)

delete(Empleado, X **)**

modify(Empleado.salario, X, X.salario*1.01 **)**

define [event-consuming | event-preserving] [deferred | immediate]
trigger <nomDis> **[for** <nomClase> **]**

UNIVERSIDAD DE LOS ANDES

Chimera

events create | delete | modify | display | generalize | specialize
[(<nomAtributo>**)]**

condition <fórmula>

actions <instrucciones>

[{before | after} <nomDisparadores> **]**

end

Algoritmo de procesamiento de los gatillos:

Mientras el conjunto conflicto C no esté vacío

1. Seleccione un disparador D con prioridad alta de C
2. Evalúe la condición de D
3. Si la condición de D es verdadera, entonces ejecute la acción de D