Bases de Datos Activas

Miguel Rodríguez Penabad

Laboratorio de Bases de Datos

Universidade da Coruña

Introducción

BD Pasivas vs. Activas

Bases de datos pasivas

- Son meros almacéns de datos
- Toda a xestión é explícita
- Hai controles mínimos (clave primaria, foránea)

Bases de datos activas

- "Reaccionan" ante eventos (normalmente cambios nos datos)
- Utilidade:
  - Control de restriccións

    Reglas de negocio: Un empleado non pode ganar máis que o seu xefe
    Validacións: NIF correcto, sexo é H ou M, ...
  - Actualización de outros datos

    Tabla facturas con campo total autoactualizado
  - Accións externas á base de datos
- A actividade impleméntase normalmente con triggers (disparadores)

Introducción

Actividade sen triggers

Restricciónes nivel de táboa

- Integridade de entidade/clave
- Integridade referencial
- Restricción `unique`
- Restricción `check`

Limitacións

- A acción sempre é abortar
- CHECK sen subconsultas en SXBD comerciais (SQL:2003 permíteas)
- Non reacciona ante borrado de filas (excepto FK)

```
CREATE TABLE EMP(
   EMPNO NUMERIC(3) NOT NULL,
   ENAME CHAR(20),
   EMAILADDRESS CHAR(25) UNIQUE,
   MGR NUMERIC(3), DEPTNO NUMERIC(3),
   SAL NUMERIC(7,2),
   CONSTRAINT PK_EMP PRIMARY KEY(EMPNO),
   CONSTRAINT FK_EMP FOREIGN KEY(MGR)
         REFERENCES EMP(EMPNO) ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT C_SAL_POS CHECK(SAL > 0) );
```


Introducción

Actividade sen triggers

Restricciónes a nivel de BD: Assertions

- `CREATE ASSERTION <nome> CHECK <condicion>`

```
CREATE ASSERTION as_salarios_similares
  CHECK( ( (SELECT MAX(SAL) FROM EMP)-
           (SELECT MIN(SAL) FROM EMP) ) <100);

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

Limitacións

- Poucos (ou ningún) SXBD as implementan
- A reacción sempre é abortar

Triggers
As reglas ECA

Trigger = Disparador = Regla ECA

Evento: Cando se produce, o trigger dispárase.
- (DML) Modificación de datos: INSERT, DELETE, UPDATE
- Outros eventos: conexión á BD, etc.

Condición: (Opcional) Debe verificarse para que se execute a acción.

Acción: Que se executa como resposta ó evento cando se da a condición

Exemplos

- Control de Nif (?)
  - E: Inserción ou Actualización
  - C: NIF incorrecto
  - A: Non permitir a inserción/actualización
- Actualizar total en táboa factura (?)
  - E: Inserción, Borrado, Modificación (táboa Liña)
  - C:
  - A: Actualizar total (táboa Factura)
- Segundo SQL:1999 e SQL:2003, un trigger controla un único evento $\Rightarrow$ 2 triggers (NIF) e 3 triggers(Facturas)

Triggers

Ventaxas dos triggers

- Simplifica a codificación de aplicacións de acceso a BDs
- Maior consistencia (control centralizado)

Triggers no estándar SQL

- Dentro do estándar ISO/IEC 9075 Part 2: (SQL/Foundation)
- Especificados no SQL:1999 (antes SQL-3)
- Manténense no SQL:2003
- Un trigger [DML] "controla" unha única táboa
- Un trigger responde a un único evento

Triggers en SQL:2003

Elementos dun trigger

- O nome do trigger
- O nome da táboa
- O evento
  - Inserción (`INSERT`)
  - Borrado (`DELETE`)
  - Modificación(`UPDATE [OF _lista_columnas_]`)
- O tempo de activación
  - Antes (`BEFORE`)
  - Despois (`AFTER`)
- Granularidade
  - A nivel sentencia (`STATEMENT`)
  - A nivel fila (`ROW`)

Triggers en SQL:2003

Elementos dun trigger (ii)

- Táboas e variables de transición
  - Valores antigos (`OLD`) e novos (`NEW`)
  - Táboa de transición: conxunto de filas afectadas

    `OLD TABLE, NEW TABLE`

EMP antes				EMP despois		
EMPNO	MGR	SAL	$\Longrightarrow$	EMPNO	MGR	SAL
0001	1000	1500		0001	1000	1500
0002	1000	1600	`UPDATE EMP`	0002	1000	3100
0010	1300	1550	`SET SAL=3100`	0010	1300	3100
1000	1100	3000	`WHERE SAL>1540`	1000	1100	3100

  - Variable de transición: recorre as filas da táboa de transición

    `OLD ROW, NEW ROW`
- A acción
  - Inclúe a condición (pode omitirse)
  - Debe ser "atómica"
  - O seguinte non se recomenda (aberto a implementación):
    - Sentencias de modificación de datos en triggers `BEFORE`
      (Podemos usar `SET <nome new row> = <valor>`)
    - Sentencias de control de transaccións, conexión a BD, ou DDL

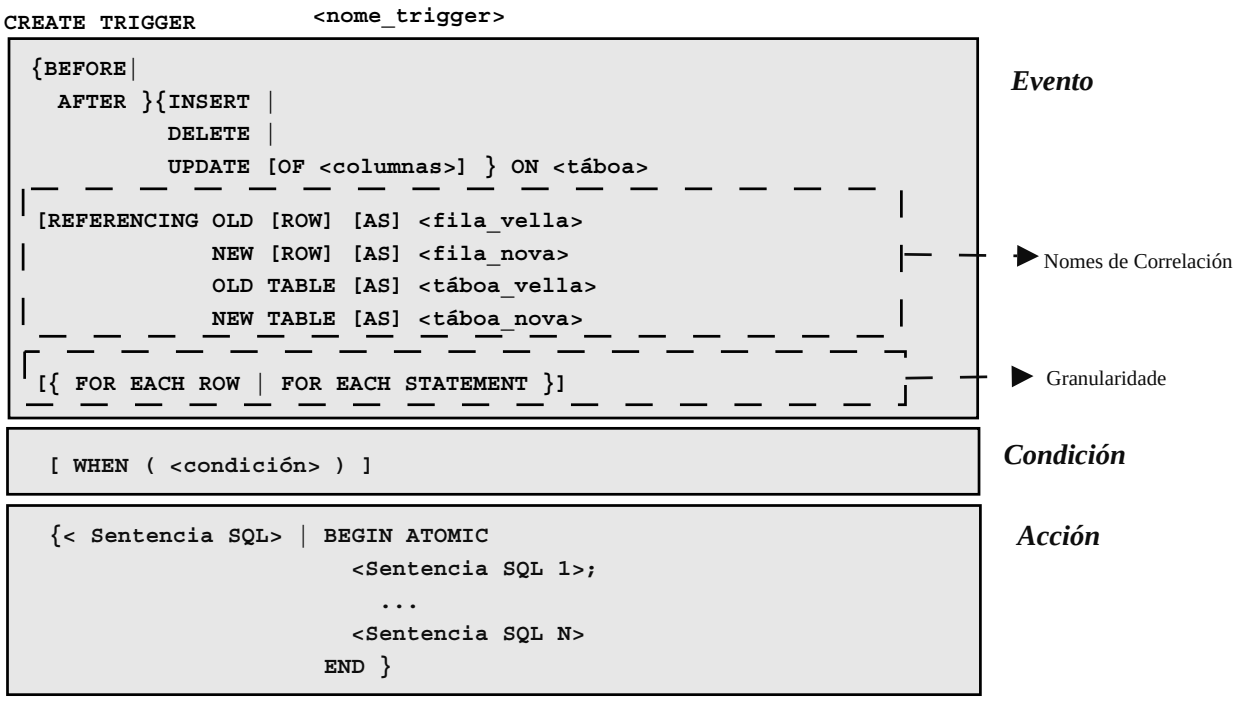
Exemplo 1: DNI correcto

- O nome do trigger: `t_dni_correcto_ins`
- O nome da táboa: `persoa`
- Evento: `inserción` (outro trigger para actualización)
- O tempo de activación: `AFTER`
- Granularidade: `FILA`
- Táboas e variables de transición
  - `OLD/NEW TABLE`: Non as necesitamos
  - `OLD ROW`: Non (non existe)
  - `NEW ROW`: Si
- Acción
  - Condición: NIF incorrecto
  - Acción: Borrar a persoa insertada

Exemplo 2: Un empregado non debe cobrar máis que o seu xefe (actualización)

- O nome do trigger: `t_mal_salario_upd`
- O nome da táboa: `empregado`
- Evento: `actualización` (outro trigger para inserción)
- O tempo de activación: `BEFORE`
- Granularidade: `FILA`
- Táboas e variables de transición
  - `OLD/NEW TABLE`: Non as necesitamos
  - `OLD ROW`: Si
  - `NEW ROW`: Si
- Acción
  - Condición: O empregado actualizado ten xefe, e este cobra menos, ou o empregado actualizado é un xefe, e ten subordinados que cobran máis
  - Acción: Deixar o salario que tiña o empregado
- Pregunta: Implicacións de facer este trigger de tipo `after`

Triggers en SQL:2003



Para borrar un trigger:

```
DROP TRIGGER <nombre_trigger>;
```


Triggers en SQL:2003

Exemplos

Exemplo 1: O salario dos empregados debe ser positivo

Táboa:
EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO))

- Usando restricción

```
ALTER TABLE Emp
    ADD CONSTRAINT sal_pos
            CHECK (sal>0)
```

- Usando assertions

```
CREATE ASSERTION emp_sal_pos
     CHECK (NOT EXISTS (SELECT *
                        FROM Emp
                        WHERE sal<=0))
```

Comentarios

- Funcionan tanto para INSERT como para UPDATE
- Comproban os datos existentes na BD
- Poden ser diferidas ata o fin da transacción
- Provocan erro + rollback

Triggers en SQL:2003

Exemplos

Exemplo 1 (cont.): O salario dos empregados debe ser positivo

- Usando triggers
  - Para a inserción, evitando a inserción

```
CREATE TRIGGER sal_pos_ins
   AFTER INSERT ON Emp
   REFERENCING NEW ROW AS nova
   FOR EACH ROW
   WHEN (nova.sal <=0)
   DELETE FROM Emp
      WHERE empno = nova.empno;
```

  - Para a actualización, deixando o salario anterior

¿Facer un novo update?

```
CREATE TRIGGER sal_pos_upd1
   AFTER UPDATE OF sal ON Emp
   REFERENCING NEW ROW AS nova
               OLD ROW AS vella
   FOR EACH ROW
   WHEN (nova.sal <=0)
   UPDATE Emp SET sal = vella.sal
       WHERE empno = nova.empno;
```

Mellor cambiar o valor na fila actual

```
CREATE TRIGGER sal_pos_upd2
   BEFORE UPDATE OF sal ON Emp
   REFERENCING NEW ROW AS nova
               OLD ROW AS vella
   FOR EACH ROW
   WHEN (nova.sal <=0)
   SET nova.sal = vella.sal;
```


Triggers en SQL:2003

Consideracións sobre os triggers

- A creación do trigger non executa o trigger
  - O trigger só se executa cando ocurre o evento e se cumpre a condición
  - Non é posible executar "directamente" un trigger
- O trigger non valida datos existentes na BD (as restriccións si)
  - Os triggers do exemplo anterior permiten salarios negativos se xa estaban na táboa
- A execución do trigger non evita que se execute o evento disparador
  - Triger `before <evento>`: execútase o trigger e logo o evento
  - Triger `after <evento>`: execútase o evento e logo o trigger

Exercicios

1. Un empregado non pode ganar máis ca o seu xefe
2. Un departamento non pode ter máis de 10 empregados

3. Manda un correo ó suministrador cando o stock dun artigo baixa do límite (`ARTIGO(CODART,PREZO,STOCK, MIN, SUMINISTRADOR)`, procedemento `PideArtigo(suministrador, codigo)`)

4. Actualiza `FACTURA(NUMERO, TOTAL)` cando se engaden, borran ou modifican liñas (`LIÑA(NUMFAC, NUMLI,IMPORTE)`)