Seguridad e Integridad

M.C. Carlos Rojas Sánchez[1]

[1]Licenciatura en Informática
Universidad del Mar - Puerto Escodido

[Bases de Datos II]

Restricciones de integridad

Proporcionan un medio de asegurar que las modificaciones hechas a la base de datos por los usuarios autorizados no provoquen la pérdida de la consistencia de los datos.

Integridad de dominios

- La declaración de que un atributo pertenezca a un determinado dominio actúa como una restricción sobre los valores que puede tomar. Las restricciones de los dominios son la forma más simple de restricción de integridad. El sistema las verifica fácilmente siempre que se introduce en la base de datos un nuevo elemento de datos.
- La cláusula **create domain** se puede usar para definir nuevos dominios.
- La cláusula **check** permite restringir los dominios.

Integridad de dominios

create domain sueldo-por-hora numeric(5,2) constraint comprobación-valor-sueldo check(value $\geq$ 4.00)

Integridad de dominios

Las condiciones check complejas pueden ser útiles cuando se desee asegurar la integridad de los datos, pero pueden ser costosas de comprobar.

Integridad referencial

```
create table cuenta
(...
foreign key (nombre-sucursal) references sucursal
(nombre-sucursal)
on delete cascade
on update cascade,
...)
```

Integridad referencial

- Debido a la cláusula on delete cascade asociada con la declaración de la clave externa, si un borrado de una tupla de sucursal da lugar a que se viole la restricción de integridad referencial, el sistema no rechaza el borrado.
- En su lugar, el borrado se realiza en “cascada” en la relación cuenta, borrando la tupla que hace referencia a la sucursal que se borró.

2 Asertos

Asertos

- Un aserto es un predicado que expresa una condición que se desea que la base de datos satisfaga siempre.
- Las restricciones de dominio y las de integridad referencial son formas especiales de los asertos.

Asertos

- NOT NULL
- CHECK CONSTRAINT
- CHECK CONSTRAINT ... IN
- CHECK CONSTRAINT ... LIKE
- (AND u OR)... CHECK CONSTRAINT
- UNIQUE CONSTRAINT

Asertos

Cuando se crea un aserto el sistema comprueba su validez. Si el aserto es válido, sólo se permiten las modificaciones posteriores de la base de datos que no hagan que se viole el aserto. Esta comprobación puede introducir una sobrecarga importante si se han realizado asertos complejos.

3 Uso de disparadores

Disparadores

Un disparador es una orden que el sistema ejecuta de manera automática como efecto secundario de la modificación de la base de datos.

Disparadores

Requisitos

- Especificar las condiciones en las que se va a ejecutar el disparador. Esto se descompone en un evento que causa la comprobación del disparador y una condición que se debe cumplir para ejecutar el disparador.
- Especificar las acciones que se van a realizar cuando se ejecute el disparador.

Disparadores

- Este modelo de disparadores se denomina modelo evento-condición-acción

Disparadores

- Los disparadores son mecanismos útiles para alertar a los usuarios o para realizar de manera automática ciertas tareas cuando se cumplen determinadas condiciones.
- Un disparador es un objeto con nombre dentro de una base de datos el cual se asocia con una tabla y se activa cuando ocu- rre en ésta un evento en particular.

Disparadores

Ejemplo en MySQL

- Por ejemplo, las siguientes sentencias crean una tabla y un disparador para sentencias INSERT dentro de la tabla. El disparador suma los valores insertados en una de las columnas de la tabla:
  - mysql> CREATE TABLE account (acct_num INT, amount DECIMAL(10,2));
  - mysql> CREATE TRIGGER ins_sum BEFORE INSERT ON account
  - -> FOR EACH ROW SET @sum = @sum + NEW.amount;

Disparadores

Sintaxis de CREATE TRIGGER

CREATE TRIGGER nombre_disp momento_disp evento_disp
ON nombre_tabla FOR EACH ROW sentencia_disp

momento_disp es el momento en que el disparador entra en acción. Puede ser BEFORE (antes) o AFTER (despues), para indicar que el disparador se ejecute antes o después que la sentencia que lo activa.

Disparadores

Sintaxis de CREATE TRIGGER

evento_disp indica la clase de sentencia que activa al disparador. Puede ser INSERT, UPDATE, o DELETE.

- NOTA: No puede haber dos disparadores en una misma tabla que correspondan al mismo momento y sentencia.

Disparadores

Sintaxis de CREATE TRIGGER

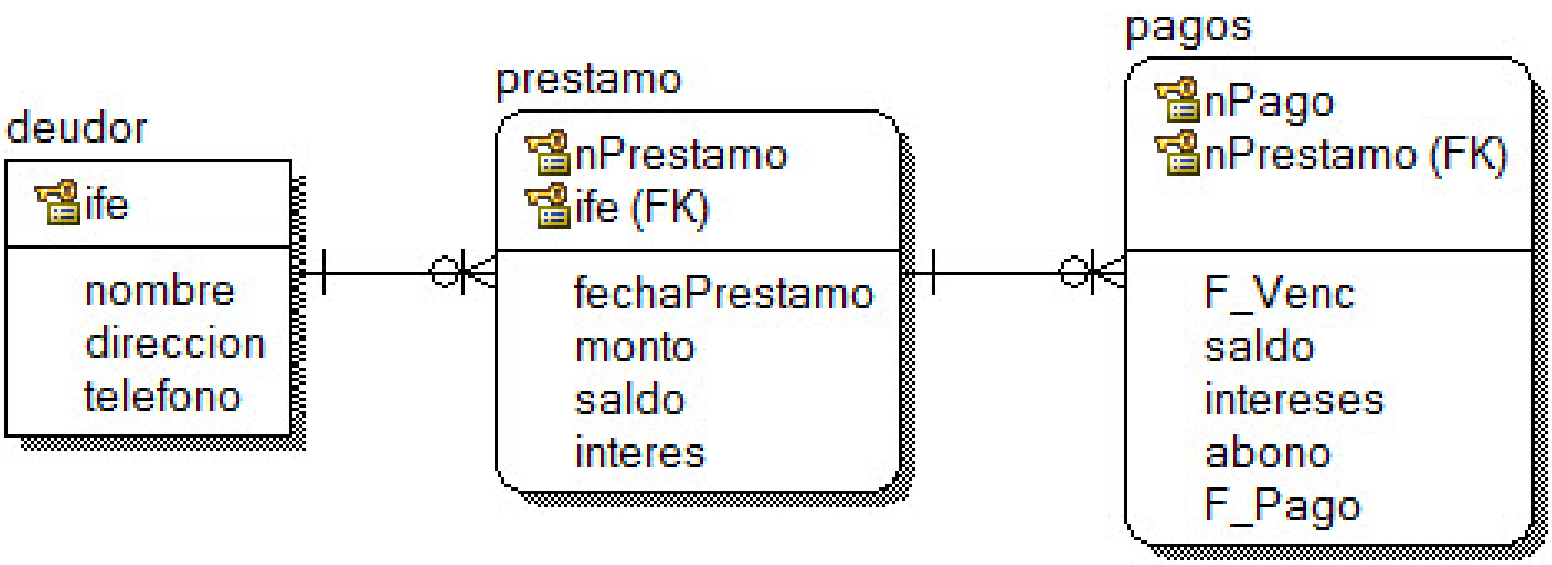
sentencia_disp es la sentencia que se ejecuta cuando se activa el disparador. Si se desean ejecutar múltiples sentencias, deben colocarse entre BEGIN ... END, el constructor de sentencias compuestas.

```
 1  CREATE TABLE test1(a1 INT);
 2  CREATE TABLE test2(a2 INT);
 3  CREATE TABLE test3(a3 INT NOT NULL AUTO_INCREMENT PRIMARY KEY);
 4  CREATE TABLE test4( a4 INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
        b4 INT DEFAULT 0 );
 5  DELIMITER |
 6  CREATE TRIGGER testref BEFORE INSERT ON test1 FOR EACH ROW
 7  BEGIN
 8  INSERT INTO test2 SET a2 = NEW.a1;
 9  DELETE FROM test3 WHERE a3 = NEW.a1;
10  UPDATE test4 SET b4 = b4 + 1 WHERE a4 = NEW.a1;
11  END |
12  DELIMITER ;
13  INSERT INTO test3 (a3) VALUES (NULL),(NULL),(NULL),(NULL),(NULL)
        ,(NULL),(NULL),(NULL),(NULL),(NULL);
14  INSERT INTO test4 (a4) VALUES (0),(0),(0),(0),(0),(0),(0),(0)
        ,(0),(0);
15  INSERT INTO test1 VALUES (1),(3),(1),(7),(1),(8),(4),(4);
```

Opciones en Triggers

- show triggers;
- drop TRIGGER nombre_disp
    - DROP TRIGGER ins_sum;

Ejemplo Triggers



4 Vistas

Vistas

- Una vista es una tabla virtual cuyo contenido está definido por una consulta.
- Las vistas suelen utilizarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario.
- Las vistas pueden emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista, pero no les conceden el permiso de obtener acceso directo a las tablas subyacentes de la vista.

Vistas

- Las tablas y las vistas comparten el mismo espacio de nombres en la base de datos, por lo tanto, una base de datos no puede contener una tabla y una vista con el mismo nombre.

Vistas

Sintaxis

CREATE [OR REPLACE] [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}] VIEW nombre_vista [(columnas)] AS sentencia_select [WITH [CASCADED | LOCAL] CHECK OPTION]

- CREATE VIEW nombre_vista AS consulta;

Ejemplo:

```
1  CREATE TABLE entidad (
2  estado      CHAR(30) NOT NULL,
3  poblacion   INTEGER  NOT NULL
4  ) ENGINE = InnoDB;
5  INSERT INTO entidad values ('Chiapas',                4796580);
6  INSERT INTO entidad values ('Nuevo␣León',             4653458);
7  INSERT INTO entidad values ('Michoacán␣de␣Ocampo',4351037);
8  INSERT INTO entidad values ('Oaxaca',                 3801962);
9  INSERT INTO entidad values ('Chihuahua',              3406465);
10 INSERT INTO entidad values ('Guerrero',               3388768);
11 CREATE OR REPLACE VIEW entidades AS
12 SELECT * FROM entidad
13 WHERE poblacion < 2000000;
14 INSERT INTO entidad values ('Estado␣de␣México',    175862);
15 INSERT INTO entidad values ('Distrito␣Federal',    151080);
```

Opciones de Vistas

- DROP VIEW nombrevista;
- SHOW CREATE VIEW nombrevista\G

5 Consideraciones generales de seguridad

Usar conexiones seguras

- La configuración estándard de MySQL tiene la misión de ser tan rápida como sea posible, así que no se usan las conexiones cifradas por defecto.
- Hacerlo, haría que el protocolo cliente/servidor fuese mucho más lento.

Requisitos (OpenSSL)

- Para utilizar conexiones SSL entre el servidor MySQL y los programas cliente
  - El sistema debe tener la capacidad de ejecutar OpenSSL
  - La versión de MySQL debe ser la 4.0.0 o superior

6 Definición de un esquema de seguridad

Requisitos (OpenSSL)

- Instalar la librería OpenSSL. MySQL ha sido comprobado con OpenSSL 0.9.6.
- Para comprobar si un servidor mysqld que se está ejecutando tiene soporte para OpenSSL, examine el valor de la variable de sistema have_openssl:
  - mysql> SHOW VARIABLES LIKE 'have_openssl';

Ejemplo

- Acceder al directorio: ...xampp\apache\bin
- Generar llaves, publica y privada,
  - Certificado
    - openssl genrsa 2048 > "C:/xampp/mysql/certs/ca-key.pem"
    - set OPENSSL_CONF=c:\xampp\apache\conf\openssl.cnf
    - openssl req -new -x509 -nodes -days 3600 -key "C:/xampp/mysql/certs/ca-key.pem" > "C:/xampp/mysql/certs/ca-cert.pem"

Ejemplo

- Clave privada
  - openssl req -newkey rsa:2048 -days 3600 -nodes -keyout "C:/xampp/mysql/certs/server-key.pem" > "C:/xampp/mysql/certs/server-req.pem"
    - openssl x509 -req -in "C:/xampp/mysql/certs/server-req.pem" -days 3600 -CA "C:/xampp/mysql/certs/ca-cert.pem" -CAkey "C:/xampp/mysql/certs/ca-key.pem" -set_serial 01 > "C:/xampp/mysql/certs/server-cert.pem"

Ejemplo

- Clave pública
  - openssl req -newkey rsa:2048 -days 3600 -nodes -keyout "C:/xampp/mysql/certs/client-key.pem" > "C:/xampp/mysql/certs/client-req.pem"
  - openssl x509 -req -in "C:/xampp/mysql/certs/client-req.pem" -days 3600 -CA "C:/xampp/mysql/certs/ca-cert.pem" -CAkey "C:/xampp/mysql/certs/ca-key.pem" -set_serial 01 > "C:/xampp/mysql/certs/client-cert.pem"

Ejemplo

- Usuario
  - GRANT ALL PRIVILEGES ON *.* TO ’ssluser’@’localhost’ IDENTIFIED BY ’ssluser’ REQUIRE SSL;
- Configurar, my.cnf
  - mysqld
    - ssl-ca = "C:/xampp/mysql/certs/ca-cert.pem"
    - ssl-cert = "C:/xampp/mysql/certs/server-cert.pem"
    - ssl-key = "C:/xampp/mysql/certs/server-key.pem"
  - mysql -ussluser -pssluser -P3306 –ssl-key="C:/xampp/mysql/certs/client-cert.pem"

Bibliografía I

- Jorge Sánchez.
  Principios sobre Bases de Datos Relacionales.
  Creative-Commons, 2004.
- Abraham Silberschatz, Henry F. Korth and S. Sudarshan.
  Fundamentos de Bases de Datos.
  McGRAW-HILL, 2002.