111mil Programadores</>

Base de Datos

Apunte 2

Prof. Germán C. Basisty

basisty.german@gmail.com

LANÚS MUNICIPIO

FUNDACIÓN DIIEJ
Desarrollo para la Inclusión e Igualdad en el Empleo Juvenil

Índice de Contenidos

Restricciones

Una restricción es una condición que obliga el cumplimiento de ciertas condiciones en la base de datos. Algunas restricciones son determinadas por los usuarios (*restricciones semánticas*), mientras que otras son inherentemente definidas por el simple hecho de que la base de datos sea relacional.

Restricciones Relacionales

No existen tuplas repetidas (obligatoriedad de clave primaria). La relación se ha definido como un conjunto de tuplas, y en matemáticas los conjuntos por definición no incluyen elementos repetidos. Hay que decir, sin embargo, que muchos de los RDBMS relacionales sí admiten duplicidad de tuplas.

El orden de las tuplas y el de los atributos no es relevante. Cada atributo de cada tupla solo puede tomar un único valor sobre el dominio sobre el que está definido.

Ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo (regla de integridad de entidad).

Las 12 reglas de Codd

1. **Información**. Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.

2. **Acceso garantizado**. Todo dato es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato.

3. **Tratamiento sistemático de los valores nulos**. El RDBMS debe permitir el tratamiento adecuado de valores nulos.

4. **Catálogo en línea basado en el modelo relacional**. Los metadatos deben de ser accesibles usando un esquema relacional.

5. **Sublenguaje de datos completo**. Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación.

6. **Actualización de vistas**. El RDBMS debe encargarse de que las vistas (ya veremos que son las vistas) muestren la última información.

7. **Inserciones, modificaciones y eliminaciones de dato nivel**. Cualquier operación de modificación debe actuar sobre conjuntos de filas, nunca deben actuar registro a registro.

8. **Independencia física**. Los datos deben de ser accesibles desde la lógica de la base de datos aún cuando se modifique el almacenamiento.

9. **Independencia lógica**. Los programas no deben verse afectados por cambios en las tablas.

10. **Independencia de integridad**. Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.

11. **Independencia de la distribución**. El sublenguaje de datos debe permitir que sus instrucciones funcionen igualmente en una base de datos distribuida que en una que no lo es.

12. **No subversión**. Si el DBMS posee un lenguaje que permite el recorrido registro a registro, éste no puede utilizarse para incumplir las reglas relacionales.

Restricciones Semánticas (o de usuario)

Restricción de Clave Primaria (PRIMARY KEY)

Permite declarar un atributo o conjunto de atributos como la clave primaria de una relación.

Ejemplo

```
CREATE TABLE person (
        dni             integer PRIMARY KEY,
        name            text
);

INSERT INTO person VALUES (25888888, 'Roberto Garcia');
OK

INSERT INTO person VALUES (25888888, 'Andres Morales');
ERROR!
```

Restricción de Unicidad (UNIQUE)

Permite que una clave alternativa o secundaria pueda tomar valores únicos para las tuplas de una relación (como si de una clave primaria se tratara). Se entiende que la clave primaria siempre tiene esta restricción.

Ejemplo

```
CREATE TABLE person (
        dni             integer PRIMARY KEY,
        passport        integer UNIQUE,
        name            text
);

INSERT INTO person VALUES (25888888, 4445555, 'Roberto Garcia');
OK

INSERT INTO person VALUES (25883455, 4445555, 'Andres Morales');
ERROR!
```

Restricción de Obligatoriedad (NOT NULL)

Permite declarar si uno o varios atributos de una relación debe tomar siempre un valor.

Ejemplo

```
CREATE TABLE person (
        dni             integer PRIMARY KEY,
        name            text NOT NULL,
        phone_number    text
);

INSERT INTO person VALUES (25888888, 'Roberto Garcia', '1544445555');
OK

INSERT INTO person VALUES (25888887, 'Alberto Garcia', NULL);
OK

INSERT INTO person VALUES (25888886, NULL, '1566667777');
ERROR
```

Restricción de Integridad Referencial (FOREIGN KEY)

Se utiliza para que mediante claves foráneas podamos enlazar relaciones de una base de datos. La integridad referencial nos indica que si una relación tiene una clave foránea que referencia a otra relación, cada valor de la clave foránea o ajena tiene que ser igual a un valor de la clave principal de la relación a la que referencia, o bien, ser completamente nulo. Los atributos que son clave foránea en una relación no necesitan tener los mismos nombres que los atributos de la clave primaria con la cual ellos se corresponden. El diseñador de la base de datos deberá poder especificar qué operaciones han de rechazarse y cuáles han de aceptarse, y en este caso, qué operaciones de compensación hay que realizar para mantener la integridad de la base de datos. Para ello el diseñador debe plantearse tres preguntas por cada clave foránea:

- ¿Puede aceptar nulos esa clave foránea? Por ejemplo, (tomando como referencia las relaciones PROVEEDORES, ARTICULOS) ¿tiene sentido la existencia de un artículo cuyo proveedor se desconoce? Evidentemente, no. En algunos casos esta respuesta podría ser distinta, por ejemplo, en una base de datos con las relaciones EMPLEADOS y DEPARTAMENTOS, podría existir un empleado no asignado de momento a un departamento.

- ¿Qué deberá suceder si se intenta eliminar una tupla referenciada por una clave foránea? Por ejemplo, si se intenta eliminar un proveedor del cual existe algún artículo. En general, para estos casos existen por lo menos tres posibilidades:

- Restringir: La operación de eliminación se restringe sólo al caso en el que no existe alguna tupla con clave foránea que la referencie, rechazándose en caso contrario. En nuestro ejemplo, un proveedor podrá ser borrado, si y sólo si, por ahora, no suministra artículos.
- Propagar en cascada: La operación de borrado se propaga en cascada eliminando también todas las tuplas cuya clave foránea la referencien. En nuestro ejemplo, se eliminaría el proveedor y todas las tuplas de artículos suministrados por él.
- Anular: Se asignan nulos en la clave foránea de todas las tuplas que la referencien y se elimina la tupla referenciada. En nuestro ejemplo, no tiene mucho sentido, pero consistiría en asignar nulos al NIF-PROV de todas las tuplas de articulos pertenecientes al proveedor que queremos borrar, y posteriormente borrar al proveedor.

- ¿Qué deberá suceder si hay un intento de modificar la clave primaria de una tupla referenciada por una clave foránea? Por ejemplo, si se intenta modificar la clave de un proveedor del cual existe algún artículo. Se actúa con las mismas tres posibilidades que en el caso anterior:

  - Restringir
  - Propagar en cascada
  - Anular

Ejemplo

```
CREATE TABLE person (
        dni             integer PRIMARY KEY,
        name            text
);

CREATE TABLE address(
        dni             integer REFERENCES person (dni) NOT NULL,
        street          text NOT NULL,
        zip              text
);

INSERT INTO person VALUES(25888777, 'Juan Perez');
OK

INSERT INTO address VALUES(25888777, 'Rivadavia 1234', '1828');
OK

INSERT INTO address VALUES(25888775, 'Rivadavia 1236', '1828');
ERROR DE INTEGRIDAD!
```

Restricción de Valor por Defecto (DEFAULT)

Permite que cuando se inserte una tupla o registro en una tabla, para aquellos atributos para los cuales no se indique un valor exacto se les asigne un valor por defecto.

Ejemplo

```
CREATE TABLE address(
        dni             integer REFERENCES person (dni) NOT NULL,
        street          text NOT NULL,
        zip             text DEFAULT 'S/N'
);

INSERT INTO address(dni, street) VALUES(25888777, 'Rivadavia 1234');
OK

SELECT * FROM address;

25888777 | 'Rivadavia 1234' | 'S/N'
```

Restricción de Verificación o Chequeo (CHECK)

En algunos casos puede ocurrir que sea necesario especificar una condición que deben cumplir los valores de determinados atributos de una relación de la BD, aparte de las restricciones vistas anteriormente.

Ejemplo

```
CREATE TABLE products (
        product_id      integer PRIMARY KEY,
        name            text NOT NULL,
        price           double precision NOT NULL CHECK (price > 0)
);

INSERT INTO product(1, 'Destornilaldor', 22);
OK

INSERT INTO product(2, 'Destornilaldor', 0);
ERROR
```

Aserciones (ASSERTION)

Esta restricción generaliza a la anterior, lo forman las aserciones en las que la condición se establece sobre elementos de distintas relaciones (por ello debe tener un nombre que la identifique).

Ejemplo

Ejemplos de aserciones y disparadores se verán más adelante.

Disparadores (TRIGGERS)

A veces puede interesar especificar una acción distinta del rechazo cuando no se cumple una determinada restricción semántica. En este caso, se recurre al uso de disparadores o triggers que nos permiten además de indicar una condición, especificar la acción que queremos que se lleve a cabo si la condición se hace verdadera. Los disparadores pueden interpretarse como reglas del tipo evento-condición-acción (ECA) que pueden interpretarse como reglas que especifican que cuando se produce un evento, si se cumple una condición, entonces se realiza una determinada acción.

Ejemplo

Ejemplos de aserciones y disparadores se verán más adelante.

Ejercitación

1) Crear una base de datos relacional para un instituto de formación profesional. Se debe registrar alumnos, docentes y cursos. Determinar qué alumnos están inscriptos en cada curso, y qué docentes están dictando cada curso.

Respecto a los alumnos se desea saber:

- Nombre (obligatorio)
- Apellido (obligatorio)
- DNI (obligatorio y único)
- Fecha de Nacimiento (obligatorio)
- Dirección (obligatorio)
- Teléfono
- eMail
- Fecha de Inscripción (obligatorio)
- Número de legajo (obligatorio y único)
- Qué curso está realizando

Respecto a los docentes se desea saber:

- Nombre (obligatorio)
- Apellido (obligatorio)
- DNI (obligatorio y único)
- Fecha de Nacimiento (obligatorio)
- Dirección (obligatorio)
- Teléfono
- eMail
- Fecha de contratación (obligatorio)
- CUIL (obligatorio y único)
- Honorarios (obligatorio, y debe ser mayor que cero)
- De qué curso es docente

Respecto a los cursos se desea saber:

- Nombre del curso (obligatorio y único)
- Descripción (obligatorio)
- Código del curso (obligatorio y único)
- Costo del curso (obligatorio, y debe ser mayor que cero)