Lenguaje para descripción de datos

Amparo López Gaona

Posgrado en Ciencia e Ingeniería de la Computación
Fac. Ciencias, UNAM
Mayo 2012

Definición de Datos

El lenguaje para definición de datos permite especificar:

- Esquema de cada relación.
- El dominio de cada atributo.
- Restricciones de integridad.
- Índices.
- Información de seguridad y autorización para cada relación.

Creación de esquemas

Los esquemas de relación incluyen:

- Nombre de la relación.
- Nombre de cada atributo.
- Dominio para cada atributo.
- Restricciones de integridad. Especificar:
  - No aceptar nulos en un atributo.
  - Valor por omisión.
  - Llaves (primarias, externas).
  - Verificación de límites para los datos.

Creación de esquemas

Los esquemas de relación incluyen:

- Nombre de la relación.
- Nombre de cada atributo.
- Dominio para cada atributo.
- Restricciones de integridad. Especificar:
  - No aceptar nulos en un atributo.
  - Valor por omisión.
  - Llaves (primarias, externas).
  - Verificación de límites para los datos.

```
CREATE TABLE R (A1  D1,
                A2  D2,
                ...
                An  Dn,
                restricción de integridad1
                ...
                restricción de integridadm );
```

Dominios

Los principales tipos de datos trabajados en SQL son:

- Cadenas de caracteres.
  - De longitud fija `CHAR(n)`.
  - De longitud variable `VARCHAR(n)`.
- Números enteros. `INT`, `INTEGER`, `SHORTINT`.
- Números reales. `FLOAT`, `REAL`, `DOUBLE PRECISION` y `DECIMAL(n,d)`.
- Fechas y horas. `DATE`, `TIME`. Ejemplo `DATE '2007-02-14'`.
  El formato para las horas es dos dígitos para la hora (24-horas), dos para los minutos y dos más para los segundos. Ejemplo: `TIME '15:00:02.5'`

Creación de Dominios

Se puede asociar un nombre a un tipo con la siguiente instrucción.

`CREATE DOMAIN` nombre tipo

```
CREATE DOMAIN DomCuenta CHAR(6);
CREATE DOMAIN DOMCuenta CHAR(6) DEFAULT 'C-0000';
```

Ejemplo

```
CREATE TABLE cuenta (
            nombreSucursal char(20),
            num_cuenta DomCuenta,
            saldo integer);
```

Ejemplo

```
CREATE TABLE cuenta (
            nombreSucursal char(20),
            num_cuenta DomCuenta,
            saldo integer);

CREATE TABLE cliente (
            nombreCliente char(20),
            calle char(30) ,
            ciudad char (30) default 'Mexico');
```

Restricciones de Integridad

Las principales restricciones de integridad que pueden indicarse son:

- La llave primaria.

Restricciones de Integridad

Las principales restricciones de integridad que pueden indicarse son:

- La llave primaria. `PRIMARY KEY` $(A_1, A_2, ..., A_n)$
- Llaves candidatas.

Restricciones de Integridad

Las principales restricciones de integridad que pueden indicarse son:

- La llave primaria. `PRIMARY KEY` $(A_1, A_2, ..., A_n)$
- Llaves candidatas. `UNIQUE`
- Llaves externas.

Restricciones de Integridad

Las principales restricciones de integridad que pueden indicarse son:

- La llave primaria. `PRIMARY KEY` $(A_1, A_2, ..., A_n)$
- Llaves candidatas. `UNIQUE`
- Llaves externas. `FOREIGN KEY` $A_1, A_2, ..., A_n$ `REFERENCES` tabla $(B_1, B_2, ..., B_n)$
- Valores no nulos.

Restricciones de Integridad

Las principales restricciones de integridad que pueden indicarse son:

- La llave primaria. `PRIMARY KEY` $(A_1, A_2, ..., A_n)$
- Llaves candidatas. `UNIQUE`
- Llaves externas. `FOREIGN KEY` $A_1, A_2, ..., A_n$ `REFERENCES` tabla $(B_1, B_2, ..., B_n)$
- Valores no nulos. `NOT NULL`.
- Verificación de cierta condición.

Restricciones de Integridad

Las principales restricciones de integridad que pueden indicarse son:

- La llave primaria. `PRIMARY KEY` $(A_1, A_2, ..., A_n)$
- Llaves candidatas. `UNIQUE`
- Llaves externas. `FOREIGN KEY` $A_1, A_2, ..., A_n$ `REFERENCES` tabla $(B_1, B_2, ..., B_n)$
- Valores no nulos. `NOT NULL`.
- Verificación de cierta condición. `CHECK` condición, `ASSERT` condición.

Llaves Primarias

Existen dos formas de definir una llave primaria al crear una tabla:

- Al definir el elemento especificar que será llave primaria:

```
CREATE TABLE persona (
  nombre    CHAR(30) PRIMARY KEY,
  dir       VARCHAR(200),
  sexo      CHAR(1),
  fechaNac DATE
);
```

Llaves Primarias

Existen dos formas de definir una llave primaria al crear una tabla:

- Al definir el elemento especificar que será llave primaria:

```
CREATE TABLE persona (
  nombre    CHAR(30) PRIMARY KEY,
  dir       VARCHAR(200),
  sexo      CHAR(1),
  fechaNac DATE
);
```

- Agregar una declaración al final de la lista de atributos.

```
CREATE TABLE persona (
  nombre    CHAR(30),
  dir       VARCHAR(200),
  sexo      CHAR(1),
  fechaNac DATE,
  PRIMARY KEY (nombre)
);
```

Llaves

Otra forma de definir llaves es usando la palabra `UNIQUE`.

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200),
  telefono CHAR(8) UNIQUE,
  sexo     CHAR(1),
  fechanac DATE
);
```

En llaves primarias no se aceptan nulos, con `UNIQUE` sí se permiten y puede haber varios `UNIQUE`.

Llaves externas

Cuando una llave c en una relación R aparece como atributo en otra relación S, se dice que c es una **llave externa** en S.
Sea R_2 una relación, se define una **llave externa** como un subconjunto FK, del conjunto de atributos de R_2 tales que:

- Existe una relación R_1 con una llave FK, y
- Cada valor de FK en R_2 es idéntico al valor de FK en alguna tupla de R_1.

Llaves externas

Cuando una llave c en una relación R aparece como atributo en otra relación S, se dice que c es una **llave externa** en S.
Sea R_2 una relación, se define una **llave externa** como un subconjunto FK, del conjunto de atributos de R_2 tales que:

- Existe una relación R_1 con una llave FK, y
- Cada valor de FK en R_2 es idéntico al valor de FK en alguna tupla de R_1.

Terminología:

- El valor de una llave externa representa una referencia a la tupla que contiene el valor de la llave.
- Regla de integridad referencial:

Llaves externas

Cuando una llave c en una relación R aparece como atributo en otra relación S, se dice que c es una **llave externa** en S.
Sea R_2 una relación, se define una **llave externa** como un subconjunto FK, del conjunto de atributos de R_2 tales que:

- Existe una relación R_1 con una llave FK, y
- Cada valor de FK en R_2 es idéntico al valor de FK en alguna tupla de R_1.

Terminología:

- El valor de una llave externa representa una referencia a la tupla que contiene el valor de la llave.
- Regla de integridad referencial: La BD no debe contener valores de llave externa que no se correspondan con un valor de la llave candidata (Si B referencia a A, entonces A debe existir).

...Llaves externas

Cuenta:

nSucursal	*numCta*	saldo
Centro	C-101	100,000
San Angel	C-215	140,000
Las Fuentes	C-102	80,000
La Selva	C-305	70,000
Cuenca	C-201	180,000
Perinorte	C-222	140,000
Cuenca	C-217	150,000

Sucursal:

nombre	ciudad	activo
Centro	México D.F.	1,800,000,000
Perinorte	Cd. Sátelite	420,000,000
Las Fuentes	México D.F.	340,000,000
San Angel	México D.F.	80,000,000
La Selva	Cuernavaca	1,600,000,000
Ocoxingo	Cuernavaca	60,000,000
Segovia	Arganzuela	1,420,000,000
Cuenca	Cd. Satélite	15,000,000

...Llaves externas

Formas de declarar las llaves externas:

- Si sólo tiene un atributo basta con terminar su declaración con `REFERENCES` *tabla (atributo)* donde el *atributo* es la llave en la tabla especificada.

```
CREATE TABLE sucursal (
             nombreSucursal char(20) PRIMARY KEY,
             ciudad         char(30),
             activo         int);

CREATE TABLE cuenta (
    nSucursal char(20) REFERENCES sucursal (nombreSucursal
    numCuenta     DomCuenta PRIMARY KEY,
    saldo int);
```

...Llaves externas

- Especificarlo en la lista de restricciones.
  `FOREIGN KEY` *atributos* `REFERENCES` *tabla (atributos)*

```
CREATE TABLE cuenta (
                nombreSucursal char(20);
                numCuenta DomCuenta PRIMARY KEY,
                saldo int,
                FOREIGN KEY nombreSucursal
                        REFERENCES sucursal (nombreSucursal))
```

Todo valor de `nombreSuc` en `Cuenta` debe estar en la relación `Sucursal`.

...Llaves externas (Mantenimiento)

1. Política por omisión. Rechazar las modificaciones que signifiquen violación a la integridad referencial.

...Llaves externas (Mantenimiento)

1. Política por omisión. Rechazar las modificaciones que signifiquen violación a la integridad referencial.
   - Al tratar de insertar un nueva cuenta cuyo valor de sucursal no sea `NULL` y no esté en la relación `sucursal`.
   - Al tratar de actualizar una cuenta cambiando el componente de la sucursal a uno no existente.
   - Al tratar de borrar de `sucursal` una tupla cuyo valor aparezca en al menos una tupla de `cuenta`.
   - Al tratar de actualizar una tupla de `sucursal` y se caiga en lo anterior.
2. Política de cascada.
   - Al suprimir una tupla cuyo valor sea referenciado desde otra relación, para mantener la integridad referencial se eliminan la tuplas referenciantes.
   - Al actualizar el valor de una tupla en `sucursal` se actualizan las de `cuenta`.
3. Asignación de nulos. Al cambiar un valor en el nombre de la sucursal referenciada, se asigna `NULL` en las relaciones referenciantes.

...Llaves externas (Mantenimiento)

El usuario puede elegir de manera independiente qué hacer:

```
CREATE TABLE
    cuenta (
        nombreSucursal char(20);
        numCuenta DomCuenta PRIMARY KEY,
        saldo int,
        FOREIGN KEY nombreSucursal
                  REFERENCES sucursal (nombreSucursal)
        ON DELETE SET NULL
        ON UPDATE CASCADE);
```

Actualización de definición de tablas

```
ALTER TABLE <tabla> <accion>
<accion> ::=
    ADD columna tipoDeDato
    | ADD restriccion
    | ALTER columna DROP DEFAULT
    | ALTER columna SET DEFAULT valor
    | DROP columna [RESTRICT | CASCADE]
    | DROP restriccion  [RESTRICT | CASCADE]
```

Al eliminar una columna se puede usar el modificador `RESTRICT` para indicar que la eliminación puede realizarse sólo si no es referida por algún otro elemento.
Si se utiliza `CASCADE` también se elimina cualquier otro elemento que haga referencia a esa columna.

Actualización de definición de tablas

```
ALTER TABLE <tabla> <accion>
<accion> ::=
    ADD columna tipoDeDato
    | ADD restriccion
    | ALTER columna DROP DEFAULT
    | ALTER columna SET DEFAULT valor
    | DROP columna [RESTRICT | CASCADE]
    | DROP restriccion  [RESTRICT | CASCADE]
```

Al eliminar una columna se puede usar el modificador `RESTRICT` para indicar que la eliminación puede realizarse sólo si no es referida por algún otro elemento.
Si se utiliza `CASCADE` también se elimina cualquier otro elemento que haga referencia a esa columna.

```
ALTER TABLE cliente ADD telefono CHAR(8)
                    ALTER calle SET DEFAULT = 'Insurgentes';
ALTER TABLE cliente ADD fechaNac DATE;
ALTER TABLE cliente DROP fechaNac;
ALTER TABLE sucursal DROP nombreSucursal RESTRICT;
```

...Actualización de definición de tablas

```
CREATE TABLE z
(llaveDeZ   VARCHAR(3) PRIMARY KEY,
 refX       VARCHAR(3) REFERENCES x(llaveDeX));
CREATE TABLE y
(llaveDeY   VARCHAR(3) PRIMARY KEY,
 refZ       VARCHAR(3) REFERENCES z(llaveDeZ));
CREATE TABLE x
(llaveDeX   VARCHAR(3) PRIMARY KEY,
 refY       VARCHAR(3) REFERENCES y(llaveDeY));
```

...Actualización de definición de tablas

```
CREATE TABLE z
(llaveDeZ   VARCHAR(3) PRIMARY KEY,
 refX       VARCHAR(3) REFERENCES x(llaveDeX));
CREATE TABLE y
(llaveDeY   VARCHAR(3) PRIMARY KEY,
 refZ       VARCHAR(3) REFERENCES z(llaveDeZ));
CREATE TABLE x
(llaveDeX   VARCHAR(3) PRIMARY KEY,
 refY       VARCHAR(3) REFERENCES y(llaveDeY));
```

Error en la compilación: z no puede refererise a x antes de ser definida.

...Actualización de definición de tablas

```
CREATE TABLE z
(llaveDeZ   VARCHAR(3) PRIMARY KEY,
 refX       VARCHAR(3) REFERENCES x(llaveDeX));
CREATE TABLE y
(llaveDeY   VARCHAR(3) PRIMARY KEY,
 refZ       VARCHAR(3) REFERENCES z(llaveDeZ));
CREATE TABLE x
(llaveDeX   VARCHAR(3) PRIMARY KEY,
 refY       VARCHAR(3) REFERENCES y(llaveDeY));
```

Error en la compilación: z no puede refererise a x antes de ser definida.

```
CREATE TABLE z
(llaveDeZ   VARCHAR(3) PRIMARY KEY);
CREATE TABLE y ( ... );
CREATE TABLE x ( ... );
ALTER TABLE z  ADD refX VARCHAR(3) REFERENCES x(llaveDeX);
```

Eliminación de definición de tablas

Si por alguna causa, ya no es necesaria una relación se puede eliminar por medio de la instrucción:
`DROP <tabla> [RESTRICT | CASCADE]`
`RESTRICT` significa que la tabla no puede eliminarse si es referenciada por otra tabla.
`CASCADE` Las tablas referenciantes también se eliminan.

Restricciones al valor de los atributos

- Restricciones de dominios para un atributo:
  - Valores no nulos.
  - Especificación de valores en la definición de un atributo.
  - Especificación de valores en la definición del dominio.
- Restricciones sobre tuplas.
- Restricciones sobre relaciones.

Valores nulos

Al insertar una tupla sin especificar todos los campos se introduce un valor NULL, a menos que haya un valor por omisión.

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200),
  telefono CHAR(8)
);
```

Valores nulos

Al insertar una tupla sin especificar todos los campos se introduce un valor NULL, a menos que haya un valor por omisión.

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200),
  telefono CHAR(8)
);
```

```
INSERT INTO persona (nombre) VALUES ('Andrea'); ?
```

"Desconozco el valor de un atributo pero existe o existirá"

...Valores nulos

```
CREATE TABLE persona (
  nombre    CHAR(30) PRIMARY KEY,
  dir       VARCHAR(200) DEFAULT 'Callejon del sapo 13',
  telefono CHAR(8)
);

INSERT INTO persona (nombre) VALUES ('Andrea'); ?
```

...Valores nulos

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200) DEFAULT 'Callejon del sapo 13',
  telefono CHAR(8)
);

INSERT INTO persona (nombre) VALUES ('Andrea'); ?
INSERT INTO persona (telefono) VALUES ('56224866'); ?
```

...Valores nulos

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200) DEFAULT 'Callejon del sapo 13',
  telefono CHAR(8)
);

INSERT INTO persona (nombre) VALUES ('Andrea'); ?
INSERT INTO persona (telefono) VALUES ('56224866'); ?
INSERT INTO persona (nombre, telefono) VALUES
('Andrea','56224866'); ?
```

...Valores nulos

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200) DEFAULT 'Callejon del sapo 13',
  telefono CHAR(8)
);

INSERT INTO persona (nombre) VALUES ('Andrea'); ?
INSERT INTO persona (telefono) VALUES ('56224866'); ?
INSERT INTO persona (nombre, telefono) VALUES
('Andrea','56224866'); ?
INSERT INTO persona (nombre, telefono) VALUES
('Karla','56224866'); ?
```

...Valores nulos

Si se desea evitar la inserción de nulos se debe especificar en la creación de la relación:

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200) DEFAULT 'Callejon del sapo 13',
  telefono CHAR(8) NOT NULL
);

INSERT INTO persona (nombre) VALUES ('Andrea'); ?
```

Consecuencias:

...Valores nulos

Si se desea evitar la inserción de nulos se debe especificar en la creación de la relación:

```
CREATE TABLE persona (
  nombre   CHAR(30) PRIMARY KEY,
  dir      VARCHAR(200) DEFAULT 'Callejon del sapo 13',
  telefono CHAR(8) NOT NULL
);

INSERT INTO persona (nombre) VALUES ('Andrea'); ?
```

Consecuencias:

- No se puede dejar de especificar el teléfono al insertar una tupla.
- No se puede modificar el teléfono a nulo.

...Valores nulos

Reglas importantes para tratar con los nulos.

- NULL no es una constante, por tanto no puede usarse como operando. Excepto:

```
SELECT numPrestamo
FROM   prestamo
WHERE  importe IS NULL;
```

...Valores nulos

Reglas importantes para tratar con los nulos.

- NULL no es una constante, por tanto no puede usarse como operando. Excepto:

```
SELECT numPrestamo
FROM   prestamo
WHERE  importe IS NULL;
```

- El resultado de una operación aritmética es nulo si cualquiera de los operandos es nulo. Es decir, es diferente de cero:

```
0 * NULL = NULL;
NULL - NULL = NULL;
0 - NULL = NULL;
```

...Valores nulos

Reglas importantes para tratar con los nulos.

- `NULL` no es una constante, por tanto no puede usarse como operando. Excepto:

```
SELECT numPrestamo
FROM   prestamo
WHERE  importe IS NULL;
```

- El resultado de una operación aritmética es nulo si cualquiera de los operandos es nulo. Es decir, es diferente de cero:

```
0 * NULL = NULL;
NULL - NULL = NULL;
0 - NULL = NULL;
```

- Las operaciones de agregación ignoran los valores nulos, con lo cual puede resultar vacía en cuyo caso devuelven nulo, excepto `COUNT(*)` que regresa cero.

```
SELECT SUM(importe)
FROM   prestamo;
```

...Valores nulos

- Las operaciones lógicas que involucran nulos, lo consideran como falso, y da como resultado desconocido, lo cual no es cierto ni falso.

...Valores nulos

- Las operaciones lógicas que involucran nulos, lo consideran como falso, y da como resultado desconocido, lo cual no es cierto ni falso.

Reglas para el tratamiento de desconocidos en operaciones lógicas:

x	*y*	*x* AND *y*	*x* OR *y*	NOT *x*
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true
true	unknown	unknown	true	false
unknown	true	unknown	true	unknown
false	unknown	false	unknown	true
unknown	false	false	unknown	unknown
unknown	unknown	unknown	unknown	unknown

...Valores nulos

- Las operaciones lógicas que involucran nulos, lo consideran como falso, y da como resultado desconocido, lo cual no es cierto ni falso.

Reglas para el tratamiento de desconocidos en operaciones lógicas:

x	*y*	*x* AND *y*	*x* OR *y*	NOT *x*
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true
true	unknown	unknown	true	false
unknown	true	unknown	true	unknown
false	unknown	false	unknown	true
unknown	false	false	unknown	unknown
unknown	unknown	unknown	unknown	unknown

En la clausula `WHERE` se produce uno de estos tres valores pero sólo trabaja con las tuplas que producen verdadero.

...Valores nulos

¿Las siguientes consultas producen el mismo resultado?

```
SELECT *
FROM   prestamo
WHERE  importe <= 10000 OR importe > 10000;

SELECT *
FROM   prestamo;
```

...Valores nulos

¿Las siguientes consultas producen el mismo resultado?

```
SELECT *
FROM   prestamo
WHERE  importe <= 10000 OR importe > 10000;

SELECT *
FROM   prestamo;
```

Son distintos si hay al menos una tupla con importe = NULL

...Valores nulos

¿Las siguientes consultas producen el mismo resultado?

```
SELECT *
FROM   prestamo
WHERE  importe <= 10000 OR importe > 10000;

SELECT *
FROM   prestamo;
```

Son distintos si hay al menos una tupla con importe = NULL
Por tanto la consulta es en realidad "encontrar los préstamos con importe no-nulo".

Restricciones basadas en un atributo

CHECK (condición)

La condición puede ser cualquiera que pueda aparecer en la clausula WHERE, es decir puede tener operadores lógicos, de comparación, LIKE, IN, BETWEEN, etc.
Se verifica cada vez que se modifica el valor de ese atributo.
Si no se cumple la condición se rechaza la modificación.
Ejemplo:

```
CREATE TABLE persona (
  nombre   CHAR(30),
  dir      VARCHAR(200),
  telefono CHAR(8) CHECK (telefono LIKE '5-------'),
  sexo     CHAR(1) CHECK (sexo IN ('F', 'M'))
);
```

... Restricciones basadas en un atributo

Con las siguientes relaciones, definir la tabla `Estudio` de tal forma que el presidente del estudio sea un ejecutivo.

```
Estudio (nombre, direccion, curpPresidente)
Ejecutivo (nombre, direccion, curp, sueldo)
```

... Restricciones basadas en un atributo

Con las siguientes relaciones, definir la tabla `Estudio` de tal forma que el presidente del estudio sea un ejecutivo.

```
Estudio (nombre, direccion, curpPresidente)
Ejecutivo (nombre, direccion, curp, sueldo)
```

Si se desea utilizar otros atributos o tuplas de la relación, o de otras relaciones, la condición debe ser una subconsulta.

```
CREATE TABLE Estudio (
  nombre CHAR(40) PRIMARY KEY,
  direccion VARCHAR(255),
  curpPresidente INT CHECK (curpPresidente
                            IN (SELECT curp FROM ejecutivo))
)
```

... Restricciones basadas en un atributo

Con las siguientes relaciones, definir la tabla `Estudio` de tal forma que el presidente del estudio sea un ejecutivo.

```
Estudio (nombre, direccion, curpPresidente)
Ejecutivo (nombre, direccion, curp, sueldo)
```

Si se desea utilizar otros atributos o tuplas de la relación, o de otras relaciones, la condición debe ser una subconsulta.

```
CREATE TABLE Estudio (
  nombre CHAR(40) PRIMARY KEY,
  direccion VARCHAR(255),
  curpPresidente INT CHECK (curpPresidente
                            IN (SELECT curp FROM ejecutivo))
)
```

La validación es correcta y es como restricción de integridad referencial, excepto que la validación se realiza sólo al agregar o modificar una tupla de `Estudio`, no al suprimir una tupla de `Ejecutivo`.

Restricciones de dominio

Es similar al punto anterior excepto que no se sabe cuál atributo tomará valores de ese dominio, en ese caso se escribe la palabra `VALUE`

```
CREATE DOMAIN domSexo CHAR(1) CHECK (VALUE IN ('F', 'M'));
```

```
sexo domSexo,
```

```
CREATE DOMAIN domP INT CHECK ( VALUE >= 1000);
```

Restricciones globales

Restricciones que involucran relación entre varios atributos o entre diferentes relaciones.

- Restricciones basadas en tupla. Restringen aspectos de cualquier tupla en una relación.

Restricciones globales

Restricciones que involucran relación entre varios atributos o entre diferentes relaciones.

- Restricciones basadas en tupla. Restringen aspectos de cualquier tupla en una relación.
- Afirmaciones (Assertions). Restricciones que pueden involucrar relaciones completas o varias tuplas de una relación.

Restricciones basadas en tuplas

- Se especifican en la lista de restricciones.
- Sintaxis: `CHECK` (condición)
  pero la condición se puede referir a cualquier atributo de la relación.
  (o de otras relaciones vía subconsultas).
- Se verifican cada vez que se pretende insertar/modificar una tupla de la relación. Si la evaluación da falso se rechaza la operación.

Ejemplo: Sólo el departamento de artículos de lujo puede tener artículos de más de $10,000.00

```
CREATE TABLE ventas (
                departamento   CHAR(30),
                articulo       VARCHAR(200),
                precio         REAL,
```

Restricciones basadas en tuplas

- Se especifican en la lista de restricciones.
- Sintaxis: `CHECK` (condición)
  pero la condición se puede referir a cualquier atributo de la relación.
  (o de otras relaciones vía subconsultas).
- Se verifican cada vez que se pretende insertar/modificar una tupla de la relación. Si la evaluación da falso se rechaza la operación.

Ejemplo: Sólo el departamento de artículos de lujo puede tener artículos de más de $10,000.00

```
CREATE TABLE ventas (
                departamento  CHAR(30),
                articulo      VARCHAR(200),
                precio        REAL,

        CHECK (departamento = 'lujoso' OR precio <= 10000.00)
                    );
```

Afirmaciones

Son condiciones que se desea se cumplan en toda la base de datos, no sólo en una relación.

`CREATE ASSERTION` nombre `CHECK` (condición);

Al crear una afirmación, el sistema comprueba su validez.
Se verifican cuando alguna de las relaciones mencionadas cambia.
La comprobación puede introducir una sobrecarga importante.

Afirmaciones

Son condiciones que se desea se cumplan en toda la base de datos, no sólo en una relación.

`CREATE ASSERTION` nombre `CHECK` (condición);

Al crear una afirmación, el sistema comprueba su validez.
Se verifican cuando alguna de las relaciones mencionadas cambia.
La comprobación puede introducir una sobrecarga importante.
Por ejemplo, NO puede haber más sucursales que clientes.

Afirmaciones

Son condiciones que se desea se cumplan en toda la base de datos, no sólo en una relación.

`CREATE ASSERTION` nombre `CHECK` (condición);

Al crear una afirmación, el sistema comprueba su validez.
Se verifican cuando alguna de las relaciones mencionadas cambia.
La comprobación puede introducir una sobrecarga importante.
Por ejemplo, NO puede haber más sucursales que clientes.

```
CREATE ASSERTION pocos
   CHECK ( (SELECT COUNT (*) FROM sucursal) <=
           (SELECT COUNT (*) FROM cliente) );
```

Se verifica cada vez que `sucursal` o `cliente` cambien.

...Afirmaciones

La suma de los importes de los préstamos de cada sucursal debe ser menor que la suma de los saldos de cuentas de esa sucursal.

```
CREATE ASSERTION suma_restringida CHECK
  (NOT EXISTS (SELECT *
               FROM sucursal
               WHERE (SELECT SUM(saldo)
                      FROM cuenta
                      WHERE cuenta.nombreSucursal =
                            sucursal.nombreSucursal)
                  >= (SELECT SUM(importe)
                      FROM prestamo
                      WHERE prestamo.nombreSucursal =
                            sucursal.nombreSucursal)
```

Se puede escribir como restricción dentro de sucursal, pero sólo se validaría al modificar esta tabla, no al modificar Cuenta.

...Afirmaciones

Verificar que la suma de los saldos de las cuentas por sucursal sea mayor que 10 millones de pesos.

```
CREATE ASSERTION sumaCuenta CHECK(10000000 <= ALL
        (SELECT SUM(saldo) FROM Cuenta GROUP BY nombreSucursa
```

...Afirmaciones

Verificar que la suma de los saldos de las cuentas por sucursal sea mayor que 10 millones de pesos.

```
CREATE ASSERTION sumaCuenta CHECK(10000000 <= ALL
        (SELECT SUM(saldo) FROM Cuenta GROUP BY nombreSucursa
```

Ésta involucra sólo a `Cuenta`, así que podría haberse escrito como `CHECK` basado en tupla.

```
CREATE TABLE Cuenta {
  ...
  CHECK(10000000 <= ALL
        (SELECT SUM(saldo) FROM prestamo GROUP BY nombreSucur
```

pero ...

...Afirmaciones

Verificar que la suma de los saldos de las cuentas por sucursal sea mayor que 10 millones de pesos.

```
CREATE ASSERTION sumaCuenta CHECK(10000000 <= ALL
        (SELECT SUM(saldo) FROM Cuenta GROUP BY nombreSucursa
```

Ésta involucra sólo a `Cuenta`, así que podría haberse escrito como `CHECK` basado en tupla.

```
CREATE TABLE Cuenta {
  ...
  CHECK(10000000 <= ALL
        (SELECT SUM(saldo) FROM prestamo GROUP BY nombreSucur
```

pero ... La validación no se efectuaría al suprimir tuplas.

Comparación de restricciones

Restricción	Declaración	Activación	¿Garantizada?
`CHECK` basado en atributos	con el atributo	inserción de tupla modificación de atributo	no, si hay subconsultas
`CHECK` basado en tuplas	elemento del esquema de la relación	inserción de tupla modificación de tupla	no, si hay subconsultas
`ASSERTION`	elemento del esquema de la base de datos	Al cambiar cualquier relación mencionada	Sí

Modificación de restricciones

Es posible modificar las restricciones en cualquier momento.
Para poder eliminar o modificar alguna restricción es necesario asignarle nombre.
Se debe preceder la restricción por `CONSTRAINT` *nombre de la restricción*.

```
nombreSucursal CHAR(30) CONSTRAINT nombreLlave PRIMARY KEY,

saldo CHAR(1) CONSTRAINT saldoPositivo
              CHECK (saldo >= 0),

CREATE DOMAIN dominioCP INT
       CONSTRAINT seisDigitos CHECK (VALUE >= 1000000);

CONSTRAINT xxxx
   CHECK (sexo 'm' OR name like 'Maria %')
```

...Modificación de restricciones

Modificar restricciones

- `ALTER TABLE` Para agregar o eliminar columnas.

...Modificación de restricciones

Modificar restricciones

- `ALTER TABLE` Para agregar o eliminar columnas.
  Para cambiar restricciones basadas en atributos y en tuplas.
- `ALTER DOMAIN` Para cambiar el valor por omisión de un dominio.

...Modificación de restricciones

Modificar restricciones

- `ALTER TABLE` Para agregar o eliminar columnas.
  Para cambiar restricciones basadas en atributos y en tuplas.
- `ALTER DOMAIN` Para cambiar el valor por omisión de un dominio.

Ejemplos:

```
ALTER TABLE sucursal DROP CONSTRAINT nombreLlave;

ALTER TABLE cuenta DROP CONSTRAINT saldoPositivo;

ALTER TABLE persona DROP CONSTRAINT dominioCP;

ALTER TABLE persona DROP CONSTRAINT xxxx;

ALTER TABLE sucursal ADD CONSTRAINT nombreLlave
                                    PRIMARY KEY (nombre);
```

...Modificación de restricciones

Modificar restricciones de dominio

```
ALTER DOMAIN dominioX DROP CONSTRAINT restriccionX;

ALTER DOMAIN dominioX ADD CONSTRAINT restriccionX
                                    CHECK (VALUE ...) ;
```

Eliminar afirmaciones

```
DROP ASSERTION nombre;
```

Disparadores

Un disparador (*trigger*) es una secuencia de acciones que el sistema ejecuta de manera automática como efecto secundario de una modificación a la BD.
Difieren de las restricciones antes mencionadas en:

- Sólo son verificados cuando ocurre cierto *evento*. Tal evento se especifica por el programador de la BD.
- Una vez que se activa el disparador, prueba una *condición*, si ésta se satisface se ejecuta la *acción* asociada con el disparador. En caso contrario no pasa nada.

Disparadores

Un disparador (*trigger*) es una secuencia de acciones que el sistema ejecuta de manera automática como efecto secundario de una modificación a la BD.
Difieren de las restricciones antes mencionadas en:

- Sólo son verificados cuando ocurre cierto *evento*. Tal evento se especifica por el programador de la BD.
- Una vez que se activa el disparador, prueba una *condición*, si ésta se satisface se ejecuta la *acción* asociada con el disparador. En caso contrario no pasa nada.

Para diseñar un disparador se debe:

- Especificar las condiciones en las que se va a ejecutar el disparador (evento, condición).
- Especificar las acciones que se van a realizar cuando se active el disparador.

...Disparadores

Ejemplo. En lugar de tener saldos negativos, se hace un préstamo por el importe de dicho saldo.

```
DEFINE TRIGGER descubierto ON UPDATE OF cuenta T
  (IF NEW T.saldo < 0 then (
                INSERT INTO prestamo VALUES
                    (T.nombreSucursal, T.numCuenta, - new T.sald
                INSERT INTO prestatario
                    (SELECT nombre_cliente, numCuenta
                     FROM   cta_cliente
                     WHERE  T.numCuenta = cta_cliente.numCuenta)
                UPDATE cuenta S SET S.saldo = 0
                     WHERE S.numCuenta = T.numCuenta))
```

...Disparadores

Se puede elegir entre diferentes opciones para las partes de evento, condición de un disparador:

- La acción puede ejecutarse antes o después del evento.
- La acción puede referirse a los valores acualizados o a los anteriores.
- Los eventos de actualización pueden limitarse a un atributo o a un conjunto de atributos.
- Se puede especificar una condición en la clausula `WHERE`.
- Se puede especicar que la acción se realice ya sea sobre cada tupla modificada o bien sobre todas las tuplas que son cambiadas en una operación de la BD.