

BASES DE DATOS

DSIC. Curso 1999-2000

Práctica Definición de Datos (SQL92)

Objetivos

- Presentar el lenguaje de definición de datos del SQL/92
 - Presentar la sintaxis de creación de tablas
 - Presentar la sintaxis de la modificación de definición de una tabla
 - Presentar la sintaxis de la creación de una vista
 - Presentar la sintaxis para otorgar autorizaciones

 - Presentar la sintaxis para la creación de reglas de actividad (disparadores)
- Todo ello para el sistema Oracle 8.x

Definición de Datos (SQL)

Instrucciones del SQL para poder definir esquemas relacionales:

- **create schema:** permite dar nombre a un esquema relacional y declarar el usuario que es el creador y propietario de dicho esquema.

- **create domain:** permite definir un nuevo dominio de datos.

ORACLE

- **create table:** define una tabla, su esquema y las restricciones asociadas.

ORACLE

- **create view:** define una vista o relación derivada en el esquema relacional.

- **create assertion:** permite definir restricciones de integridad generales.

ORACLE

- **grant:** permite definir autorización de operaciones sobre objetos de la BD.

Todas estas instrucciones tienen asociada la operación inversa (DROP / REVOKE) y modificación (ALTER).

Definición del Esquema (SQL)

```
create schema [esquema] [authorization usuario]  
[lista_elemento_esquema];
```

Un elemento de esquema puede ser uno de los siguientes:

- Definición de dominio.
- Definición de tabla.
- Definición de vista.
- Definición de restricción.
- Definición de privilegio.

Eliminación de la definición de un esquema relacional:

```
drop schema esquema {restrict | cascade};
```

Definición de Dominios (SQL)

```
create domain dominio [as] tipo_dato  
    [default { literal | función_sistema | null }]  
    [definición_restricción_dominio];
```

Funciones del sistema:

- user
- current_user
- session_user
- current_date
- current_time
- current_timestamp.

Definición de Dominios (SQL)

A un dominio se le puede asociar un conjunto de restricciones:

[constraint *restricción*]

check (expresión_condicional)

[not] deferrable

- *expresión_condicional* permite expresar cualquier condición que debe cumplir siempre el dominio (debe ser CIERTA o INDEFINIDA)
- **deferrable** indica que el sistema ha de comprobar la restricción al finalizar la transacción activa.
- **Not deferrable** indica que el sistema ha de comprobar la restricción después de cada operación de actualización a la base de datos.

Definición de Dominios (SQL). Ejemplo

```
CREATE DOMAIN ángulo AS FLOAT  
    DEFAULT 0  
    CHECK (VALUE >= 0 AND VALUE < 360)  
    NOT DEFERRABLE;
```

Eliminación de un Dominio:

```
drop domain dominio [restrict | cascade]
```

Definición de Tablas (SQL).

```
CREATE TABLE tabla
    comalista_definición_columna
    [comalista_definición_restricción_tabla];
```

La definición de una columna de una tabla se realiza como sigue:

```
columna {tipo_dato | dominio}
    [default {literal | función_sistema | null }]
    [lista_definición_restricción_columna]
```

Las restricciones que se pueden definir sobre las columnas son las siguientes:

- not null: restricción de valor no nulo.
- Definiciones de restricciones de CP, UNI, CAj de una sola columna.
- Definición de restricciones generales con la cláusula check.

Definición de Tablas (SQL).

La cláusula para definir restricciones de tabla es la siguiente:

[constraint *restricción*]

{ primary key (*comalista_columna*)

| unique (*comalista_columna*)

| foreign key (*comalista_columna*)

references *tabla*(*comalista_columna*)

[match {full | partial}] * NO ORACLE

[on update [cascade | * NO ORACLE

set null | set default | no action]] * NO ORACLE

[on delete [cascade |

set null | set default | no action]] * NO ORACLE

| check expresión_condicional }

[comprobación_restricción]

- debe ser CIERTA o INDEFINIDA.

- no puede incluir subconsultas ni referencias a otras tablas.

Ejemplo: Proveedor-Piezas-Suministro

d_cod_pieza: tira(4)

d_cod_proy: tira(4)

d_dni: entero (positivo)

Proveedor(dni: d_dni, nombre: tira(40), dirección: tira(25), ciudad: tira(30))

CP: {dni}

VNN: {nombre}

Pieza(código: d_cod_pieza, desc: tira(40), color: tira(20), peso: real)

CP: {código}

Suministro(dni: d_dni, código: tira(4), precio: real)

CP: {dni, código}

CAj: {dni} \rightarrow Proveedor

CAj: {código} \rightarrow Pieza

Restricciones de integridad:

R1) Px: Pieza $\forall Px (\text{Pieza}(Px) \wedge Px.\text{color}='rojo' \rightarrow Px.\text{peso}>100)$

R2) Px: Pieza, Sx: Suministro $\forall Px (\text{Pieza}(Px) \rightarrow \exists Sx (\text{Suministro}(Sx) \wedge Sx.\text{código}=Px.\text{código}))$

Ejemplo: Proveedor-Piezas-Suministro (SQL)

```
create schema Almacén
authorization pepe
create domain d_cod_pieza as char(4)
create domain d_cod_proy as char(4)
create domain d_dni as integer check value>0
create table Proveedor ( dni          d_dni primary key,
                        nombre       varchar(40) not null,
                        dirección     char(25),
                        ciudad        char(30) )

create table Pieza ( código         d_cod_pieza primary key,
                   desc            varchar(40) not null,
                   color           char(20),
                   peso            float,
                   constraint r1 check (color<>'rojo' or peso>100))

create table Suministro ( dni          d_dni,
                        código         d_cod_pieza references Pieza,
                        precio         float,
                        primary key (dni, código),
                        foreign key (dni) references Proveedor(dni) );
```

← R1

¿Y R2?

Definición de Tablas (SQL). Cláusula MATCH

- completa (**match full**): en cada tupla de R la clave ajena CA tiene el valor nulo o no lo tiene, en cada una de sus columnas. En el segundo caso, ha de existir una fila en la tabla S cuyo valor en las columnas de CU sea idéntico.
- parcial (**match partial**): en cada tupla de R la clave ajena CA tiene el valor nulo en cada una de sus columnas, o ha de existir una fila en la tabla S , de forma que para las columnas de la clave ajena CA que no tienen valor nulo, el valor en las columnas correspondientes de CU es idéntico.
- débil (**no se incluye cláusula match**): en cada tupla de R si la clave ajena CA no tiene el valor nulo, en cada una de sus columnas, ha de existir una fila en la tabla S tal que el valor en coincida en todas las columnas.

ORACLE

Modificación de Definición de Tablas (SQL).

Para modificar la definición de una tabla:

```
alter table tabla_base
```

```
  { add [column] definición_columna
```

```
    | alter [column] columna
```

```
      { set default { literal | función_sistema | null } 
```

```
        | drop default }
```

```
    | drop [column] columna { restrict | cascade }  };
```

En ORACLE cambian
algunas cosas

Para eliminar una tabla del esquema relacional:

```
drop table tabla_base { restrict | cascade };
```

En ORACLE es CASCADE
CONSTRAINTS

Definición de Vistas (SQL).

Las vistas son los objetos que proporciona el lenguaje SQL para definir esquemas externos.

- Una vista es una tabla virtual (no tiene una correspondencia a nivel físico)
- Se puede manejar como una tabla básica.
- Una vista se define en base a otras tablas (básicas o virtuales).
- Las actualizaciones se transfieren a la/s tabla/s original/es (con ciertas limitaciones).

```
CREATE | REPLACE VIEW vista [(columna)]  
AS expresión_tabla [with check option]
```

Definición de Vistas (SQL). Ejemplo.

```
CREATE VIEW PROVEEDORES_VAL  
AS SELECT * FROM PROVEEDOR  
WHERE ciudad = 'Valencia'
```

WITH CHECK OPTION;

El Check Option impide que yo pueda añadir proveedores que no sean de Valencia

Eliminación de una Vista

```
DROP VIEW vista [restrict | cascade];
```

OJO EN EL SENTIDO
DE CASCADE!

Vistas Actualizables.

Motivos por los que una vista no es actualizable:

- contiene operadores conjuntistas (UNION, INTERSECT,...)
- el operador DISTINCT
- funciones agregadas (SUM, AVG, ..)
- la cláusula GROUP BY

Vistas Actualizables.

Vista sobre una tabla básica:

- el sistema traducirá la actualización sobre la vista en una operación de actualización sobre la relación básica
- siempre que no se viole ninguna restricción de integridad definida sobre dicha relación

Vistas Actualizables.

Vista sobre una concatenación de relaciones:

- la actualización sólo puede modificar una de las tablas básicas
- actualización modificará la relación básica que cumpla la propiedad de *conservación de la clave* (aquella relación tal que su clave primaria podría ser también clave de la vista)
- la actualización no se realizará si viola alguna de las restricciones definidas sobre la relación básica que se va a actualizar

Definición de Restricciones (SQL)

```
create assertion restricción  
check (expresión_condicional)  
[comprobación_restricción];
```

La condición debe ser CIERTA.

Ejemplo: Proveedor-Piezas-Suministro (SQL)

La restricción R2 :

R2) Px: Pieza, Sx: Suministro $\forall Px (\text{Pieza}(Px) \rightarrow \exists Sx (\text{Suministro}(Sx) \wedge Sx.\text{código}=Px.\text{código}))$

se define mediante una restricción general:

```
create assertion R2 check
not exists(select * from Pieza P
           where not exists(select *
                           from Suministro S
                           where P.código=S.código));
```

Eliminación de una Restricción

```
DROP ASSERTION restricción
```

Definición de Privilegios (SQL).

Para realizar cualquier operación sobre un objeto (tabla o vista) se debe tener el privilegio necesario. Las operaciones con privilegios son:

- update (especificando qué columnas)
- insert (especificando qué columnas)
- delete
- select
- create view: es necesario tener el privilegio sobre la expresión de tabla de la definición de la vista (el SELECT de la vista).

Definición de Privilegios (SQL).

```
grant {all | select | insert[(comalista_ columna)] |  
      delete | update[(comalista_ columna)]}  
on objeto to {comalista_ usuarios | public }  
[with grant option]
```

En ORACLE cambian
algunas cosas

Eliminación de un Privilegio

```
revoke [grant option for]  
      {all | select | insert[(comalista_ columna)] |  
      delete | update[(comalista_ columna)]}  
on objeto to {comalista_ usuarios | public }  
      {restrict | cascade }
```