Unitat 5: SGBD. El Llenguatge de definició de dades. Edició de dades.

Bases de Dades

Profesora: Marina Fornés Giner

SQL

- SQL és un llenguatge que ens permet interactuar amb els SGBD Relacionals per especificar les operacions que volem fer sobre les dades i la seva estructura.
- SQL són les sigles de Structured Query Language (Llenguatge de Consulta Estructurat).
- És un llenguatge declaratiu, la qual cosa vol dir que s'hi especifica al Sistema Gestor de Base de Dades què volem obtenir i no la manera de com aconseguir-ho.
- És un llenguatge no procedimental perquè no necessitem especificar el procediment per aconseguir lobjectiu, sinó lobjectiu en si. No és un llenguatge de programació com ara Java o C.

SQL

- Ja saps que a través de SQL interactuarem amb un SGBDR que alhora gestiona una o més bases de dades relacionals. Per tant, el primer que hem de fer per començar a aprendre i practicar SQL és disposar d'un SGBDR al nostre ordinador de pràctiques. Per a les pràctiques utilitzarem com a SGBD **PostgreSQL.**

Components del llenguatge SQL

- El llenguatge SQL es compon d'enunciats. Aquestes frases es poden classificar en grups:
- Declaracions DDL (Data Definition Language): S'utilitzen per crear, modificar i eliminar elements estructurals en DBMS, com ara:
- bases de dades
- taules
- índexs
- restriccions, etc.
- Les definicions d'aquests objectes s'emmagatzemen al diccionari de dades del sistema.

Components del llenguatge SQL

- **Declaracions DML (Data Manipulation Language): Ens permeten indicar al sistema les operacions que volem realitzar amb les dades emmagatzemades en les estructures creades mitjançant les declaracions DDL. Per exemple, aquestes són les frases que permetran:**
- **generar consultes**
- **ordre**
- **filtre**
- **afegir**
- **modificar**
- **suprimir**
- **etc.**

Disseny de Bases de Dades

- Ja sabeu que els passos que se segueixen per dissenyar una base de dades relacional són:
- **Estudio una situació del món real** que es pretén modelar. Disseny d'un model conceptual de la situació.
- **Diagrama entitat-relació**.
- **Passar del disseny conceptual al disseny lògic. Model relacional.**
- **Implementació del model relacional en el DBMS a utilitzar** (SQL).
-

Crear una Base de Dades

Aquesta és la sintaxi de la declaració per crear bases de dades en SQL, `CREATE DATABASE`:

CREATE DATABASE <nombre_bd>

Per suprimir una base de dades utilitzeu la declaració `DROP DATABASE`:

DROP DATABASE <nombre_bd>

Tipus de dades

	TIPUS DE DADES	DESCRIPCIÓ	BYTES
CARÀCTER	CHAR	Un caràcter.	1 byte
	CHAR(n)	Cadena fixa de n caràcters.	(4+n) bytes
	VARCHAR(*n*)	Cadena de caràcters de llargària variable, amb un màxim de *n* caràcters. S'ha d'especificar la llargària.	(4+x) bytes
	TEXT	Igual que l'anterior, però no s'ha d'especificar la llargària màxima.	(4+x) bytes
NUMÈRIC	DECIMAL(*n*,*d*)	Número amb una precisió de *n* xifres, amb *d* decimals. Si no es posa *d* no hi ha decimals. La precisió màxima és de 1000 xifres.	variable
	NUMERIC(*n*,*d*)	Igual que l'anterior	
	FLOAT4	Coma flotant de simple precisió (6 xifres decimals)	4 bytes
	FLOAT8	Coma flotant doble precisió (15 xifres decimals)	8 bytes
	INT2	Enter (-32.768,32767)	2 bytes
	INT4	Enter (-2.147.483.648, 2.147.483.647)	4 bytes
	INT8	Enter amb unes 18 xifres	8 bytes
	SERIAL	Autonumèric (internament es crea una seqüència)	4 bytes

Tipus de dades

DATA	DATE	Tipus data. Valor mínim 1-1-4713 AC. Valor màxim 31-12-5874897 DC.	4 bytes
	TIME	Tipus hora. Guarda fins a la micra de segon	8 bytes
	TIMESTAMP	Tipus data-hora (combinant les característiques dels dos anteriors)	8 bytes
	INTERVAL	Un interval de temps (amb precisió d'un microsegon, però que pot arribar als 178.000.000 anys)	12 bytes
	BOOL	Booleà, amb valors True i False	1 byte

GEOMÈTRIC	POINT	Un punt de l'espai bidimensional (x,y) (dos float8)	16 bytes
	LSEG	Segment de línia definit per 2 punts (x1,y1) (x2,y2)	32 bytes
	BOX	Rectangle, definit pels extrems (x1,y1) (x2,y2)	32 bytes
	PATH	Conjunt de punts que representen una figura oberta o tancada: (x1,y1), ..., (xn,yn)	16+16n bytes
	POLYGON	Conjunt de punts que representen un figura tancada (x1,y1), ..., (xn,yn) (similar al path tancat)	40+16n bytes
	CIRCLE	Cercle representat pel centre i el radi	24 bytes
	INET	Adreça IP, de 4 números separats per punts, amb número de bits de la màscara separat per /	variable

Conversions

```
INSERT INTO taula VALUES (To_Number('18.901.234','99g999g990'));
```

Com podem veure en l'exemple, per a poder jugar amb els formats ens recolzarem en unes funcions de conversió:

- **TO_CHAR(*data, format*)** converteix una data en una tira de caràcters, utilitzant el format especificat.
- **TO_CHAR(*número, format*)** converteix un número en una tira de caràcters.
- **TO_NUMBER(*exp., format*)** converteix una tira de caràcters en un número, suposant que estava en el format indicat.
- **TO_DATE(*exp., format*)** converteix una tira de caràcters en un data.
-
- **TO_DATETIME(*exp., format*)** converteix una tira de caràcters en un data-hora.

Creació de Taules

```
CREATE TABLE nombre_tabla(

nombre_columna1 TIPO_COLUMNA1 (tamaño_columna1),
nombre_columna2 TIPO_COLUMNA2 (tamaño_columna2) NOT NULL ,
nombre_columna3 TIPO_COLUMNA3 (tamaño_columna3),

CONSTRAINT PK_nomtabla PRIMARY KEY (columnas),
CONSTRAINT UK_nomtabla UNIQUE (columnas),
CONSTRAINT FK_nomtabla  FOREIGN KEY (columnas)
        REFERENCES  nomtabla (columnas)
)

CREATE  TABLE  USUARIOS (
alias varchar(15),
Email  varchar(20),
password varchar(8),
CONSTRAINT  FK_usuarios  PRIMARY KEY (alias))
```

```
CREATE TABLE ALUMNOS (
DNI VARCHAR2(10) NOT NULL,
NOMBRE VARCHAR2(60),
EDAD NUMBER,
CONSTRAINT PK_ALUMNOS
PRIMARY KEY (DNI));
```

```
CREATE TABLE ASIGNATURAS
(CODIGO VARCHAR2(2) NOT
NULL,
NOMBRE VARCHAR2(60),
CONSTRAINT
PK_ASIGNATURAS PRIMARY
KEY (CODIGO));
```

```
CREATE TABLE MATRICULAR (DNI_ALU VARCHAR2(10) NOT NULL,
COD_ASIG VARCHAR2(2) NOT NULL, FECHA DATE,

CONSTRAINT PK_MATRICULAR PRIMARY KEY (DNI_ALU),

CONSTRAINT UK_CODASIG UNIQUE (COD_ASIG),

CONSTRAINT FK_MATRICULAR_ALUMNOS FOREIGN KEY
(DNI_ALU) REFERENCES ALUMNOS (DNI),

CONSTRAINT FK_MATRICULAR_ASIGNATURAS FOREIGN KEY
(COD_ASIG) REFERENCES ASIGNATURAS (CODIGO));
```

```
DROP TABLE MATRICULAR;

 ALTER TABLE ASIGNATURAS ADD (DNI_ALU VARCHAR(10)
NOT NULL, FECHA DATE);

ALTER TABLE ASIGNATURAS ADD CONSTRAINT
UK_ALUMNOS_DNI UNIQUE (DNI_ALU);

 ALTER TABLE ASIGNATURAS
ADD CONSTRAINT FK_ASIGNATURAS_ALUMNOS FOREIGN KEY
(DNI_ALU) REFERENCES ALUMNOS (DNI);
```

Modificació de Taules

ALTER TABLE nombre_tabla
RENAME TO nuevo_nombre

ALTER TABLE nombre_tabla
ADD COLUM nombre_columna tipo_columna
(tamaño_columna)

ALTER TABLE nombre_tabla
DROP COLUMN nombre_columna

ALTER TABLE nombre_tabla
ADD CONSTRAINT PK_nomtabla PRIMARY KEY
(columnas)

ALTER TABLE. Restriccions

La integritat referencial és una eina essencial de les bases de dades relacionals. Però això provoca diversos problemes. Per exemple, si eliminem un registre de la taula principal que està relacionat amb un o més dels secundaris, es produirà un error, ja que si se'ns permet eliminar el registre, es produirà un error d'integritat (hi haurà claus secundàries referides a una clau primària que ja no existeix). • És per això que Oracle ens ofereix dues solucions a afegir després de la clàusula REFERENCES:

• ON DELETE SET NULL. Anul·la totes les tecles secundàries relacionades amb la supressió.

• ON DELETE CASCADE. Suprimeix tots els registres la clau secundària dels quals sigui la mateixa que la clau de registre suprimida.

Eliminació de Taules

DROP TABLE nombre_tabla

Check

Una restricció de verificació es el tipus de restricció més genéric. Permet especificar que el valor en una columna determinada ha de satisfer una expresió. Per exemple, per a exigir preus de productes en positiu, podem utilitzar:

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CHECK (price > 0)
);
```

Check

```
CREATE TABLE employees (
 id serial PRIMARY KEY,
 first_name VARCHAR (50),
 last_name VARCHAR (50),
 birth_date DATE CHECK (birth_date > '1900-01-01'),
 joined_date DATE CHECK (joined_date > birth_date),
 salary numeric CHECK(salary > 0)
);
```

```
CREATE TABLE products (
    pk_product integer,
    name       text,
    price      numeric,
    discounted numeric,
    CONSTRAINT cn_price CHECK (price > 0 AND
                               price > discounted)
);
```

Check

```
ALTER TABLE EMPLEADOS ADD CONSTRAINT check_edad
  CHECK (EDAD BETWEEN 10 and 99)
```

O que queremos que sea una edad de 20, 25 o 30 o ninguna.

```
ALTER TABLE EMPLEADOS add CONSTRAINT check_edad2
  CHECK (EDAD IN (20,25,30) OR EDAD IS NULL);
```

Si es alfanumérico debe ir entre comillas simples:

```
ALTER TABLE EMPLEADOS add CONSTRAINT tipo_empleado
  CHECK (TIPO IN ('INGENIERO', 'ADMINISTRATIVO', 'DIRECTIVO') OR EDAD IS NULL);
```

Default

```
CREATE TABLE products (
    pk_product integer NOT NULL,
    name       text    NOT NULL,
    summary    text,
    price      numeric NOT NULL,
    discounted numeric DEFAULT 0 NOT NULL,
    company    integer NOT NULL
);
```

Ejemplos

```
CREATE TABLE ALUMNOS
(DNI VARCHAR2(10) NOT NULL,
 NOMBRE VARCHAR2(60),
 EDAD NUMBER,
 CONSTRAINT PK_ALUMNOS PRIMARY KEY (DNI));

 CREATE TABLE ASIGNATURAS
 (CODIGO VARCHAR2(2) NOT NULL,
  NOMBRE VARCHAR2(60),
  CONSTRAINT PK_ASIGNATURAS PRIMARY KEY (CODIGO));

 CREATE TABLE MATRICULAR
 (DNI_ALU VARCHAR2(10) NOT NULL,
  COD_ASIG VARCHAR2(2) NOT NULL,
  FECHA DATE,
  CONSTRAINT PK_MATRICULAR PRIMARY KEY (DNI_ALU),
  CONSTRAINT UK_CODASIG UNIQUE (COD_ASIG),
  CONSTRAINT FK_MATRICULAR_ALUMNOS FOREIGN KEY (DNI_ALU) REFERENCES ALUMNOS (DNI),
  CONSTRAINT FK_MATRICULAR_ASIGNATURAS FOREIGN KEY (COD_ASIG) REFERENCES
  ASIGNATURAS (CODIGO));
```