

# SQL: DDL

Francisco Moreno

# SQL: DDL

- DDL: Lenguaje de Definición de Datos
- Permite crear objetos en la BD

Tipos de objetos:

- **Tablas**: corresponden a las relaciones del modelo relacional
- **Índices**: estructuras que pueden ayudar a mejorar el desempeño de algunas consultas
- **Vistas**: son “tablas virtuales”\* → Recordar nivel externo
- Otros...

\* Aunque también existen las **vistas materializadas**

Una tabla se crea así:

```
CREATE TABLE nombre_tabla  
( atributos: cada uno con su  
  • tipo de datos y  
  • restricciones: de no nulidad, de clave  
    primaria, de clave foránea, de clave  
    alternativa, etc.  
);
```

Un índice se crea así:

```
CREATE INDEX nombreindice ON tabla(columna(s));
```

Una vista se crea así:

Ejemplo:

```
CREATE VIEW nombre_vista AS consulta SQL;
```



Se ven luego

# Restricciones de integridad

- Garantizan que los cambios hechos a la BD no ocasionen inconsistencias (según las reglas del negocio) en los datos.

## 1.1. Restricciones de dominio (tipo de datos): Conjunto de valores y de operaciones permitidas sobre ellos.

### Dominios esenciales de SQL:

- `CHAR(p)`: Cadena de caracteres de longitud fija `p`. Máxima longitud `p`.
- `VARCHAR(p)*`: Cadena de caracteres de longitud variable. Máxima longitud `p`.
- `NUMBER(p, s)`: Valor numérico de precisión `p` y escala `s` (para el manejo de decimales)
- `DATE`: Fechas.
- El programador puede definir sus propios tipos de datos → Objeto-relacional → Curso Bases de Datos 2.

\* También `CHARACTER VARYING` y en Oracle `VARCHAR2`.

# Restricciones de integridad

- 1.2. Nulos: Un atributo puede o no admitir nulos. En SQL se especifica mediante la cláusula `NOT NULL`.
- 1.3. Integridad referencial: Claves foráneas (CF). En SQL se especifica mediante las cláusulas `REFERENCES` y `FOREIGN KEY`
- 1.4. Clave Primaria (CP): Garantiza la **unicidad** y **obligatoriedad** del o de los atributos que conforman la CP. En SQL se especifica mediante la cláusula `PRIMARY KEY`.

# Restricciones de integridad

**1.5.** Clave Alternativa: Garantiza la **unicidad** de los atributos declarados como tal. Se especifica mediante la cláusula **UNIQUE**.

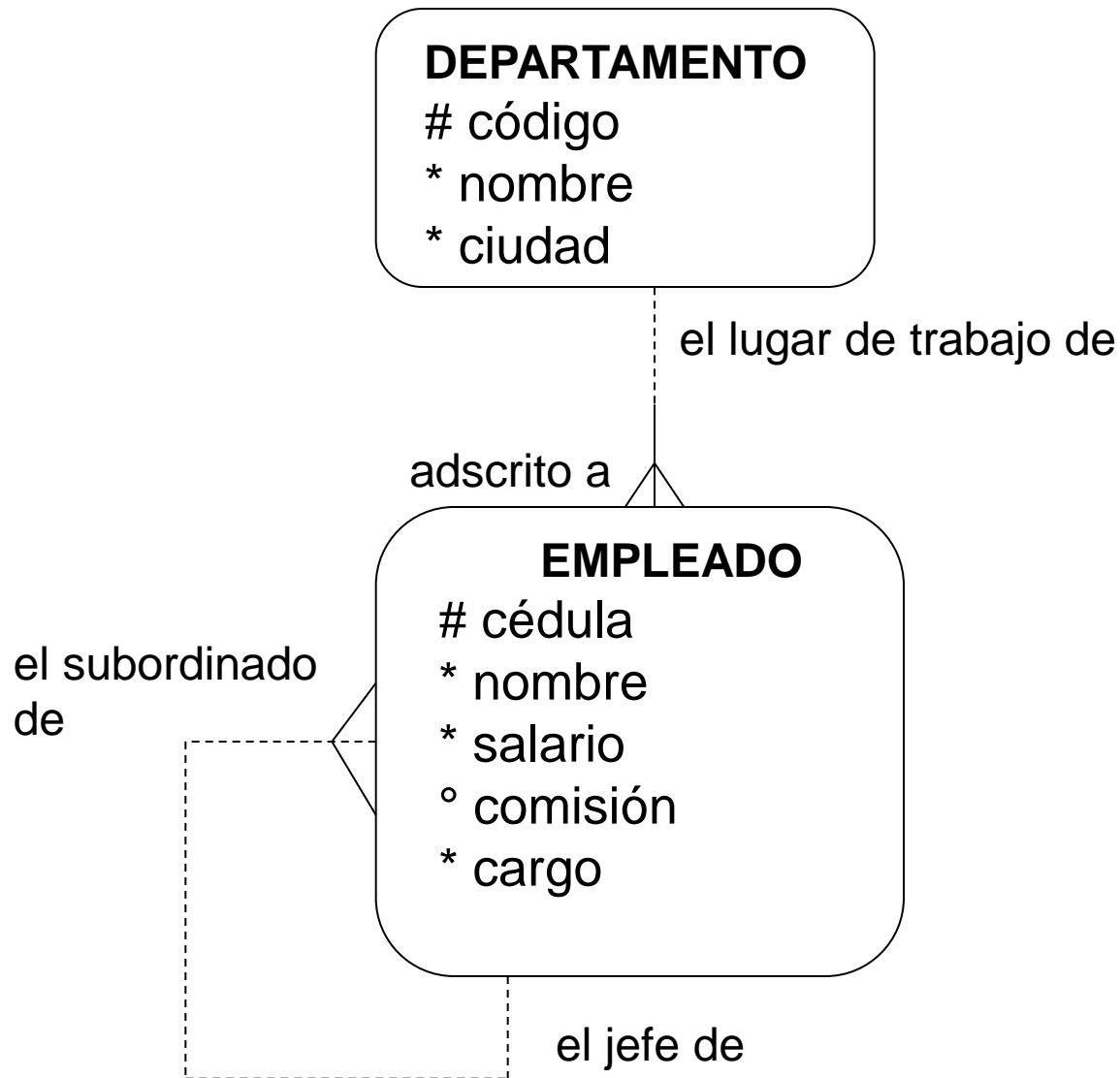
Si se desea hacer obligatoria, se debe especificar **NOT NULL**.

**1.6** Las reglas de tipo **CHECK**:

Condiciones de verificación para los valores de uno o varios atributos.

Sintaxis: **CHECK**(condición). Por ejemplo, validar que un valor sea positivo o esté en un rango.


## Ejemplo. Sea el modelo E-R:



**Notas:** en  
DEPARTAMENTO

- **nombre** es un id. alternativo.
- **ciudad** debe ser Medellín, Bogotá o Cali.

```
CREATE TABLE departamento
( codigo NUMBER(6) PRIMARY KEY,
  nombre VARCHAR(6) NOT NULL UNIQUE,
  ciudad VARCHAR(12) CHECK (ciudad IN
    ('Medellín', 'Bogotá', 'Cali')) NOT NULL
);
```



El atributo ciudad solo admitirá una de estas tres ciudades.

Si no se coloca, permite ciudades con NULL.



```
CREATE TABLE empleado
( cedula NUMBER(10) PRIMARY KEY,
  nombre VARCHAR(30) NOT NULL,
  salario NUMBER(10, 2) NOT NULL,
  comision NUMBER(2) ,
  cargo VARCHAR(20) NOT NULL,
  jefe NUMBER(10) REFERENCES empleado,
  depto NUMBER(6) NOT NULL REFERENCES
  departamento
);
```

CF hacia sí misma

An arrow originates from the 'jefe' column in the SQL statement and points diagonally upwards and to the right towards the 'CF hacia sí misma' box.

CF hacia departamento

An arrow originates from the 'depto' column in the SQL statement and points diagonally downwards and to the right towards the 'CF hacia departamento' box.

Se pueden especificar las políticas de integridad referencial ante borrado y actualización así:

```
CREATE TABLE empleado
( cedula NUMBER(10) PRIMARY KEY,
  nombre VARCHAR(30) NOT NULL,
  salario NUMBER(10,2) NOT NULL,
  comision NUMBER(2) ,
  jefe NUMBER(10) REFERENCES empleado,
  cargo VARCHAR(20) NOT NULL,
  depto NUMBER(6) REFERENCES departamento
  ON UPDATE SET NULL,
  ON DELETE CASCADE
);
```

Oracle, por ejemplo, no soporta esta opción...

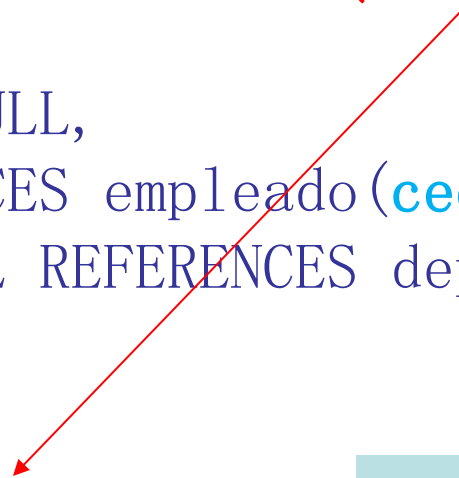
**Nota:** No todos los SGBD soportan estas opciones

```
CREATE TABLE empleado
(cedula NUMBER(10) PRIMARY KEY,
 nombre VARCHAR(30) NOT NULL,
 salario NUMBER(10,2) NOT NULL CHECK(salario > 0),
 comision NUMBER(3) NOT NULL CHECK (comision BETWEEN
 0 AND 100),
 cargo VARCHAR(20) NOT NULL,
 jefe NUMBER(10) REFERENCES empleado(cedula),
 depto NUMBER(6) NOT NULL REFERENCES departamento
);
```

El salario debe ser mayor que cero.



La comisión debe estar entre 0 y 100.



Explícitamente se puede especificar el atributo hacia el cual se refiere la CF




## Especificación de una CP compuesta:

```
CREATE TABLE envio(  
    snro NUMBER(6),  
    pnro NUMBER(6),  
    cantidad NUMBER(6) NOT NULL,  
    PRIMARY KEY (snro, pnro)  
);
```

**Nota:** Es incorrecto colocar PRIMARY KEY al frente de snro y de pnro.

## Especificación de una CF compuesta:

```
CREATE TABLE revision(  
  codrevision NUMBER(5) PRIMARY KEY,  
  mi_snro NUMBER(6) NOT NULL,  
  mi_pnro NUMBER(6) NOT NULL,  
  revisor VARCHAR(20) NOT NULL,  
  FOREIGN KEY(mi_snro, mi_pnro) REFERENCES envio  
);
```



**Cuando la CF es compuesta  
se debe usar esta sintaxis**

– Se puede modificar el esquema de una tabla con la instrucción `ALTER TABLE`

Ej: `ALTER TABLE revision ADD fecharev  
DATE;`

– Para destruir una tabla (esquema y datos)

`DROP TABLE revision;`

## Para ingresar filas:

- `INSERT INTO envio VALUES (10, 20, 100) ;`
- `INSERT INTO departamento VALUES (100, 'Aseo', 'Cali') ;`

Sea la tabla:

```
CREATE TABLE t (  
  a NUMBER(3) PRIMARY KEY,  
  b NUMBER (3),  
  c VARCHAR(3) NOT NULL);
```

```
INSERT INTO t VALUES (10, 20, 'hi');
```

```
INSERT INTO t(a, b, c) VALUES (11, 22, 'si');
```

```
INSERT INTO t(c, a) VALUES ('bye', 20);
```

¿Qué pasa con el atributo **b** en este último caso?



- Para eliminar filas de una tabla:

DELETE

FROM tabla

[WHERE condicion];

Si no se coloca la cláusula WHERE, se eliminan todas las filas de la tabla

Ej: DELETE

FROM t

WHERE a = 11;

Se ven luego

La condición puede incluir subconsultas...

- Para actualizar filas de una tabla:

```
UPDATE tabla  
SET atributo = nuevo_valor →  
[WHERE condicion];
```

Se pueden actualizar varios atributos, se separan por comas

- La condición y nuevo\_valor pueden incluir subconsultas...

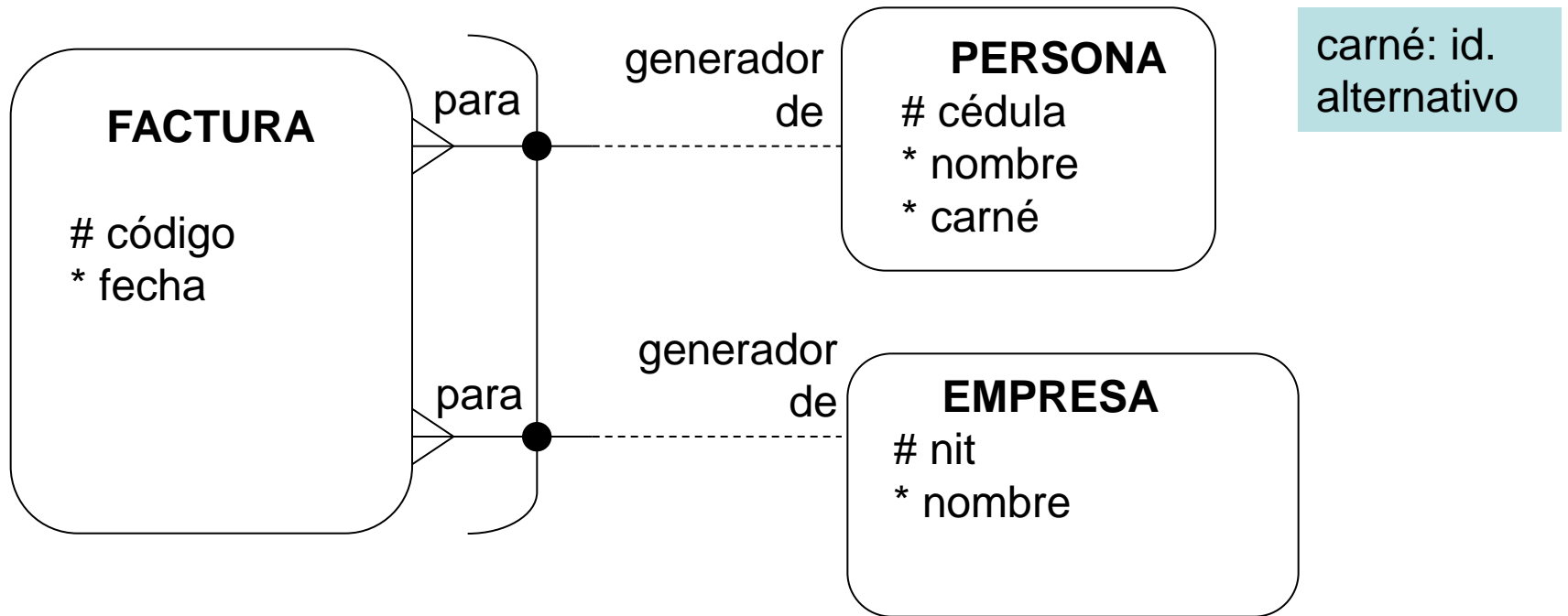
Ejemplo:

```
UPDATE t
```

```
SET b = 90, c = 'yes'
```

```
WHERE a = 20;
```

# Implementación de arcos




```
CREATE TABLE persona(  
cedula NUMBER(8) PRIMARY KEY,  
nombre VARCHAR(25) NOT NULL,  
carne NUMBER(5) UNIQUE NOT NULL);
```

```
INSERT INTO persona VALUES(10, 'Dino', 912);
```

```
CREATE TABLE empresa(  
nit NUMBER(10) PRIMARY KEY,  
nombre VARCHAR(20) NOT NULL);
```

```
INSERT INTO empresa VALUES(40, 'BDW');
```

```
CREATE TABLE factura(  
codigo NUMBER(6) PRIMARY KEY,  
fecha DATE NOT NULL,  
cedula NUMBER(8) REFERENCES persona,  
nit NUMBER(10) REFERENCES empresa,  
CHECK ((nit IS NULL AND cedula IS NOT NULL)  
      OR  
      (nit IS NOT NULL AND cedula IS NULL))  
);
```



Por medio del CHECK se implementa el arco, este garantiza que si una CF **es nula**, la otra CF **es no nula**.

Genera la fecha  
actual (en Oracle)\*

- INSERT INTO factura VALUES (300, SYSDATE, 10, NULL) ;
- INSERT INTO factura VALUES (9, SYSDATE - 1, NULL, 40) ;

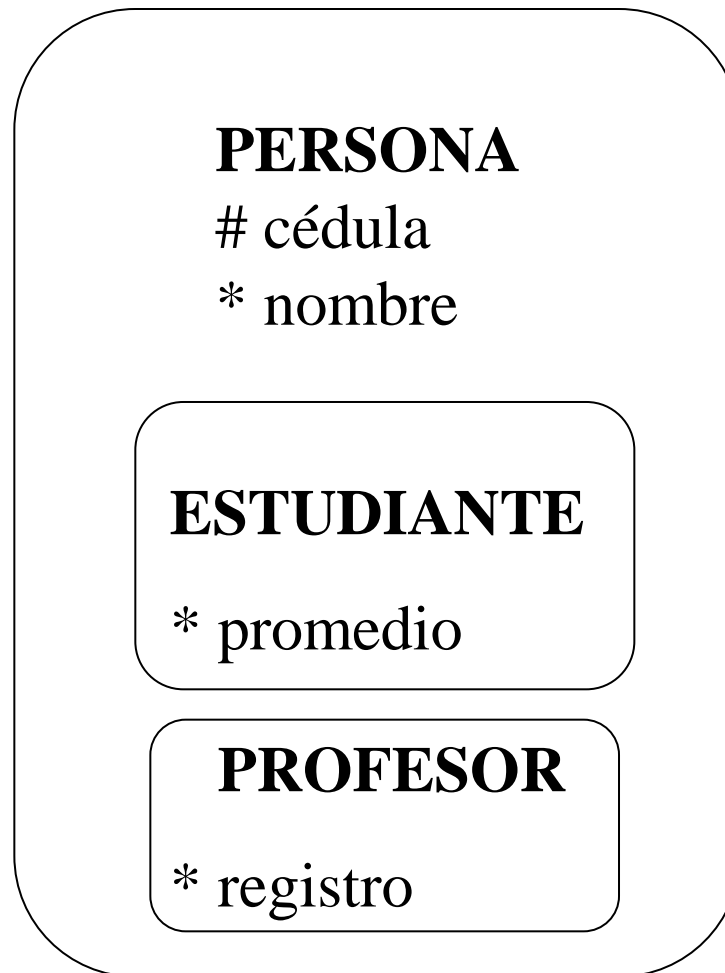
Se resta un día (ayer)

**Note que las dos siguientes inserciones fallan:**

- INSERT INTO factura VALUES (500, SYSDATE, 10, 40) ;
- INSERT INTO factura VALUES (600, TO\_DATE(' 10-jul-2014', ' dd-mon-yyyy' ), NULL, NULL) ;

\* En SQL estándar es `CURRENT_DATE` (en Oracle también funciona).

# Implementación del supertipo y sus subtipos



registro es id.  
alternativo  
para los  
profesores



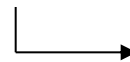
- Repasar la presentación 8: Conversión E-R a Relacional
- Veamos la alternativa dos para implementar supertipos y subtipos:

- Se crea la tabla para el supertipo:

```
CREATE TABLE persona(  
    cedula NUMBER(8) PRIMARY KEY,  
    nombre VARCHAR(20) NOT NULL,  
    tipo VARCHAR(1) NOT NULL CHECK (tipo =  
        'e' OR tipo = 'p')  
);
```

- Se crean tablas para cada uno de los subtipos:

```
CREATE TABLE estudiante(  
  cedest NUMBER(8) PRIMARY KEY REFERENCES  
  persona,  
  promedio NUMBER(3, 2) NOT NULL);
```



3 dígitos: 1 entero, 2 decimales

```
CREATE TABLE profesor(  
  cedprof NUMBER(8) PRIMARY KEY REFERENCES  
  persona,  
  registro NUMBER(5) UNIQUE NOT NULL);
```

- En esta alternativa se debe validar que la cédula de una persona **no exista en ambas tablas para garantizar la exclusividad**. Esto se puede lograr, por ejemplo, mediante un disparador (*trigger*)\*

\* Se ven en Bases de Datos 2.