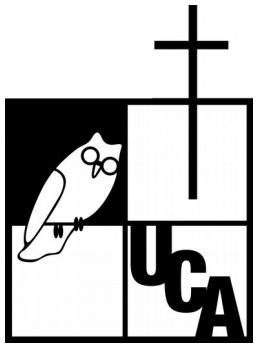
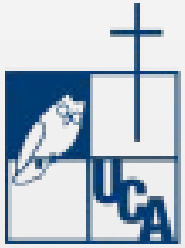


Structured Query Language

Data Manipulation Language

Administración de Bases de Datos



Departamento de
electrónica e informática

Carlos Juan Martín Pérez

INSERTA una o varias tuplas en una relación. Formato: **`INSERT INTO`** `<tabla>` **`VALUES`** `(a1, a2, ..., an);`

Lista de valores: En el mismo orden en el que fue creada la tabla con **`CREATE TABLE`**. id_p nombre_p Peso Cantidad

Ej.: `INSERT INTO pieza VALUES (4,'Teja',50,1000);`

Puede especificarse sólo un **subconjunto de atributos**. Formato: **`INSERT INTO`** `<t>(<lista_atrbs>)` **`VALUES`** `(...);`

Lista de atributos: Nombres de los atributos en los que se desea insertar algún valor. Los valores deben estar en ese orden.

Atributos ausentes: Se les asigna su valor por defecto o NULL.

Deben estar todos los atributos que no admiten NULL y que además no tienen valor por defecto (restr. **`NOT NULL`** y **`DEFAULT`**).

Insertar varias tuplas: Separadas por comas y entre paréntesis.

SGBD: Debe gestionar que se cumplan las restricciones (especialmente la de integridad referencial).

Cláusula SELECT:

- Pueden **renombrarse** los nombres de las columnas del resultado, poniendo el nuevo nombre justo después del atributo (usar comillas si son varias palabras).
- Puede ponerse un **`*`** para indicar que se recuperen **todos los atributos** de todas las tablas de la cláusula `FROM`.
- Puede usarse también el formato **`<tabla>.*`** para referirse a **todos** los atributos de esa tabla.
- Pueden seleccionarse **tuplas repetidas** (con todos sus atributos iguales), ya que SQL no establece esa condición (no son conjuntos de tuplas sino multi-conjuntos). Pueden eliminarse tuplas repetidas usando: **`SELECT DISTINCT`**.

Cláusula FROM: Pueden ponerse alias a las tablas: `<tabla> [AS] <alias>`.
También pueden renombrarse todos los atributos de las tablas después de establecer el alias: `<tabla> [AS] <alias(nuevos nombres de los atributos)>`.

Cláusula WHERE: La condición puede usar los op. lógicos **`NOT`**, **`AND`** y **`OR`**.

- **Si no existe cláusula `WHERE`:** Se supone que se recuperan todas las tuplas (como si la condición fuese siempre verdad).
  - Si no existe cláusula `WHERE` y en la clásula `FROM` hay varias tablas, se obtiene el **Producto Cartesiano:** Por tanto, si hay varias tablas la condición establece una selección de tuplas del Producto Cartesiano.
  - Las condiciones de una **operación de COMPOSICIÓN** hay que explicitarlas (puede ser con un JOIN).

Structured Query Language

Data Manipulation Language

```
SELECT <lista_atributos>
   FROM <lista_tablas>
   WHERE <condición>;
```

Esquema:
Suministrador (id_s, nombre_s, Dirección, Ciudad);
Pieza (id_p, nombre_p, Peso, Cantidad);
Suministro (suministrador_id_s,pieza_id_p);

"Nombres de Piezas que pesen más de 15 gramos y que su Cantidad sea menor o igual que 30":

```
SELECT nombre_p FROM Pieza
WHERE Peso>15 AND Cantidad<=30;
```

"Piezas de las que se ignore el Peso":

```
SELECT * FROM Pieza
WHERE Peso IS NULL;
```

No usar Peso=NULL.

"Números de Suministradores que Suministren una Pieza de 15 gramos":

```
SELECT suministrador_id_s FROM Pieza, Suministro
WHERE Pieza.id_p = Suministro.pieza_id_p AND Peso=15;
```

"Nombres de Suministradores que Suministren una Pieza de 15 gramos":

```
SELECT Nombre_s FROM Suministrador, Pieza, Suministro
WHERE Suministro.id_p = Pieza.id_p AND Peso=15
  AND Suministro.suministrador_id_s = Suministrador.id_s;
```

"Nombres de Piezas suministradas desde Soyapango": Usando **alias** de tablas.

```
SELECT nombre_p FROM Suministrador S, Pieza P, Suministro SP
WHERE SP.pieza_id_p = P.id_p AND
      SP.suministrador_id_s = S.id_s AND Ciudad='Soyapango';
```

Tuplas repetidas: Las tablas en SQL no son conjuntos de tuplas, pues admiten duplicados (son multiconjuntos). En las consultas:

Los duplicados pueden interesar al usuario, pero si no, se usa DISTINCT.

"Nombres de Piezas diferentes suministradas desde Soyapango":

```
SELECT DISTINCT nombre_p FROM Suministrador S, Pieza P,
Suministro SP WHERE SP.pieza_id_p = P.id_p AND
SP.suministrador_id_s = S.id_s AND Ciudad='Soyapango';
```

Operaciones de conjuntos: Pueden usarse entre varias subconsultas, actuando sobre los conjuntos de tuplas de cada una:

UNION: Unión SIN duplicados de las tuplas de las consultas.

INTERSECT: Intersección de conjuntos

EXCEPT: Diferencia (resta) de conjuntos.

UNION ALL: Unión CON duplicados.

Ejemplo: "Números de Suministradores que NO suministren la Pieza 8":

```
SELECT id_s FROM Suministrador
EXCEPT
SELECT suministrador_id_s FROM Suministro
WHERE pieza_id_p='p008';
```

SUM, MAX, MIN, AVG, STDEV, VARIANCE: Calcula la **suma** de los valores del grupo, el valor **mayor**, el valor **menor**, la **media aritmética** (*average*), la **desviación típica** (*standard deviation*) y la **varianza** (el cuadrado de la desviación típica).

```
SELECT
      SUM(Salario), MAX(Salario), MIN(Salario),
      AVG(Salario), VARIANCE(Salario)
      STDDEV(Salario), '=',
  SQRT(VARIANCE(Salario)) FROM Empleado;

SELECT AVG(DISTINCT Salario) "Media"
FROM Empleado
  WHERE Dpto NOT BETWEEN 5 AND 9;

SELECT SUM(Salario), AVG(Salario)
  FROM Empleado E, Dpto D
  WHERE E.Dpto=D.NumDpto AND D.Nombre='I+D';
```

Observación: `DISTINCT` y `ALL` no tienen efecto en las funciones **`MAX`** y **`MIN`**.

Funciones de Grupo o de Agregación: Son funciones que se aplican sobre un grupo de valores del mismo dominio. Se aplican sobre un grupo de tuplas (o de atributos concretos de esas tuplas).

COUNT: Cuenta el número de tuplas del grupo (indicadas por *****).

"¿Cuántas piezas hay que pesen más de 15 gramos?":

```
SELECT COUNT(*) FROM Pieza WHERE Peso>15;
```

"¿Cuántos empleados hay en el Departamento de I+D?":

```
SELECT COUNT(*) FROM Empleado E, Dpto D
WHERE E.Dpto=D.NumDpto AND D.Nombre='I+D';
```

"¿Cuántos salarios **distintos** hay entre los empleados?":

```
SELECT COUNT(DISTINCT Salario) FROM Empleado;
```

"¿Cuántas piezas hay de las que se ignore su peso?":

```
SELECT COUNT(*) FROM Pieza WHERE Peso IS NULL;
```

Observaciones:

Los NULL se ignoran, excepto por **COUNT(*).**

DISTINCT elimina los valores duplicados (y no cuenta los NULL).

En el lugar de **DISTINCT** se puede usar **ALL** (opción por defecto).

DISTINCT y **ALL** pueden usarse en todas las funciones de grupo.

Comparación de cadenas: Puede usarse el comparador **[NOT] LIKE** con los caracteres **"%"** (para cualquier número de caracteres) y **"_"** (para un único carácter). "_" y "\%" son sendos símbolos. El operador "||" concatena cadenas.

"Seleccionar Empleados que tengan un apellido López":

```
SELECT * FROM Empleado WHERE Nombre LIKE '%López%';
```

"Seleccionar Valores que hayan bajado un porcentaje acabado en 5":

```
SELECT * FROM Valores WHERE Variacion LIKE '-_5\%';
```

Operaciones Aritméticas (+,-,*,/):

"Productos con su porcentaje de descuento posible aplicado si el descuento aplicado no supera los 5 Dólares":

```
SELECT Precio, Descuento as Porcentaje_Descuento,
 Precio-(Precio*Descuento/100) as Precio_Final
FROM Producto
WHERE (Precio*Descuento)/100 < 5;
```

Ordenar Resultados: Cláusula **ORDER BY** seguida de una lista de atributos o de posiciones en la lista de atributos del **SELECT**. Tras cada atributo puede ponerse **ASC** (ordenación ascendente, por defecto) o **DESC** (ordenación descendente):

"Seleccionar Empleados y sus salarios aumentados un 10% si su salario inicial está entre 1 y 3 mil dólares, ordenando descendentemente según su sueldo incrementado":

```
SELECT Nombre, 1.1*Salario as salario_aum
FROM Empleado
WHERE Salario BETWEEN 1000 AND 3000
ORDER BY salario_aum DESC;
```

Insertar el resultado de una consulta: Formato:

```
INSERT INTO <tabla> <subconsulta>;
```

Ej.: Suponemos que la tabla TOTAL está creada correctamente.

```
INSERT INTO TOTAL
SELECT S.id_s, nombre_s, COUNT(*)
       FROM Suministro SP, Suministrador S
       WHERE SP.suministrador_id_s=S.id_s;
```

Otra opción es usar **CREATE TABLE AS SELECT** …

Observaciones:

- La tabla TOTAL es una tabla normal: Puede consultarse, modificarse y borrarse como cualquier otra tabla.
- La tabla TOTAL no cambiará si se producen cambios en las tablas de las que se ha extraído su información. Sus datos pertenecerán a la fecha en la que se ejecutó la sentencia **INSERT**.
- Si deseamos que la tabla esté actualizada, crear una **VISTA**.

BORRA una o varias tuplas en una relación. Formato:

```
DELETE [FROM] <tabla> [<alias>] [WHERE <cond>];
```

Borra las tuplas que cumplan la condición**.**

Puede implicar el borrado de otras tuplas en otras tablas, para que se cumplan la restricción de integridad referencial.

Si se omite la cláusula `WHERE` (y su condición), se borran todas las tuplas de la tabla: La tabla quedará vacía (pero sigue existiendo).

Ejemplos:

```
DELETE Suministrador WHERE id_s IN (2,4,8);

DELETE Suministro SP
  WHERE suministrador_id_s IN (
     SELECT id_s FROM Suministrador
      WHERE Ciudad='Tegucigalpa');

DELETE Pieza
  WHERE Peso-250 > (SELECT AVG(Peso) FROM Pieza
                    WHERE Peso > (SELECT AVG(Peso)
                                  FROM Pieza) );
```

ACTUALIZA o **MODIFICA** tuplas de una relación. Formato:

```
UPDATE <tabla> [<alias>] SET <actualizaciones>
    [ WHERE <cond> ];
```

`<actualizaciones>` puede ser:

1. Una lista de asignaciones separadas por coma del tipo:
   `<atributo> = <expresión>`
   (Una expresión puede ser una subconsulta que recupere sólo un valor).
2. Una lista de atributos entre paréntesis a los que se le asigna una subconsulta: `(<A1>, ..., <Am>) = (<subconsulta>)`

La cláusula **WHERE** selecciona las tuplas a modificar.

Modificar la **llave primaria** puede acarrear la modificación de llaves externas en otras tablas.

```
UPDATE Pieza SET Nombre='Eje', Peso=9 WHERE id_p=5;
UPDATE Pieza SET Cantidad=Cantidad+100 WHERE id_p IN
  (SELECT pieza_id_p FROM Suministro WHERE id_s IN
  (3,7));
UPDATE Pieza SET (Peso,Cantidad)=(SELECT
  Peso,Cantidad FROM Pieza WHERE id_p=6) WHERE
  id_p=5;
```