

# Structured Query Language (SQL)



Fundamentos de Bases de  
Datos

InCo - 2011

# Un poco de historia...

---

- Lenguajes de consulta relacionales:
  - SEQUEL (IBM-1970) → SQL
  - QUEL (Ingres-1970)
  - QBE (IBM-1970)
  
- SQL es el lenguaje comercial más aceptado
  - Standard aprobado por ANSI e ISO en 1986.
  - Existen varios dialectos: SQL-86, SQL-89, SQL-92, SQL:1999, SQL:2003, SQL:2006

# Características de SQL Standard

---

- ❑ Lenguaje no procedural.
- ❑ Opera sobre conjuntos de tuplas:
  - Incluso para las operaciones de inserción, borrado y actualización.
- ❑ No elimina automáticamente tuplas repetidas.
- ❑ El mismo lenguaje puede ser usado:
  - En forma interactiva (EJ: consola)
  - Embebido en un lenguaje de programación.

# Características de SQL Standard (2)

---

- ❑ Su poder de expresión incluye al álgebra relacional y la extiende
- ❑ Se identifican dos sublenguajes:
  - DDL (Data Definition Language):
    - ❑ Permite crear, modificar y eliminar objetos de la base
    - ❑ EJ: CREATE, ALTER, DROP
  - DML (Data Manipulation Language):
    - ❑ Permite crear, modificar, eliminar y recuperar datos
    - ❑ EJ: INSERT, UPDATE, DELETE, SELECT

# SQL como Data Definition Language

---

- Permite crear, modificar y eliminar objetos de la base de datos
  - Tablas
  - Vistas
  - Usuarios
  - Etc.

# ¿Qué es una tabla?

---

- ▣ Un conjunto de valores organizados en columnas y filas
- ▣ Una tabla es la representación de una relación, pero no son estrictamente equivalentes

# Operaciones sobre tablas

---

## ❑ CREATE TABLE

- Crea una nueva tabla de la base
- Parámetros (entre otros):
  - ❑ Nombre de la tabla
  - ❑ Nombre de cada columna
  - ❑ Tipo de datos de cada columna
  - ❑ Restricciones de clave primaria y clave foránea sobre otras tablas

## ❑ ALTER TABLE

- Modifica una tabla existente

## ❑ DROP TABLE

- Elimina la definición de la tabla (y los datos almacenados en ella si existen)

# Ejemplo

---

**Productos** ( #prod, nombre, peso)

*Contiene todos los productos*

**Fabricantes** ( #fab, nombre, departamento)

*Contiene los fabricantes de productos*

**Ventas** ( #fab, #prod, precio)

*Indica que fabricante vende que producto y a  
que precio*



# Ejemplo

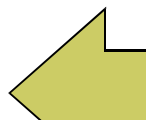
---

```
CREATE TABLE productos
( "#prod" integer NOT NULL,
  nombre character varying(50),
  CONSTRAINT productos_pk PRIMARY KEY
( "#prod" )
)
```



**CREO LA  
TABLA  
PRODUCTOS**

```
ALTER TABLE productos
add column peso double precision
```



**AGREGO LA  
COLUMNA  
PESO A LA  
TABLA  
PRODUCTOS**

# ¿Qué es una vista?

---

- Una vista es una tabla virtual que se basa en el resultado de una consulta.
- Sus atributos son atributos de tablas o de otras vistas.
- Pueden usarse en consultas como si fueran tablas

# Operaciones sobre vistas

---

- ❑ CREATE VIEW: crea una vista
- ❑ ALTER VIEW: modifica una vista
- ❑ DROP VIEW: elimina la vista

# Ejemplo

---

Quiero ofrecer una vista sobre la tabla productos  
en la que NO aparezca el peso

```
CREATE VIEW productos2 AS
```

```
( SELECT "#prod", nombre  
  FROM productos  
)
```

***Devuelve el código de  
producto y el nombre de  
cada producto***

# SQL como Data Manipulation Language

- Permite crear, modificar, eliminar y recuperar datos
- INSERT: agrega tuplas a una tabla
- UPDATE: cambia tuplas de una tabla
- DELETE: borra tuplas de una tabla
- SELECT: recupera datos

# Ejemplos

---

- ❑ Insertar una nueva venta del producto 530 a precio 15000, para el fabricante 25:

```
INSERT INTO VENTAS  
VALUES (25, 530, 15000);
```

# Ejemplos

---

- Incrementar el precio del producto número 530 en un 10%

**UPDATE** VENTAS

SET precio = precio \* 1.1

WHERE "#prod" = 530;

- Borrar toda la información de ventas con precio mayor a \$50000.

**DELETE** FROM VENTAS

WHERE precio > 50000;

# Recuperación de datos

---

## □ Select : devuelve tuplas de la base

SELECT A1 , . . . , An

FROM R1 , . . . , Rm

WHERE P

donde:

- A1,...,An son nombres de atributos. (puede usarse \*)
- R1,...,Rm son nombres de tablas.
- P es una condición.

## □ en Algebra Relacional :

$$\square \Pi_{A1,...,An} ( \sigma_P (R1 \times \dots \times Rm) )$$



# Ejemplos

---

- ▣ Precio del que vende el producto 7 el fabricante 2.

```
SELECT precio
```

```
FROM VENTAS
```

```
WHERE "#fab" = 2 and "#prod" = 7;
```

- ▣ Número de los fabricantes que vendieron el producto 200 a menos de \$100000.

```
SELECT "#fab" FROM VENTAS
```

```
WHERE precio < 100000 and "#prod" = 200;
```

# Orden de tuplas

---

- ❑ Dar los #prod de los productos fabricados, pero ordenados en forma ascendente.

```
SELECT "#prod"  
FROM VENTAS  
ORDER BY "#prod" asc;
```

- ❑ La cláusula ORDER BY permite indicar el campo por el cual se ordena el resultado.
- ❑ Se indica el orden (ASC o DESC)

# Filtrado de repetidos

---

- Dar los #prod de los productos vendidos, filtrando los repetidos.

```
SELECT DISTINCT    "#prod"  
FROM  VENTAS  
ORDER BY "#prod" ;
```

- La cláusula **DISTINCT** filtra tuplas repetidas.

# Join

---

- Dar los nombres de fabricantes y los #prod que venden.

```
SELECT nombre, "#prod"  
FROM VENTAS, FABRICANTES  
WHERE VENTAS."#fab" = FABRICANTES."#fab";
```

- Dar los nombres de fabricantes y los #prod que venden, tales que vendió el producto a \$100.

```
SELECT nombre, "#prod"  
FROM VENTAS, FABRICANTES  
WHERE VENTAS."#fab" = FABRICANTES."#fab" and  
precio = 100;
```

## Join (2)

---

- En lugar de imponer la igualdad puedo usar el operador JOIN

```
SELECT nombre, "#prod"  
FROM (VENTAS JOIN FABRICANTES ON  
      VENTAS."#fab"=FABRICANTES."#fab" )
```

- También existe el NATURAL JOIN (elimina columnas con nombres repetidos)

```
SELECT nombre, "#prod"  
FROM (VENTAS NATURAL JOIN FABRICANTES )
```

# Otros tipos de JOIN

---

- ❑ En el JOIN sólo se incluyen en el resultado tuplas que coincidan en valor en los campos de JOIN.
- ❑ LEFT JOIN: se agrega, para cada tupla de T1 que no satisface la condición de JOIN con NINGUNA de T2, una fila con NULOS en las columnas de T2
- ❑ RIGHT JOIN: análogo al LEFT pero se incluyen todos los de T2
- ❑ FULL JOIN: equivale a la unión del LEFT y RIGHT

# Ejemplo de JOINS

▣ T1

A	B
1	a
2	b
3	c

T2

A	C
1	xxx
3	yyy
5	zzz

**SELECT \* FROM t1 JOIN t2  
ON t1.A = t2.A;**



A	B	A	C
1	a	1	xxx
3	c	3	yyy

**SELECT \* FROM t1 NATURAL  
JOIN t2;**



A	B	C
1	a	xxx
3	c	yyy

# Ejemplo de JOINS (2)

□ **SELECT \* FROM**

t1 **LEFT JOIN** t2

**ON** t1.A = t2.A;



A	B	A	C
1	a	1	xxx
2	b	NULL	NULL
3	c	3	yyy

□ **SELECT \* FROM**

t1 **RIGHT JOIN** t2

**ON** t1.A = t2.A;



A	B	A	C
1	a	1	xxx
3	c	3	yyy
NULL	NULL	5	zzz

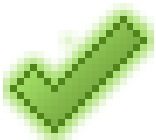


# Renombre de atributos

- Dar los nombres de fabricantes y los nombres de los productos que venden.



```
SELECT nombre, nombre  
FROM VENTAS, FABRICANTES, PRODUCTOS  
WHERE VENTAS."#fab" = FABRICANTES."#fab"  
      AND VENTAS."#prod" = PRODUCTOS."#prod";
```



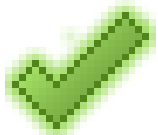
```
SELECT FABRICANTES.nombre as "nomFab",  
PRODUCTOS.nombre as "nomProd"  
FROM VENTAS, FABRICANTES, PRODUCTOS  
WHERE VENTAS."#fab" = FABRICANTES."#fab"  
      AND VENTAS."#prod" = PRODUCTOS."#prod";
```

# Alias

- Dar las parejas de nombres de productos con igual peso.



```
SELECT nombre, nombre  
FROM PRODUCTOS, PRODUCTOS  
WHERE PRODUCTOS."#prod" != PRODUCTOS."#prod"  
AND PRODUCTOS.peso = PRODUCTOS.peso;
```



```
SELECT p1.nombre, p2.nombre  
FROM PRODUCTOS p1, PRODUCTOS p2  
WHERE p1."#prod" != p2."#prod"  
AND p1.peso = p2.peso;
```

# Union

---

- ▣ Dar los #fab que son de Montevideo o que venden algún producto a precio mayor que \$500.

```
SELECT  "#fab"  FROM FABRICANTES
WHERE   departamento = 'Montevideo'

UNION
```

```
SELECT  "#fab"  FROM VENTAS
WHERE   precio > 500;
```

- ▣ La UNION elimina tuplas repetidas.

# Diferencia

- Dar los #fab que no venden ningún producto.

**EQUIVALENTES**

```
SELECT    "#fab" FROM    FABRICANTES  
EXCEPT  
SELECT    "#fab" FROM    VENTAS ;
```

```
SELECT    "#fab" FROM    FABRICANTES  
WHERE     "#fab"    not IN  
                ( SELECT    "#fab"  
                  FROM    VENTAS )
```

# Funciones de agregación

---

- ▣ Las funciones de agregación extienden al álgebra relacional (EJ: contar tuplas, sumar cantidades, etc.)
- ▣ Decir cuantos fabricantes hay.

```
SELECT count( "#fab" )  
FROM FABRICANTES;
```

- ▣ Dar el precio máximo de venta.

```
SELECT max(precio)  
FROM VENTAS;
```

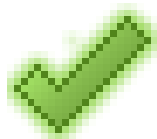
# Funciones de agregación (2)

---

- ❑ Las funciones de agregación aplican sobre conjuntos de tuplas, no sobre tuplas individuales.
- ❑ Ejemplo: devolver el fabricante que ha vendido el producto más caro.



```
SELECT "#fab" FROM VENTAS  
WHERE precio = max(precio);
```



```
SELECT "#fab" FROM VENTAS  
WHERE precio = (SELECT max(precio)  
FROM VENTAS);
```

# Funciones y Operadores aritméticos

---

- Expresiones en las cláusulas SELECT y WHERE
  - Se pueden utilizar expresiones aritméticas que combinen atributos (+, \*, /)
  - Se pueden aplicar funciones sobre atributos:
    - round (n), abs (n), mod (m, n), sign (n)

# Ejemplos sobre el SELECT

---

- ❑ Para cada producto dar su numero y su peso en gramos (supongamos que el peso esta kg.).

```
select "#prod", peso * 1.000  
from PRODUCTOS;
```

- ❑ Queremos que el resultado de peso \* 1.000 sea redondeado a 2 cifras decimales.

```
select "#prod",  
round (peso * 1.000, 2)  
from PRODUCTOS;
```



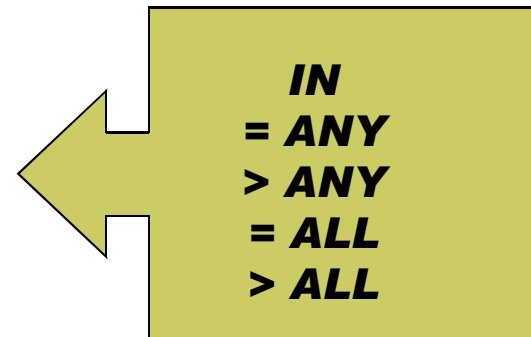
# Consultas anidadas

## □ ¿Qué son?:

- Consultas que se ubican dentro de la cláusula WHERE de otra consulta.

## □ Ejemplo

```
SELECT A1, ..., An
FROM R1, ..., Rm
WHERE (Aj, ..., Ak) <op-comp>
      (SELECT Bj, ..., Bk
       FROM S1, ..., Sn
       WHERE Q)
```



# Ejemplo

---

- Dar los nombres de los fabricantes que venden productos que también son vendidos por el fabricante número 10.

```
SELECT f.nombre  
FROM FABRICANTES f, VENTAS v  
WHERE f."#fab" = v."#fab" AND  
       f."#fab" <> 10 AND v."#prod" IN
```

```
       (SELECT "#prod"  
        FROM VENTAS  
        WHERE VENTAS."#fab" = 10)
```

**Productos  
vendidos por  
el fabricante 10**

# Función EXISTS

---

- ¿Para qué sirve?:
  - para chequear si el resultado de una consulta anidada es vacío.
- Ejemplo:
  - **Dar los nombres de los fabricantes que sólo venden el producto numero 200.**

```
SELECT nombre
FROM FABRICANTES F1, VENTAS V1
WHERE F1."#fab" = V1."#fab" AND
      V1."#prod" = 200 AND NOT EXISTS
```

***Vende el  
producto 200***

```
(SELECT *
FROM VENTAS V2
WHERE V2."#fab"=F1."#fab" AND
      V2."#prod" <> 200)
```

***Y no vende  
nada más***

# Agrupamiento de tuplas

---

- ❑ Dar, para cada fabricante, la cantidad de productos que vendió.
  - Esto implica:
    - ❑ Tomar cada grupo de tuplas en Ventas correspondiente a un fabricante.
    - ❑ Contar la cantidad de tuplas en el grupo.
  - Para realizar esta operación es necesaria la cláusula **GROUP BY**.

# GROUP BY

---

- Es una cláusula más que se agrega al SELECT, FROM, WHERE

- Ejemplo:

```
SELECT "#fab", count(*)
```

```
FROM VENTAS
```

```
GROUP BY "#fab"
```

- ¿Cómo funciona?
  - Genera un grupo por cada fabricante distinto.
  - De cada grupo devuelve el fabricante y la cantidad de tuplas **de dicho grupo**.

# Ejemplos

---

- Dar el número de fabricante y los promedios de precios a los cuales vendió.

```
SELECT    "#fab", avg(precio)
FROM      VENTAS
GROUP BY  "#fab"
```

- Dar los nombres de fabricante y sus totales de precio vendidos.

```
SELECT    F."#fab" as "nFab", nombre,
sum(precio)
FROM      FABRICANTES F, VENTAS V
WHERE     F."#fab" = V."#fab"
GROUP BY  F."#fab", nombre
```

# GROUP BY - Regla de sintaxis

---

En una sentencia SQL (PostgreSQL) que tiene cláusula GROUP BY, las expresiones en el SELECT pueden ser sólo :

- Atributos presentes en la cláusula GROUP BY.
  - Funciones de agregación sobre atributos.
  - Expresiones aritméticas que utilicen los anteriores.
- El agrupamiento se realiza **después** de aplicar el WHERE. O sea, sobre las tuplas que cumplen la condición.

# Condiciones sobre grupos

---

- Con la cláusula HAVING se pueden especificar condiciones sobre los grupos.
- Por ejemplo:
  - Dar el número de fabricante y los promedios de precios a los cuales vendió, **pero para los fabricantes con más de 3 ventas.**

```
SELECT    "#fab", avg(precio)
FROM      VENTAS
GROUP BY  "#fab"
HAVING count(*) > 3;
```



# Ejemplos

---

- Dar los nombres de fabricantes que hicieron más de 5 ventas sobre productos con #prod mayor que 2, junto con el total de precio vendido.

```
SELECT F."#fab", nombre, sum(precio)
FROM    FABRICANTES F, VENTAS
WHERE    F.."#fab" = VENTAS."#fab"
and     "#prod" > 2
GROUP BY F."#fab", nombre
HAVING count(*) > 5;
```

# Sub-consultas en el FROM

---

- ❑ Equivalente al uso de vistas.
- ❑ Dar el máximo del promedio de ventas de cada fabricante.

```
SELECT MAX (promedio)
      FROM (SELECT  AVG (precio) as promedio
            FROM ventas
            GROUP BY "#fab") as promedios
```

# Valor NULL

---

- SQL permite chequear en una consulta si un valor es nulo.
- Predicados: IS NULL, IS NOT NULL

## □ Ejemplo

- Devolver los nombres de los fabricantes de los que no se conoce el departamento donde trabajan.

```
SELECT nombre  
FROM FABRICANTES  
WHERE departamento IS NULL;
```

# Cambios de formato

---

- Dar el nombre y departamento de los fabricantes separados por " - ".

```
SELECT nombre || '-' || departamento AS fab  
FROM FABRICANTES;
```

- || es el operador de concatenación de string.

# Cambios de formato (2)

---

## □ Mayúsculas y minúsculas

- Dar el peso de los productos con nombre = "play I". No recordamos si la descripción fue ingresada con minúsculas o con mayúsculas.

```
select peso
from PRODUCTOS
where upper (nombre) = 'PLAY I';
```

```
select peso
from PRODUCTOS
where lower (nombre) = 'play i';
```

**EQUIVALENTES**

# Cambios de formato (3)

---

## □ En resumen:

- Existen funciones que permiten manipular los tipos de datos por defecto.
- También hay funciones específicas de cada manejador.
- RECURRIR A LOS MANUALES

# Material de consulta en la WEB

---

## ❑ Tutorial sobre SQL de la W3C

- <http://www.w3schools.com/sql/default.asp>

## ❑ Tutorial SQL y ejecución de consultas

- <http://sqlzoo.net/>

## ❑ Documentación PostgreSQL

- <http://www.postgresql.org/docs/>

## ❑ LABORATORIO

- Vamos a usar PostgreSQL 8.4

- ❑ <http://www.postgresql.org>