

# SQL

---

Curso Fundamentos de Bases de Datos

Instituto de Computación, Facultad de Ingeniería  
Universidad de la República

# SQL

---

- Structured Query Language
- Originalmente llamado SEQUEL (Structured English QUERy Language)
- Lenguaje estándar para los Sistemas de gestión de Bases de Datos Relacionales (DBMS)
  - Primera versión del estándar nace en 1986
  - Lo suceden varias versiones extendiéndolo e incorporando características
  - Actualmente el estándar tiene un *core* y *extensiones especializadas*
- Cada DBMS puede tener diferencias en las funcionalidades del lenguaje, pero todos se apegan al estándar en la mayor parte de lo que proveen y todos son compatibles con el core.

# Lenguaje de base de datos integral

---

- DDL (Data Definition Language)
  - Create Table
  - Alter Table
- DML (Data Manipulation Language)
  - Insert
  - Delete
  - Update
  - Select
- Vistas
- Seguridad y permisos
- Definición de restricciones
- Control de transacciones

# Consulta SQL básica – SELECT-FROM-WHERE

---

SELECT <lista de atributos>

FROM <lista de tablas>

WHERE <condiciones sobre tuplas>

- <lista de atributos> - nombres de atributos cuyos valores devolverá la consulta
- <lista de tablas> - nombres de las tablas requeridas para procesar la consulta
- <condiciones sobre tuplas> - expresión condicional (booleana) que identifica a las tuplas a ser devueltas por la consulta.
- Operadores para comparar valores de atributos entre sí y con constantes:  
=, <, <=, >, >=, <>

# Ejemplo - esquema

---

- FABS (numF, nombre, dir)
- PRODS (numP, desc)
- VENTAS (numF, numP, fecha, cant, precioUnit)

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden el producto con nro 7848

```
SELECT nombre  
FROM Fabs T, Ventas V  
WHERE numP = 7848 AND T.numF = V.numF
```

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden el producto 7848 y también el producto 8747

```
SELECT nombre
FROM Fabs T, Ventas V
WHERE numP = 7848 AND T.numF = V.numF AND
      EXISTS (SELECT *
              FROM Ventas V1
              WHERE V1.numF = T.numF AND
                   V1.numP = 8747)
```

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden el producto 7848 y también el producto 8747

Otra solución:

```
SELECT nombre
FROM Fabs T, Ventas V
WHERE numP = 7848 AND T.numF = V.numF AND
      T.numF IN (SELECT numF
                  FROM Ventas V1
                  WHERE V1.numP = 8747)
```

Operadores  $\in$  y  $\notin$   
como IN y NOT IN

# Ejemplos

---

- Devolver los nombres de los fabricantes que **no** venden el producto 7848

```
SELECT nombre
FROM Fabs T
WHERE NOT EXISTS (SELECT *
                  FROM Ventas V1
                  WHERE V1.numF = T.numF AND
                        V1.numP = 7848)
```

- ¿Cómo se haría utilizando NOT IN?

# Ejemplos

---

- Devolver los nombres de los fabricantes que **no** venden el producto 7848

```
SELECT nombre
FROM Fabs
WHERE numF NOT IN (SELECT numF
                   FROM Ventas
                   WHERE numP = 7848)
```

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden **sólo** el producto 7848

```
SELECT nombre
FROM Fabs T, Ventas V
WHERE numP = 7848 AND T.numF = V.numF AND
      NOT EXISTS (SELECT *
                  FROM Ventas V1
                  WHERE V1.numF = T.numF AND
                       V1.numP <> 7848)
```

- ¿Cómo se haría utilizando NOT IN?

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden **sólo** el producto 7848

```
SELECT nombre
FROM Fabs T, Ventas V
WHERE V.numP = 7848 AND T.numF = V.numF AND
      T.numF NOT IN (SELECT V1.numF
                     FROM Ventas V1
                     WHERE V1.numP <> 7848)
```

# Consulta con cláusula JOIN

---

SELECT <lista de atributos>

FROM <tabla1> **JOIN** <tabla2> **ON** <tabla1.atr1 = tabla2.atr2>

WHERE <condiciones sobre tuplas>

SELECT <lista de atributos>

FROM <tabla1> **NATURAL JOIN** <tabla2>

WHERE <condiciones sobre tuplas>

Otros posibles:

LEFT OUTER JOIN  
RIGHT OUTER JOIN  
FULL OUTER JOIN

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden el producto con nro 7848

```
SELECT nombre
FROM Fabs T NATURAL JOIN Ventas V
WHERE numP = 7848
```

- Devolver los nombres de los fabricantes que venden sólo el producto 7848

```
SELECT nombre
FROM Fabs T NATURAL JOIN Ventas V
WHERE V.numP = 7848 AND
      T.numF NOT IN (SELECT V1.numF
                     FROM Ventas V1
                     WHERE V1.numP <> 7848)
```

# Ejemplos

---

- Devolver los nombres de los fabricantes que venden **todos** los productos con descripción “d1”

```
SELECT nombre
FROM Fabs F
WHERE NOT EXISTS (SELECT *
                  FROM Prods P
                  WHERE P.desc = 'd1' AND
                  NOT EXISTS (SELECT *
                             FROM Ventas V
                             WHERE V.numP = P.numP AND
                                    V.numF = F.numF) )
```

¿Qué pasa si no hay ningún producto con descripción “d1”?

# Ejemplos

---

- ¿Qué pasa si no hay ningún producto con descripción “d1”?

```
SELECT nombre
FROM Fabs F
WHERE NOT EXISTS (SELECT *
                  FROM Prods P
                  WHERE P.desc = 'd1' AND
                        NOT EXISTS (SELECT *
                                   FROM Ventas V
                                   WHERE V.numP = P.numP AND
                                         V.numF = F.numF)) AND
                EXISTS (SELECT *
                       FROM Prods P
                       WHERE P.desc = 'd1')
```

# Otros operadores de elemento-conjunto

---

- Además del IN y NOT IN, existen otros operadores para comparar un elemento con un conjunto
  - = ANY (equivalente a IN)
  - > ANY, >= ANY, < ANY, <= ANY, <> ANY
  - > ALL, >= ALL, < ALL, <= ALL, <> ALL
- Ejemplo: Devolver los nombres de los fabricantes que venden los productos más caros

```
SELECT nombre
FROM Fabs T NATURAL JOIN Ventas V
WHERE V.precioUnit >= ALL (SELECT V1.precioUnit
                           FROM Ventas V1)
```

# Conjuntos de tuplas en SQL

---

- SQL puede devolver tuplas duplicadas. Si queremos eliminarlas utilizamos el término **DISTINCT** en el **SELECT**.

`SELECT DISTINCT <atributo>`

- Ejemplo

```
SELECT DISTINCT precioUnit
FROM Ventas
WHERE numP = 7848
```

**VENTAS**

numF	numP	fecha	cant	precioUnit
1	7848	1/1/2020	50	500
2	126	1/1/2020	4	68
2	127	1/1/2020	1	200
3	7848	1/1/2020	3	450
4	7848	1/1/2020	20	450

# Conjuntos de tuplas en SQL

---

- Incorpora operaciones de conjuntos:

UNION  
EXCEPT  
INTERSECT

- Ejemplo

```
(SELECT precioUnit  
FROM Ventas  
WHERE numP = 7848)  
UNION  
(SELECT precioUnit  
FROM Ventas  
WHERE numP = 7550)
```

Eliminan  
repetidos!

# Ordenar resultados de las consultas

---

- Obtener el monto obtenido para cada día, por número de producto, y nombre del fabricante, ordenado por fecha, número de producto y nombre de fabricante, sólo para los casos de monto mayor que 100.

```
SELECT fecha, numP, nombre, precioUnit*cant  
FROM Ventas NATURAL JOIN Fabs  
WHERE precioUnit*cant > 100  
ORDER BY fecha, numP, nombre
```

# Valor NULL

---

- En SQL el valor NULL puede tener distintas interpretaciones: *desconocido, no disponible, no aplica*.
- Se considera que dos valores NULL nunca son iguales, por lo tanto no se pueden comparar como los otros valores.
- Funciones **IS NULL** y **IS NOT NULL**
- Ejemplo

```
SELECT nombre  
FROM Fabs  
WHERE dir IS NULL
```

# Funciones de agregación

---

- Agregar (totalizar) valores de múltiples tuplas
- Funciones: **COUNT, SUM, AVG, MAX, MIN**
- Ejemplos

```
SELECT COUNT(*)  
FROM Ventas  
WHERE precioUnit > 100
```

```
SELECT AVG(precioUnit)  
FROM Ventas  
WHERE numF = 4
```

¿Cuántos valores devuelven estas consultas?

# Agrupar tuplas

---

- Devolver el fabricante y la cantidad de ventas de productos que tienen precio unitario mayor que 100

```
SELECT numF, COUNT(*)  
FROM Ventas  
WHERE precioUnit > 100  
GROUP BY numF
```

Cuenta cantidad  
de tuplas en  
cada grupo

# Agrupar tuplas

---

- Devolver el fabricante, **su nombre** y la cantidad de ventas **para cada producto**, que tienen precio unitario mayor que 100

```
SELECT numF, nombre, numP, COUNT(*)  
FROM Ventas NATURAL JOIN Fabs  
WHERE precioUnit > 100  
GROUP BY numF, nombre, numP
```

# Condiciones sobre grupos

---

- Dar, para los fabricantes que vendieron más de 5 productos distintos, el promedio de cantidad vendida de los productos, cuyo precio unitario es mayor que 100.

```
SELECT numF, nombre, AVG(cant)
FROM Ventas NATURAL JOIN Fabs
WHERE precioUnit > 100
GROUP BY numF, nombre
HAVING COUNT (DISTINCT numP) > 5
```

Selecciona los grupos  
que tienen al menos 6  
numP distintos

# Estructura básica

---

