SQL

José Abásolo

Diana Benavides

SQL

Lenguajes:

- → **Cálculo Relacional**
- → **Álgebra Relacional**
- → SQL

Concepto: Lenguaje estructurado de consulta.

Estándares SQL86, SQL89,**SQL92**, SQL1999 y SQL2003.

SQL

Componentes:

- DDL
  - Esquemas
  - Reglas de integridad
  - Vistas
  - Acceso a relaciones, vistas, etc.
- **DML**
- Control de transacciones
- SQL embebido y SQL dinámico.

SQL: Estructura básica

SELECT **X**

FROM **Y**

[WHERE **Z**]

X = Proyección del AR

Y = Producto cartesiano de AR

Z = Selección en AR

SQL: Estructura básica: SELECT

SELECT nom_beb

FROM Bebedor

SELECT distinct nom_beb

FROM Bebedor

SELECT all nom_beb

FROM Bebedor

SELECT *

FROM Bebedor

SQL: Estructura básica: WHERE

SELECT nom_beb

FROM Bebedor

WHERE edad > 23

SELECT nom_beb

FROM Bebedor

WHERE edad > 23 AND id_beb>1

SELECT nom_beb

FROM Bebedor

WHERE edad BETWEEN 20 AND 30

SQL: Estructura básica: FROM

SELECT nom_beb

FROM Bebedor

SELECT nom_beb, id_bar

FROM Bebedor, Frecuenta

WHERE Bebedor.id_beb = Frecuenta.id_beb

SELECT nom_beb, id_bar

FROM Bebedor, Frecuenta

WHERE Bebedor.id_beb = Frecuenta.id_beb AND edad>20

SQL: RENOMBRAMIENTO

```
SELECT nom_beb as NombreBebedor
FROM Bebedor
```

```
SELECT B1.nom_beb, B2.nom_beb
FROM Bebedor B1, Bebedor B2
WHERE B1.edad=B2.edad AND B1.id_beb<>B2.id_beb
```

SQL: CADENAS DE CARACTERES

Caracter %: Cualquier subcadena

Caracter _: Cualquier caracter

Ejemplo:

‘Per%’: Cualquier cadena que empiece con ‘Per’

‘%er%’: Cualquier cadena que contenga ‘er’

‘_ _ _’: Cualquier cadena de tres caracteres

‘_ _ _ %’: Cualquier cadena de al menos tres caracteres

```
SELECT nom_beb
FROM Bebedor
WHERE nom_beb LIKE ‘%an%’
```

SQL: ORDEN EN LAS TUPLAS RESULTADO

```
SELECT nom_beb
FROM Bebedor
ORDER BY edad
```

```
SELECT nom_beb
FROM Bebedor
ORDER BY edad DESC
```

```
SELECT nom_beb
FROM Bebedor
ORDER BY nom_beb, edad
```

SQL: UNION

(Relación A) **union** (Relación B)

(Relación A) **union all** (Relación B)

Suponiendo compatibilidad, seleccione el identificador de los bebedores que gustan de alguna cerveza o frecuentan algún bar

```
(SELECT id_beb FROM GUSTA)
union
(SELECT id_beb FROM FRECUENTA)
```

SQL: INTERSECT

(Relación A) **intersect** (Relación B)

(Relación A) **intersect all** (Relación B)

Seleccione los nombres de los bebedores que gustan de alguna cerveza y además frecuentan algún bar

```
(SELECT nom_beb
FROM Bebedor, Gusta
WHERE Bebedor.id_beb = Gusta.id_beb)
intersect
(SELECT nom_beb
FROM Bebedor, Frecuenta
WHERE Bebedor.id_beb = Frecuenta.id_beb)
```

SQL: EXCEPT

(Relación A) **except** (Relación B)

(Relación A) **except all** (Relación B)

Suponiendo compatibilidad, seleccione el identificador de los bares que sirven alguna cerveza pero no son frecuentados por ningún bebedor

```
(SELECT id_bar  FROM SIRVE)
except
(SELECT id_bar FROM FRECUENTA)
```

SQL: AGREGACIÓN

Cláusula **GROUP BY**

Funciones conjuntos de valores de entrada, para generar un único valor de salida:

- **sum, avg, min, max, count**

Seleccione el promedio de edad de los bebedores que frecuentan cada bar

```
SELECT id_bar, avg(edad) as EdadPromedio
FROM Bebedor, Frecuenta
WHERE Bebedor.id_beb=Frecuenta.id_beb
GROUP BY (id_bar)
```

SQL: AGREGACIÓN

Se puede imponer condiciones sobre los grupos, con la cláusula **HAVING**:

Seleccione el promedio de edad de los bebedores que frecuentan cada bar, para promedios mayores a 20:

```
SELECT id_bar, avg(edad) as EdadPromedio
FROM Bebedor, Frecuenta
WHERE Bebedor.id_beb=Frecuenta.id_beb
GROUP BY (id_bar)
HAVING avg(edad) > 20
```

SQL: AGREGACIÓN

Se suele utilizar **count(*)** para contar el número de tuplas de una relación:

Seleccione el número total de bebedores

```
SELECT count(*)
FROM Bebedor
```

SQL: VALORES NULOS

Palabra clave: **null**

is null
is not null

Seleccione el nombre de los bebedores que no tienen registrada una dirección

```
SELECT nom_beb
FROM Bebedor
WHERE dir_beb is null
```

SQL: VALORES NULOS

Predicados con valores nulos:

- **and:**
  - Cierto y desconocido = desconocido
  - Falso y desconocido = falso
  - Desconocido y desconocido = desconocido
- **or:**
  - Cierto y desconocido = cierto
  - Falso y desconocido = desconocido
  - Desconocido y desconocido = desconocido

SQL: SUBCONSULTAS ANIDADAS

SELECT X

FROM Y

WHERE Z conectiva (SELECT A FROM B WHERE C)

Comprobaciones sobre:

- Pertenencia a conjuntos
- Comparación de conjuntos
- Relaciones vacías

SQL: SUBCONSULTAS ANIDADAS: Pertenencia a conjuntos

Conectiva **in**, **not in**

Permite evaluar la pertenencia de una tupla a una relación

Seleccione los nombres de los bebedores que gustan de alguna cerveza y además frecuentan algún bar

```
SELECT nom_beb
FROM Bebedor, Gusta
WHERE Bebedor.id_beb = Gusta.id_beb
AND nom_beb IN (SELECT nom_beb
FROM Bebedor, Frecuenta
WHERE Bebedor.id_beb = Frecuenta.id_beb )
```

SQL: SUBCONSULTAS ANIDADAS: Comparación de conjuntos

Conectivas **some**, **all**

Permiten comparar el valor de un atributo de una tupla contra los valores de las tuplas de otra relación, mediante operadores: >, <, >=, <=, =, <>

Seleccione los nombres del(los) bebedor(es) de mayor edad

```
SELECT nom_beb
FROM Bebedor
WHERE edad >= all (SELECT edad FROM Bebedor)
```

SQL: SUBCONSULTAS ANIDADAS: Comprobación de relaciones vacías

Conectivas **exists**, **not exists**

Permite comprobar si una subconsulta produce o no produce ninguna tupla de resultado.

Seleccione los nombres de los bebedores que gustan de alguna cerveza y además frecuentan algún bar

```
SELECT nom_beb
FROM Bebedor as B1
WHERE exists (SELECT nom_beb
FROM Bebedor as B2
WHERE B1.id_beb=B2.id_beb)
```

Ejercicios SQL

BEBEDOR

id_beb	nom_beb	dir_beb	tel_beb	edad
CP				

BAR

id_bar	nom_bar	dir_bar	tel_bar
CP			

CERVEZA

id_cerv	nom_cerv	grado
CP		

FRECUENTA

id_beb	id_bar
CP, CF1	CP, CF2

SIRVE

id_bar	id_cerv
CP, CF1	CP, CF2

GUSTA

id_beb	id_cerv
CP, CF1	CP, CF2

Ejercicios SQL

1. Obtener los nombres y edades de las bebedoras
2. Obtener los nombres de los bebedores que no tienen registrada la edad
3. Obtener los nombres de los bebedores hombres que no tienen registrada la edad
4. Obtener las diferentes edades de los bebedores
5. Obtener todos los datos de los bebedores cuyas direcciones comienzan con ‘Calle’
6. Obtener todos los datos de los bebedores mayores a 35 años, ordenados por nombre
7. Obtener las cervezas con grado de alcohol entre 75 y 100

Ejercicios SQL (2)

1. Obtener los nombres de los bares que sirven al menos una cerveza que le gusta al bebedor “Juan Pérez”
2. Obtener los pares Nombre Bebedor, Nombre Cerveza de la que gusta el bebedor, ordenados de forma ascendente por Nombre Bebedor y Nombre Cerveza
3. Obtener los pares Nombre Cerveza, Número de Bebedores, ordenados por el número de bebedores que gustan cada cerveza
4. Obtener el nombre del bar más frecuentado, con el número de bebedores que lo frecuentan
5. Obtener el nombre del bar y el promedio de grado de alcohol de las cervezas que sirve, donde este promedio es mayor a 50
6. Obtener el nombre de las cervezas que tienen un grado de alcohol mayor a al menos alguna otra cerveza

Ejercicios SQL (3)

1. Dar los datos de los bares que frecuenta el bebedor de nombre “Juan Pérez”.
2. Dar los datos de los bebedores que frecuentan al menos un bar que sirve al menos una cerveza que les gusta.
3. Dar los datos de los bebedores que frecuentan al menos un bar que no sirve cervezas que le gustan.
4. Dar los datos de los bebedores que solo frecuentan bares que sirven al menos una cerveza que les gusta.
5. Dar los datos de los bebedores que solo frecuentan bares que no sirven cervezas que les gustan.
6. Dar los datos de los bebedores a quienes les gustan todas las cervezas.

FIN DE LA PRESENTACIÓN