Bases de Datos
Ayudantía 2: SQL

Profesor: Víctor Reyes

Ayudantes: Diego Banda, Felipe Gutiérrez

Contactos

Diego Banda

diego.banda@mail.udp.cl

darklouds

Felipe Gutiérrez

felipe.gutierrez_l@mail.udp.cl

felipx#0485

SQL: Structured Query Language

Lenguaje estándar para consultas en bases de datos relacionales (algunas no relacionales también jeje)

Permite:

- Realizar consultas.
- Creación y modificación de los datos.
- Administración de la base de datos: creación de perfiles, vistas, permisos, etc (esto va más ligado a la BD en sí que al lenguaje, en este caso MySQL).

MySQL®

Estructura

SELECT <atributos y/o funciones de agregación>

FROM <tablas>

WHERE <condiciones>

- Para seleccionar todos los atributos se utiliza el símbolo “ * “ (no siempre es recomendable utilizarlo).
- Se pueden abreviar los nombres de las tablas para su posterior utilización (Ej: Persona AS p).
- No es obligatoria la condición (WHERE), solo si es necesaria según contexto.

Ejemplos

```sql
SELECT *
FROM jugadores;
```

Selecciona todos los atributos de todos los jugadores.

Selecciona solo el nombre y edad de los jugadores que tengan por juego favorito ‘Elden Ring’.

```sql
SELECT nombre, edad
FROM jugadores
WHERE juego_favorito = 'Elden Ring';
```

Funciones de agregación

Basicamente funciones que hacen cositas :p

- **SUM()**: Retorna la suma de los valores.
- **COUNT()**: Retorna la cantidad de valores.
- **AVG()**: Retorna el promedio de los valores.
- **MIN()**: Retorna el valor mínimo.
- **MAX()**: Retorna el valor máximo.

Estructura

SELECT funcion(<atributo>**)**
FROM <tabla>
WHERE <condicion>

- **DISTINCT**: Para tomar solo valores distintos.
- También se puede utilizar el comando AS para nombrar una columna o tabla temporal.

Ejemplos

```sql
SELECT AVG(edad)
FROM jugadores;
```

Muestra la edad promedio de los jugadores

```sql
SELECT COUNT(nombre)
FROM jugadores
WHERE juego_favorito = 'Dragon Ball: Sparking Zero';
```

Cuenta los jugadores que tengan como juego favorito ‘Dragon Ball: Sparking Zero’

Operadores

Viéndolo como programación

AND: Ambas condiciones deben cumplirse → &&

OR: Una de las dos condiciones debe cumplirse → ||

NOT: Debe cumplirse el opuesto de lo propuesto → !=

Ejemplo

```sql
SELECT nombre
FROM jugadores
WHERE juego_favorito = 'Elden Ring' OR juego_favorito = 'Dark Souls 3';
```

Muestra los nombres de los jugadores que tengan como juego favorito ‘Elden Ring’ o ‘Dark Souls 3’

Operadores

- **BETWEEN:** Busca los elementos dentro de un rango del tipo [x, y], puede ser negado (NOT BETWEEN).
- **IN:** Busca valores específicos para un atributo.

```sql
SELECT nombre
FROM jugadores
WHERE edad BETWEEN 18 AND 30;
```

Muestra el nombre de los jugadores que tengan entre 18 y 30 años

```sql
SELECT nombre, juego_favorito
FROM jugadores
WHERE nombre IN ('Victor', 'Yerko');
```

Muestra el nombre y juego favorito de los jugadores que se llamen ‘Victor’ o ‘Yerko’

Operadores

- ORDER BY: Permite ordenar lo mostrado en pantalla según atributo especificado.
- ASC: Orden ascendente (predeterminado), en caso de texto es alfabéticamente.
- DESC: Orden descendente.

```sql
SELECT nombre, edad
FROM jugadores
ORDER BY edad DESC;
```

Mostrará los nombres y edades de los jugadores en orden descendente según la edad

```sql
SELECT nombre
FROM jugadores
ORDER BY nombre ASC;
```

Muestra los nombres de los jugadores en orde ascendente (alfabéticamente)

Operadores

- GROUP BY: Agrupa datos según uno o más atributos.
- HAVING: Es como un WHERE, pero aplica la condición después de procesar una función de agregación.

```sql
SELECT juego_favorito, COUNT(*) AS cantidad_jugadores
FROM jugadores
GROUP BY juego_favorito;
```

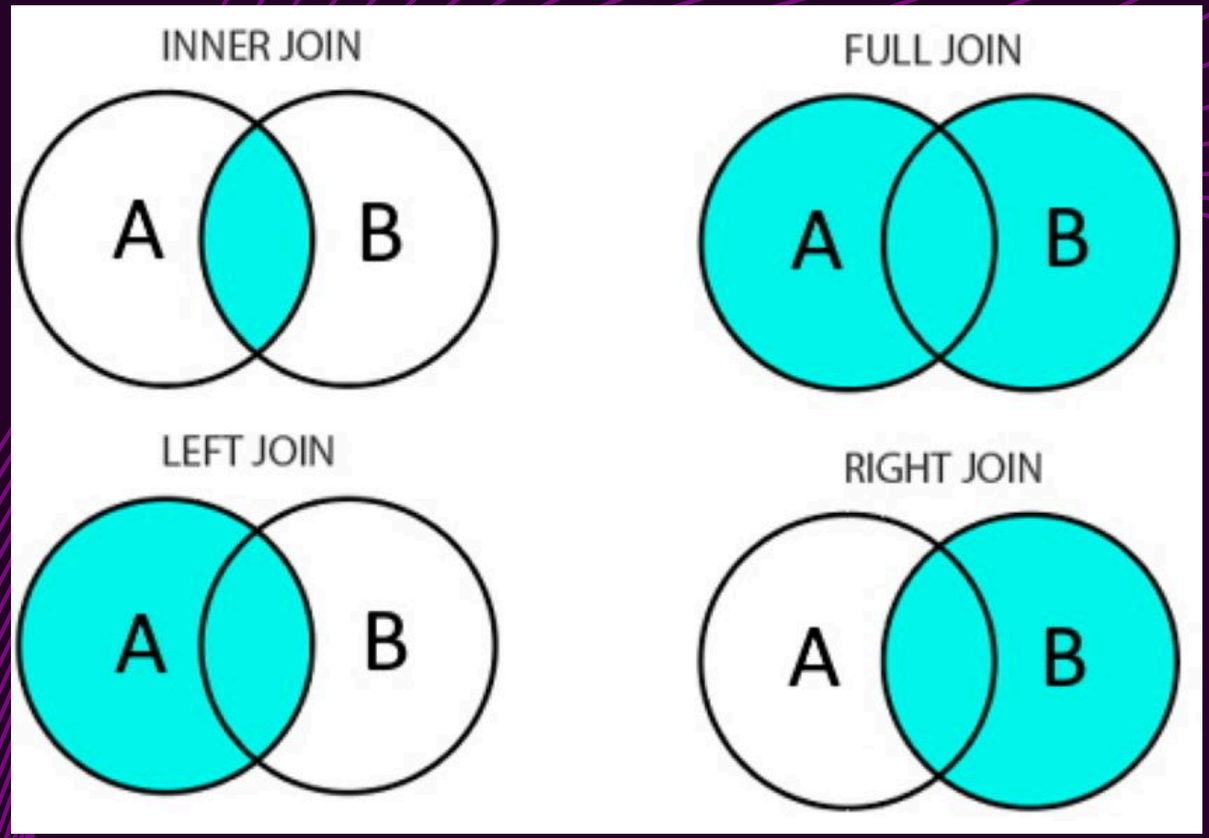
Agrupa según juego favorito y muestra en pantalla el juego con la cantidad de jugadores que lo tienen asociado como Juego Favorito

```sql
SELECT juego_favorito, AVG(edad) AS edad_promedio
FROM jugadores
GROUP BY juego_favorito
HAVING edad_promedio > 25;
```

Agrupa según juego favorito, luego calcula promedio de edad para cada grupo de juegos favorito y finalmente filtra por edad_promedio > 25, mostrará por pantalla el nombre del juego favorito y el promedio de edad de los jugadores, pero solo si el promedio es mayor a 25.

Unión de tablas

- **INNER JOIN:** Join por defecto, mostrará solo las columnas de ambas tablas que muestren coincidencias.
- **LEFT JOIN:** Muestra todos los datos de la tabla de la izquierda y los de la derecha en caso de existir, en caso contrario muestra NULL.
- **RIGHT JOIN:** Muestra todos los datos de la tabla de la derecha y los de la izquierda en caso de existir, en caso contrario muestra NULL.
- **FULL JOIN:** Mostrará todos los datos de ambas tablas, en caso de no existir conexión muestra NULL.



Ejercicio

Se tiene la siguiente base de datos:

- **Jugadores:** id, nombre, edad, juego_favorito_id
- **Juego:** id, nombre, genero, consola
- **Estadisticas_Jugador:** id, jugador_id, juego_id, nivel, horas_jugadas, platino

Realice las siguientes consultas:

1.-Nombre y genero de juegos para ‘PC’.

2.-Nombre de jugadores que tienen juego favorito alguno con genero ‘SOULS LIKE’.

3.-Nombre de juego, jugador con más horas jugadas y las horas jugadas.

4.-Nombre y horas de jugadores de ‘League of Legends’ ordenados por sus horas descendientemente.

5.-Nombre de jugador y nombre de juego en el cual el jugdor tiene platino.

6.-Jugador con mayor nivel en cada juego.