PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE

AYUDANTÍA 7

SQL: Consultas

Ayudante: Felipe Fuentes Porras

CONSULTAS

Ya tenemos nuestra base de datos armada, llena de información. Ahora queremos acceder a esta, utilizar la información guardada, como lo hacemos:

- CON CONSULTAS

ESTRUCTURA DE UNA CONSULTA BASICA

SELECT: Se especifican que atributos de la tabla se desean mostrar y manipular (lo que esta en el select es lo que se retorna)

FROM: De que tablas estamos sacando la información, si se seleccionan varias tablas estas se cruzan.

WHERE: Se agregan condicionales para la construcción de la consulta.

EJEMPLO:

Consulta que retorna las filas cuyo en cuyo atributo “nombre” sea igual a “Bundesliga”.

```
"""SELECT *
    FROM Ligas
    WHERE Ligas.nombre= "Bundesliga"""
```

SELECCIONAR ATRIBUTOS A RETORNAR

- Especificar la tabla y el atributo. Un * retorna toda la fila. Con DISTINCT se eliminan los duplicados.
- Ejemplo con una tabla:

```
"""SELECT nombre
    FROM Ligas
    WHERE Ligas.nombre= "Bundesliga"""
```

- Ejemplo con 2 o más tablas:

```
"""SELECT Ligas.nombre, Partidos.fecha
    FROM Ligas, Partidos
    WHERE Ligas.nombre= "Bundesliga
    AND Ligas.lid= Partidos.liga"""
```

JOIN ENTRE TABLAS

- Queremos comparar 2 tablas, y que cada fila se relacione correctamente con su contraparte en la otra tabla.

```
"""SELECT Ligas.nombre, Partidos.fecha
    FROM Ligas, Partidos
    WHERE Ligas.nombre= "Bundesliga
    AND Ligas.lid= Partidos.liga"""
```

MATCHING THE STRING

Se puede hacer estrictamente igual usando un = o hacer que sea parcialmente igual con un LIKE:

```
SELECT *
FROM Estudiantes E
WHERE E.nombre LIKE 'MA%'
```

MANEJO DE CONJUNTOS

- UNION : Junta 2 tablas en una consulta. La segunda tabla se anexa al final de la primera
- INTERSECT: Las filas que se encuentran en ambas tablas.
- EXCEPTS: Se retornan todos los valores menos los que no están en la segunda tabla.
- IN: Si es que esta dentro de la segunda tabla.

CONSULTAS ANIDADAS

- Una consulta dentro de otra.
- Se puede añadir dentro de un WHERE o FROM, entre otros

```
SELECT E.nombre
FROM Estudiantes E
WHERE E.id NOT IN (SELECT I.id_estudiante
                   FROM Inscritos I
                   WHERE I.id_curso = 'IIC2115')
```

GROUP BY-HAVING

- Agrupa valores iguales de una columna que nosotros seleccionamos.
- HAVING: Agrega condicionalidades después de la agrupación (WHERE no sirve para aplicar condiciones con atributos ligados al GROUP BY)
- Ejemplo: Agrupar por liga y partidos.OJO que hay que especificar que queremos hacer al agrupar la fila, sino retorna una fila cualquiera (como en este caso).

```
"""SELECT *
    FROM Ligas, Partidos
    WHERE Ligas.lid= Partidos.liga
    GROUP BY Ligas.nomre, Partido.equipos"""
```

AGREGACIÓN

- COUNT ([DISTINCT] A): La cantidad de valores (unicos) en la columna A.
- SUM ([DISTINCT] A): La suma de todos los valores (unicos) en la columna A.
- AVG ([DISTINCT] A): El promedio de todos los valores (unicos) en la columna A.
- MAX (A): El máximo valor en la columna A.
- MIN (A): El mínimo valor en la columna A.

PYTHON Y SQLITE

Fetchone(): Entrega el primer valor,
Fetchall(): Retorna todos, una lista con tuplas.

```
connection = sqlite3.connect('example.db')
cursor = connection.cursor()
cursor.execute('SELECT * FROM countries')

#obtenemos sólo una fila, si queremos todas, debemos usar fetchall (no cambia el resultado en este caso)
print(cursor.fetchone())
connection.close()
```

PARAMETRIZAR

```
connection = sqlite3.connect('example.db')
cursor = connection.cursor()
country = "'China'"
cursor.execute("SELECT * FROM countries WHERE name = %s" % country)
print(cursor.fetchone())
```

```
country = ('Andorra',)
cursor.execute('SELECT * FROM countries WHERE name = ?', country)
print(cursor.fetchall())
```