Laboratorio #4

- **CONSULTAS SIMPLES CON SELECT Y JOINS**

Consultas SQL (SELECT)

La sentencia SELECT es una sentencia DML (Lenguaje de Manipulación de Datos) utilizada para seleccionar datos de una base de datos, es también conocida como consulta de selección.

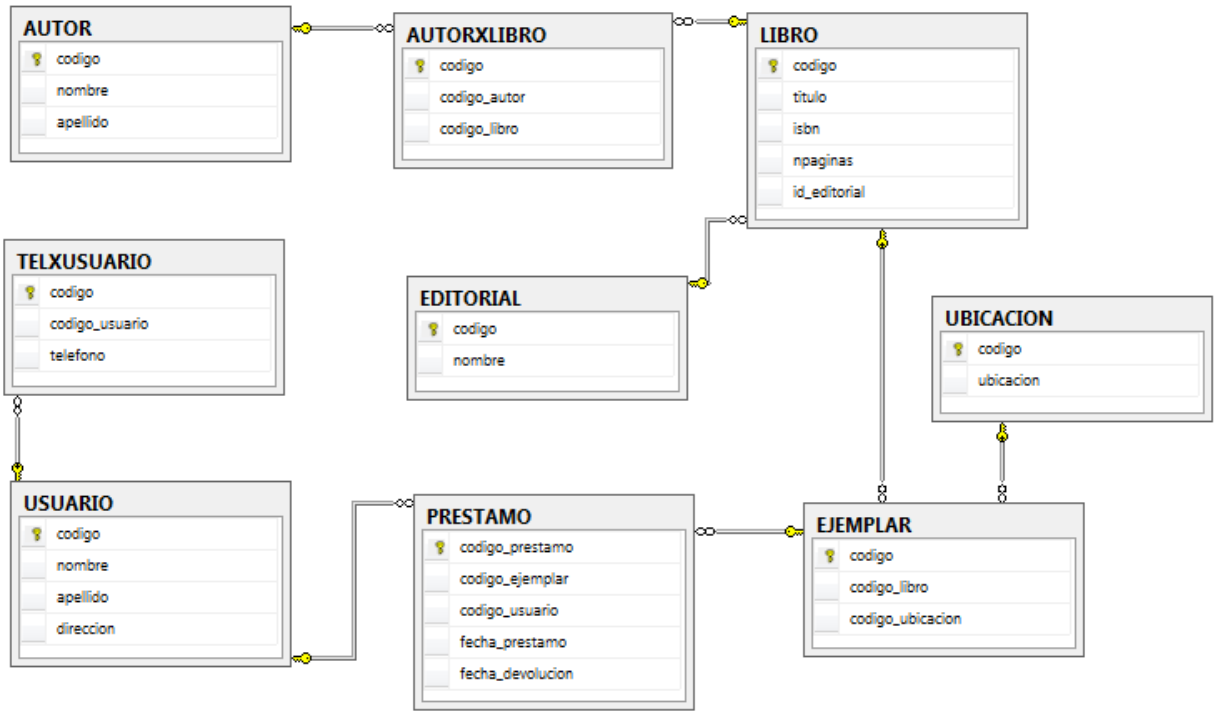
Estructura del SELECT.

La sintaxis básica de una consulta de selección es la siguiente:

```
SELECT [CAMPOS] FROM [TABLA] [WHERE [CONDICION]];
```

En donde [CAMPOS] son los atributos o columnas que se desean extraer de la [TABLA] que se especifique y WHERE es opcional, esta condición será utilizada más adelante.

Consideremos la siguiente base de datos:



Si se quiere extraer solamente el nombre y apellido de la tabla USUARIO se hace de la siguiente manera:

```
SELECT nombre, apellido FROM usuario;
```

Dando como resultado:

	nombre	apellido
1	Maria	Perez
2	Juan	Miranda
3	Daniel	Lopez
4	Lucy	Cardoza
5	Monica	Jimenez

Si se desea extraer todos los campos de una tabla se utiliza * de la siguiente manera:

```
SELECT * FROM usuario;
```

Esta instrucción devolverá todos los registros y todos los campos que la tabla posea.

ALIAS (AS)

En muchos casos es necesario asignar un nombre a alguna columna o tabla, cuando se le asigna un alias a una columna el nombre de la columna cambia en el resultado, mientras que cuando se usa el alias en la tabla este sirve para identificar campos que tengan el mismo nombre en distintas tablas.

Por ejemplo se desea extraer todos los nombres de la tabla AUTOR y todos los nombres de la tabla USUARIO, pero que al mostrarse se llamen Nombres de Autores y Nombres de Usuarios; *¿Qué problema habría si se quita el alias a las tablas?*

```
SELECT u.nombre as "Nombre usuario",a.nombre as "Nombre autor" FROM usuario as u,autor as a;
```

¿Por qué el resultado de esta consulta está lleno de duplicidad?

Cláusula condicional WHERE.

Esta cláusula devuelve un subconjunto de los registros de una tabla que cumplan con dicha condición.

```
SELECT * FROM LIBRO WHERE npaginas = 300;
```

Esta consulta devuelve los registros de la tabla LIBRO que cumplan la condición de tener 300 páginas.

Otros operadores que se pueden usar en la cláusula WHERE:

Operador	Significado
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que

Además de estos operadores lógicos se disponen otros adicionales:

BETWEEN

Indica un intervalo de valores:

```
SELECT * FROM LIBRO WHERE npaginas BETWEEN 1 AND 500;
```

Esta instrucción devuelve los registros de la tabla LIBRO que tengan entre 1 a 500 páginas.

LIKE

Sirve para comparar patrones de texto:

```
SELECT * FROM AUTOR WHERE nombre LIKE 'l%';
```

Devuelve todos los autores cuyo nombre comience con “L”.

Lista de patrones de texto: https://www.w3schools.com/sql/sql_wildcards.asp

IN

Sirve para comparar con una lista de valores fijos:

```
SELECT num, calle, direccion
FROM DIRECCION
WHERE ciudad IN ('Sevilla', 'Córdoba', 'Huelva', 'Cádiz')
```

IS NULL / IS NOT NULL

Devuelve los registros que sean o no sean nulos. Este operador lógico se verá más adelante cuando se trabajen los JOINS.

Operadores lógicos.

Sirven para componer expresiones y filtrar aún más los registros de una tabla.

Operador	Significado
AND	Y lógico
OR	O lógico
NOT	Negación lógica

```
SELECT * FROM LIBRO WHERE npaginas > 550 AND titulo LIKE 'a%';
```

Esta consulta devuelve los registros de la tabla libro que tengan más de 550 páginas y el título comience con la letra a.

Consultas entre tablas JOIN.

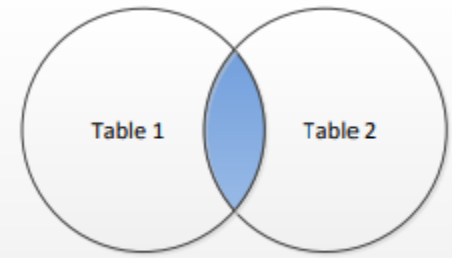
Se pueden generar consultas en las cuales es necesario obtener datos de varias tablas relacionadas entre sí.

La forma básica de hacerlo es utilizando la cláusula condicional WHERE también conocido como INNER JOIN implícito:

```
SELECT l.titulo FROM libro as l,ejemplar as e
WHERE l.codigo = e.codigo_libro;
```

A continuación se verán los principales tipos de JOINS. Imágenes cortesía de Steve Stedman.

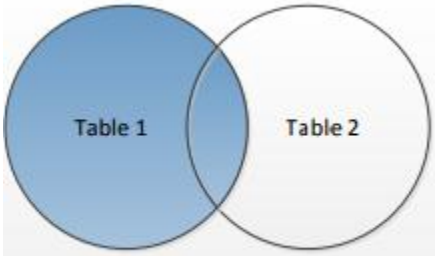
INNER JOIN.




```
SELECT *
  FROM Table_1 t1
 INNER JOIN Table_2 t2
    ON t1.id = t2.fk;
```

Esta consulta obtiene lo mismo que el JOIN implícito.

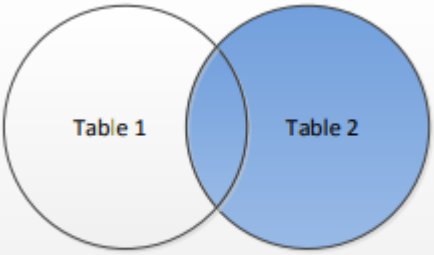
LEFT JOIN.




```
SELECT *
  FROM Table_1 t1
  LEFT JOIN Table_2 t2
    ON t1.id = t2.fk;
```

El resultado de esta consulta contiene todos los registros de la relación izquierda (primera tabla indicada) y aquellos de la tabla derecha que cumplen la condición, para los demás aparecerá en los campos correspondientes un NULL.

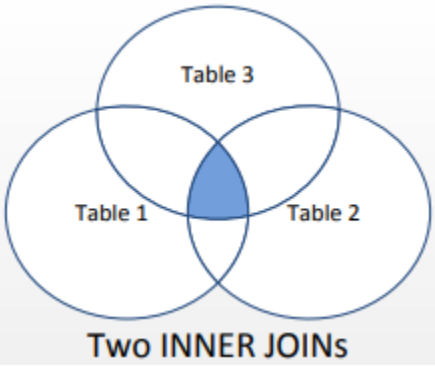
RIGHT JOIN.




```
SELECT *
  FROM Table_1 t1
 RIGHT JOIN Table_2 t2
    ON t1.id = t2.fk;
```

Es análogo al LEFT JOIN, pero devolviendo todos los registros de la relación derecha (segunda tabla que aparece), y únicamente aquellos de la tabla izquierda que cumplen la condición del JOIN. Y aquellos sin equivalente en la parte izquierda tendrán en los campos correspondientes a dicha tabla un NULL.

TWO INNER JOINS.




```
SELECT *
  FROM Table_1 t1
 INNER JOIN Table_2 t2
    ON t1.id = t2.fk
 INNER JOIN Table_3 t3
    ON t1.id = t3.fk;
```

Para conocer los otros tipos de JOINS ver el archivo “JOINS.pdf”

EJEMPLOS.

Ejemplo 1: Mostrar los libros que no han sido prestados en los últimos 6 meses.

```
SELECT l.titulo, p.fecha_prestamo FROM LIBRO as l
 inner join EJEMPLAR as e ON l.codigo = e.codigo_libro
 inner join PRESTAMO as p ON e.codigo = p.codigo_ejemplar
 WHERE fecha_prestamo NOT BETWEEN CAST('2018-12-12'  AS DATE) AND CAST('2019-05-05'  AS DATE);
```

Ejemplo 2: Mostrar que libros ha prestado el usuario Sara.

```
SELECT U.nombre AS Usuario, L.titulo AS Libro FROM USUARIO AS U INNER JOIN PRESTAMO P
 ON U.codigo = P.codigo_usuario INNER JOIN EJEMPLAR E
 ON E.codigo = P.codigo_ejemplar INNER JOIN LIBRO AS L
 ON L.codigo = E.codigo_libro WHERE U.nombre='Daniel';
```