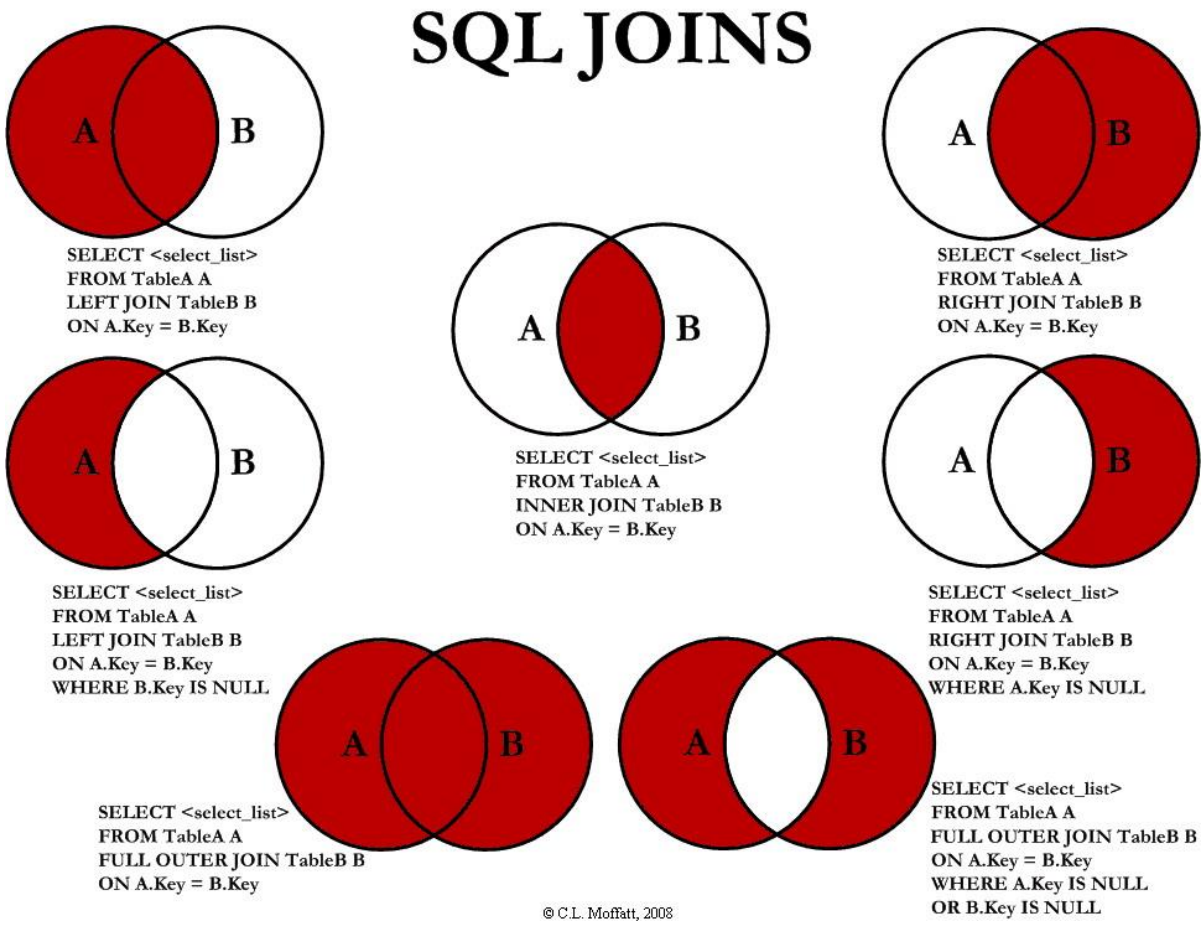
8.- CONSULTAS AVANZADAS

8.1.- INNER, LEFT Y RIGHT JOINS

Las más comunes suelen ser las uniones internas entre tablas (inner join). Un inner join, se podría representar como la intersección entre dos conjuntos, es decir, una consulta con inner join mostraría los registros de las tablas que coincidan en los campos de la unión que se ha definido en dicha consulta.

En cuanto left join recogemos todos los datos de la tabla que está a la izquierda de la unión en la consulta y si usamos en cambio right join las de la derecha.

De manera gráfica tenemos:



8.2.- UNION, INTERSETC Y MINUS

La sintaxis es la siguiente:

```
[Instrucción SQL 1]
UNION [ALL] | INTERSECT | MINUS
[Instrucción SQL 2];
```

- UNION

El propósito del comando SQL UNION es combinar los resultados de dos consultas juntas. En este sentido, UNION es parecido a joins, ya que los dos se utilizan para información relacionada en múltiples tablas. Una restricción de UNION es que todas las columnas correspondientes necesitan ser del mismo tipo de datos. También, cuando utilizamos UNION, sólo se seleccionan valores distintos (similar a SELECT DISTINCT).

Si se quieren unir dos SELECT independientes y con estructura homogénea se utilizan las cláusulas UNION y UNION ALL.

Si especificamos la cláusula UNION no aparecerán los registros que estén repetidos, mientras que, con la cláusula UNION ALL, si aparecerán los registros repetidos.

- INTERSECT

Parecido al comando UNION, INTERSECT también opera en dos instrucciones SQL. La diferencia es que, mientras UNION actúa fundamentalmente como un operador OR (O) (el valor se selecciona si aparece en la primera o la segunda instrucción), el comando INTERSECT actúa como un operador AND (Y) (el valor se selecciona si aparece en ambas instrucciones).

- MINUS

MINUS opera en dos instrucciones SQL. Toma todos los resultados de la primera instrucción SQL, y luego sustrae aquellos que se encuentran presentes en la segunda instrucción SQL para obtener una respuesta final. Si la segunda instrucción SQL incluye resultados que no están presentes en la primera instrucción SQL, dichos resultados se ignoran.

Ejemplo:

Tabla ***Store_Information***

Store_Name	Sales	Txn_Date
Los Angeles	1500	05-Jan-1999
San Diego	250	07-Jan-1999
Los Angeles	300	08-Jan-1999
Boston	700	08-Jan-1999

Tabla ***Internet_Sales***

Txn_Date	Sales
07-Jan-1999	250
10-Jan-1999	535
11-Jan-1999	320
12-Jan-1999	750

```
SELECT Txn_Date
FROM Store_Information
UNION
SELECT Txn_Date
FROM Internet_Sales;
```

```
SELECT Txn_Date
FROM Store_Information
INTERSECT
SELECT Txn_Date
FROM Internet_Sales;
```

```
SELECT Txn_Date
FROM Store_Information
MINUS
SELECT Txn_Date
FROM Internet_Sales;
```

05-Jan-1999 07-Jan-1999 08-Jan-1999 10-Jan-1999 11-Jan-1999 12-Jan-1999	07-Jan-1999	05-Jan-1999 08-Jan-1999

Ejemplo 2:

```
SELECT NOMEM, SALAR, 'SIN' "COMISION"
FROM TEMPLE
WHERE COMIS IS NULL AND NUMDE = 112
UNION
SELECT NOMEM, SALAR + COMIS, ' CON'
FROM TEMPLE
WHERE COMIS IS NOT NULL AND NUMDE=112;

 SELECT NOMEM, SALAR, 'SIN' "COMISION"
FROM TEMPLE
WHERE COMIS IS NULL AND NUMDE = 112
UNION ALL
SELECT NOMEM, SALAR + COMIS, ' CON'
FROM TEMPLE
WHERE COMIS IS NOT NULL AND NUMDE =112;
```

8.3.- LA CLAUSULA EXIST

Este operador devuelve verdadero si la consulta que le sigue devuelve algún valor. Si no, devuelve falso. Se utiliza sobre todo en consultas correlacionadas.

Ejemplo

Obtener el nombre de los centros de trabajo si hay alguno que este en la calle Atocha.

```
SELECT NOMCE
FROM  TCENTR
WHERE EXISTS (SELECT *
        FROM TCENTR
        WHERE DOMI LIKE '%ATOCHA%')
ORDER BY NOMCE;
```