SUBCONSULTAS

SELECT

```
SELECT columna
FROM tabla
WHERE condicion
```

INSERT

```
INSERT INTO tabla (columna1, columna2, columna3…)
VALUES (valor1, valor2, valor3…)
```

UPDATE

```
UPDATE tabla
SET columna 1 = valor1, columna2 = valor2, columna3 = valor3 ….
WHERE condicion
```

DELETE

```
DELETE FROM tabla
WHERE condicion
```

Combinación de dos tablas: Sintaxis

```
SELECT ...
FROM tabla1
        INNER JOIN Tabla2
        ON condición

SELECT COMARQUES.
        nom_c,
        POBLACIONS.
        nom
FROM COMARCAS
        INNER JOIN POBLACIONES
        ON COMARQUES.nom_c = POBLACIONS.nom_c
```

-
-

INNER JOIN con USING : En caso de que los campos a reunir de las dos tablas se digan exactamente igual, podemos sustituir la condición puesta en ON para la expresión USING , con el campo de la reunión entre paréntesis

```
SELECT COMARQUES.
        nom_c,
        POBLACIONS.
        nom
FROM COMARCAS
        INNER JOIN POBLACIONES
        USING (nom_c)
```

NATURAL JOIN : También para el caso anterior, en el que el campo en las dos tablas se llama igual, podemos hacerlo de forma aún más abreviada. Hará una reunión, igualando todos los campos que se digan igual de las dos tablas. Debemos tener cuidado, por si hay algún otro campo en las dos tablas que se diga igual.

```
SELECT COMARQUES.
        nom_c,
        POBLACIONS.
        nom
FROM COMARCAS
        NATURAL JOIN POBLACIONES
```

Tres o más tablas

```
SELECT COMARQUES.nom_c,
        provincia,
        POBLACIONS.nom,
        INSTITUTS.nom
FROM (COMARCAS
        INNER JOIN POBLACIONES
        ON COMARQUES.nom_c = POBLACIONS.nom_c)
INNER JOIN INSTITUTOS
ON POBLACIONS.cod_m = INSTITUTS.cod_m
ORDER BY 1,3;
```

```
SELECT COMARQUES.nom_c,
        provincia,
        POBLACIONS.nom,
        INSTITUTS.nom
FROM COMARCAS,
        POBLACIONES,
        INSTITUTOS
WHERE COMARQUES.nom_c = POBLACIONS.nom_c
        AND POBLACIONS.cod_m = INSTITUTS.cod_m
ORDER BY 1,3;
```

Clave externa formada por más de un campo

```
SELECT C.n_ent,
        C.n_suc,
        n_cc,
        S.nom,
        C.saldo
FROM SUCURSAL S
        INNER JOIN COMPTE_CORRENT C
        ON S.n_ent = C.n_ent
        AND S.n_suc = C.n_suc

SELECT C.n_ent,
        C.n_suc,
        n_cc,
        S.nom,
        C.saldo
FROM SUCURSAL S,
        COMPTE_CORRENT C
WHERE S.n_ent = C.n_ent
        AND S.n_suc = C.n_suc
```

Combinación externa

Tendremos dos posibilidades: sacar todas las de la izquierda o sacar todas las de la derecha

```
SELECT ...
FROM tabla1
        LEFT [OUTER] JOIN Tabla2
        ON condición

SELECT POBLACIONS.nom,
        INSTITUTS.nom
FROM POBLACIONES
        LEFT JOIN INSTITUTOS
        ON POBLACIONS.cod_m = INSTITUTS.cod_m
ORDER BY 1


SELECT ...
FROM tabla1
        RIGHT [OUTER] JOIN Tabla2
        ON condición
```

Sintaxis en el FROM

```
SELECT ...
FROM ( Subconsulta ) AS Nom_Subconsulta
```

```
SELECT AVG (cuantos)
FROM
        (SELECT COUNT (*) AS cuantos
        FROM POBLACIONES
        GROUP BY nom_c) AS S;
```

Sintaxis en el WHERE o HAVING

```
SELECT ...
FROM Tabla
WHERE campo operador ( Subconsulta )
```

```
SELECT *
FROM POBLACIONES
WHERE altura =
        (SELECT MAX (altura)
        FROM POBLACIONES))
```

La subconsulta devuelve 34 valores (uno por cada comarca), y de esta manera no se puede comparar el valor de laizquierda del igual con los 34 valores de la derecha. Para solucionar el problema de cuándo vuelve más de un valor podemos utilizar los predicados ALL , ANY , SOME .
Si utilizamos ALL el resultado será cierto si la comparación es cierta con TODOS los valores que devuelve la subconsulta. Si utilizamos AÑO o SOME (que son sinónimos) el resultado será cierto si la comparación es cierta con ALGUNO valor de la subconsulta .

- Si utilizamos ALL el resultado será cierto si la comparación es cierta con TODOS los valores que devuelve la subconsulta.
- Si utilizamos AÑO o SOME (que son sinónimos) el resultado será cierto si la comparación es cierta con ALGUNO valor de la subconsulta .

```
SELECT *
FROM POBLACIONES
WHERE altura = ANY
        (SELECT MAX (altura)
        FROM POBLACIONES
        GROUP BY nom_c)
```

- **El operador IN** . No será problema de que la subconsulta devuelva un valor o muchos. La condición será cierta si el valor del campo (o de la expresión) está entre la lista de valores que devuelve la subconsulta. También pueden utilizar NOT IN, y la condición será cierta cuando el valor del campo no está en la lista.

  ```
  SELECT *
  FROM POBLACIONES
  WHERE cod_m NOT IN (
        SELECT cod_m
        FROM INSTITUTOS)
  ```

- **El operador EXISTS** . Es seguramente el más incómodo. No se compara un campo (o expresión) con la subconsulta, sino únicamente se pone [NOT] EXISTS (subconsulta) . La condición será cierta si la subconsulta devuelve alguna fila , y no será cierta si no vuelve ninguna fila

  ```
  SELECT *
  FROM POBLACIONES
  WHERE NOT EXISTS
        (SELECT *
        FROM INSTITUTOS
        WHERE cod_m = POBLACIONS.cod_m)
  ```

Sintaxis en el SELECT

```
SELECT ... ( Subconsulta )
FROM Tabla
```

```
SELECT nombre, altura, altura (
        SELECT AVG (altura)
        FROM POBLACIONES)
FROM POBLACIONES
```

Sintaxis de la UNIÓN

Al igual que en la unión de conjuntos, el resultado serán todas las filas de las dos (o más) consultas individuales, pero sin repetir filas, es decir, si de las dos consultas obtienen filas iguales, éstas sólo saldrán una vez. Lo anterior se puede evitar si ponemos el predicado ALL, y entonces sí que saldrán las filas repetidas.

```
[TABLE] consulta1
UNION [ALL]
[TABLE] consulta2 …
```

```
SELECT nom_c, provincia
FROM COMARCAS
UNION
SELECT nombre, provincia
FROM COMARCAS
INNER JOIN POBLACIONES USING (nom_c)
ORDER BY nom_c;
```

Sintaxis de la INTERSECCIÓN

Es idéntica a la unión, pero poniendo la palabra INTERSECT , y servirá para sacar únicamente las filas que están en las dos consultas.

```
[TABLE] consulta1
INTERSECT [ALL]
[TABLE] consulta2 …
```

```
SELECT nom_c, provincia
FROM COMARCAS
INTERSECT
SELECT nombre, provincia
FROM COMARCAS
        INNER JOIN POBLACIONES USING (nom_c)
ORDER BY nom_c
```

Sintaxis de la DIFERENCIA

Es idéntica a las anteriores, pero poniendo la palabra EXCEPT, y servirá para sacar las filas que están en la primera consulta pero que no están en la segunda.

```
[TABLE] consulta1
EXCEPT [ALL]
[TABLE] consulta2 …
```

```
SELECT nom_c, provincia
FROM COMARCAS
EXCEPT
SELECT nombre, provincia
FROM COMARCAS I
        NNER JOIN POBLACIONES USING (nom_c)
ORDER BY nom_c;
```