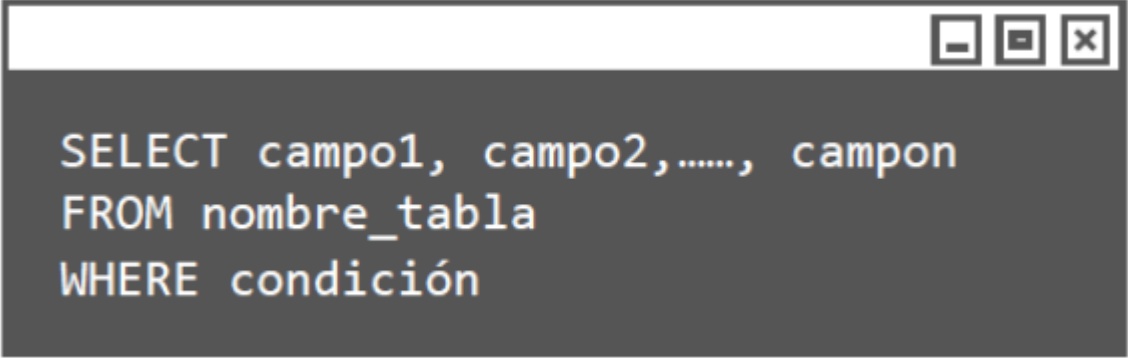


Consultas SQL

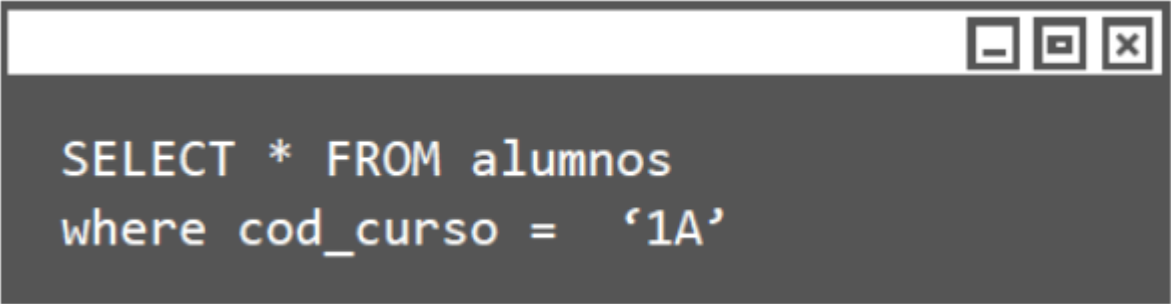
La operación básica para recuperar registros es la sentencia **SELECT** que permite obtener registros de una tabla origen, su sintaxis es la siguiente.

Sentencia SELECT



```
SELECT campo1, campo2,....., campon  
FROM nombre_tabla  
WHERE condición
```

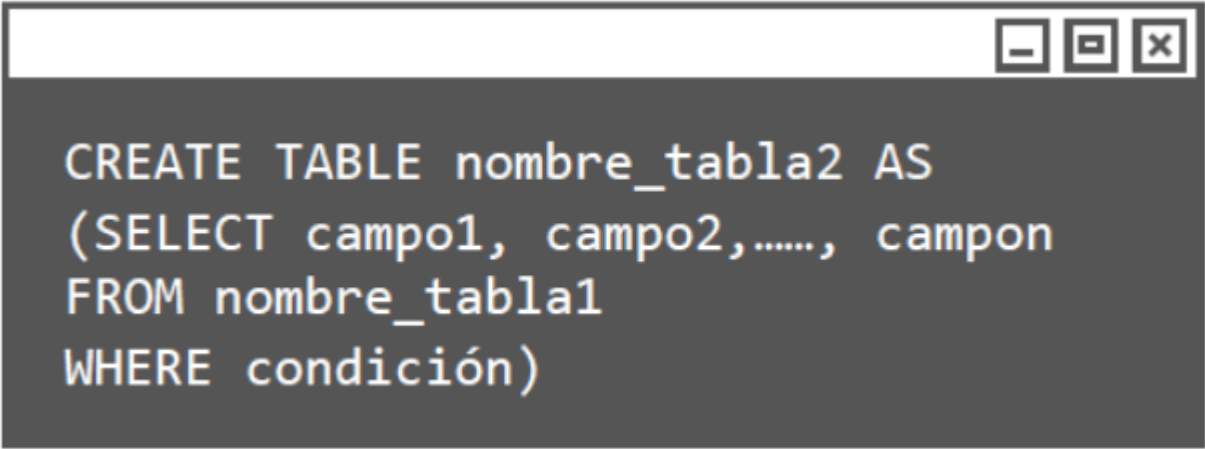
Ejemplo de select en tabla alumnos:



```
SELECT * FROM alumnos  
where cod_curso = '1A'
```

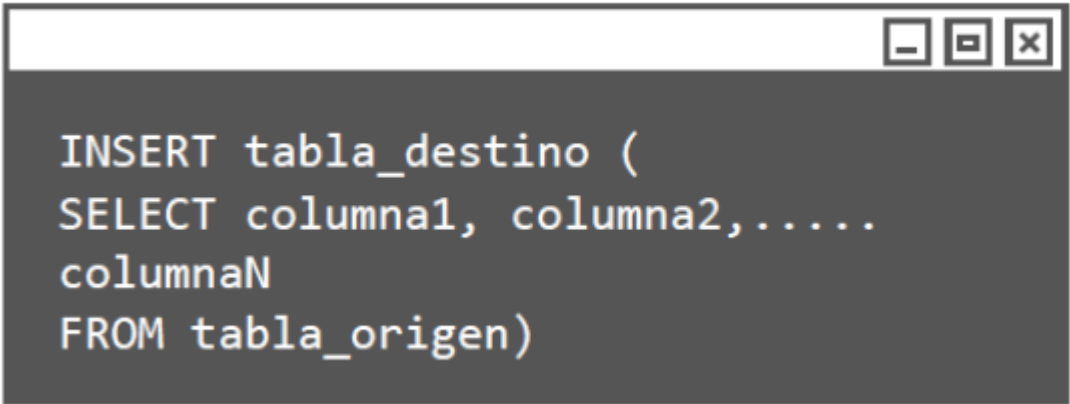
En la notación SQL con el símbolo ***** nos referimos a todos los campos de la tabla.

Podemos utilizar el resultado de una **SELECT** para crear una nueva tabla, del siguiente modo:



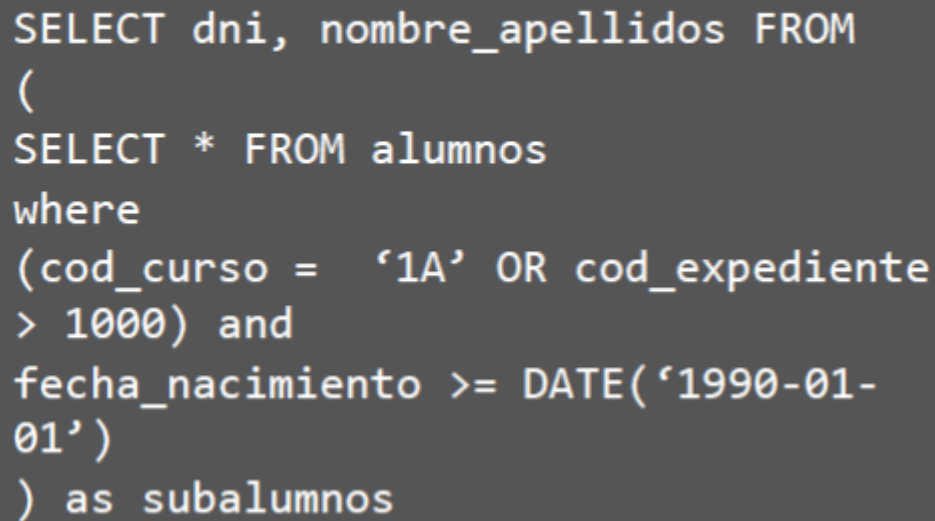
```
CREATE TABLE nombre_tabla2 AS  
(SELECT campo1, campo2,....., campon  
FROM nombre_tabla1  
WHERE condición)
```

Podemos combinar la sentencia **INSERT** con **SELECT** los que nos va a permitir insertar registros recuperados en una selección en una segunda tabla, la sintaxis es la siguiente:



```
INSERT tabla_destino (  
SELECT columna1, columna2,.....  
columnaN  
FROM tabla_origen)
```

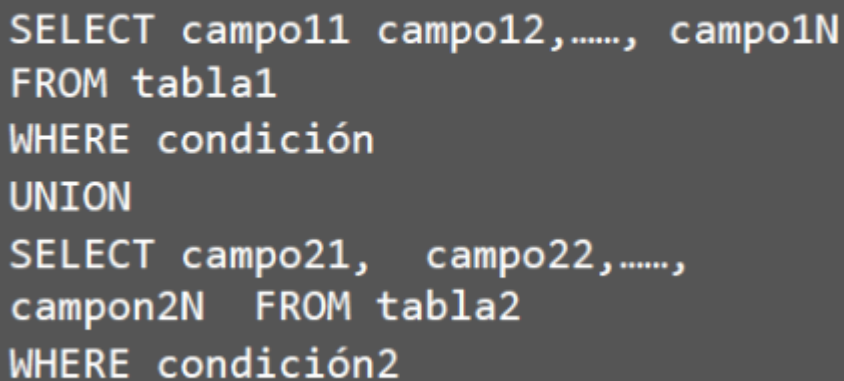
Ejemplo de subconsulta sobre tabla alumnos:

A screenshot of a terminal window with a dark background and light-colored text. The window has a title bar with three icons (minimize, maximize, close) on the right. The SQL query is as follows:

```
SELECT dni, nombre_apellidos FROM
(
SELECT * FROM alumnos
where
(cod_curso = '1A' OR cod_expediente
> 1000) and
fecha_nacimiento >= DATE('1990-01-
01')
) as subalumnos
```

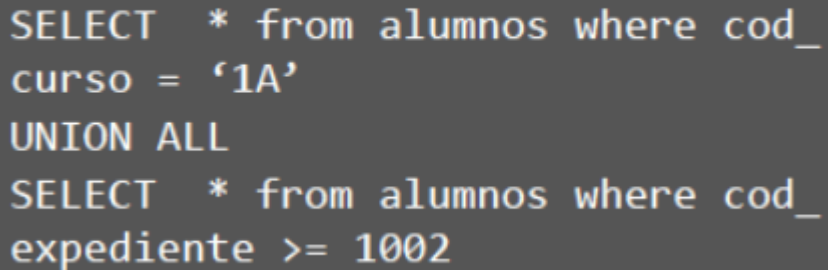
Cláusula UNION

Es posible también unir el resultado de dos select empleando el operador **UNION**:

A screenshot of a terminal window with a dark background and light-colored text. The window has a title bar with three icons (minimize, maximize, close) on the right. The SQL query is as follows:

```
SELECT campo11 campo12,....., campo1N
FROM tabla1
WHERE condición
UNION
SELECT campo21, campo22,.....,
campon2N FROM tabla2
WHERE condición2
```

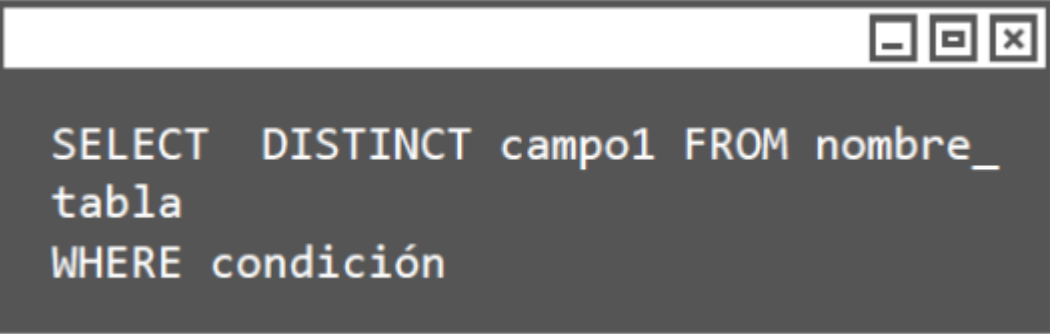
Este operador elimina duplicados, para permitir duplicados emplear **UNION ALL**.
Ejemplo de **UNION** en tabla alumnos:



```
SELECT * from alumnos where cod_
curso = '1A'
UNION ALL
SELECT * from alumnos where cod_
expediente >= 1002
```

Cláusula DISTINCT

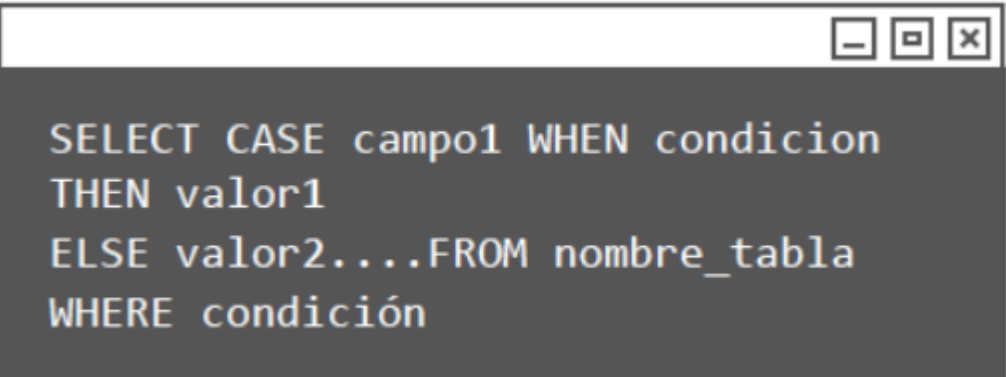
Para obtener los valores distintos de un campo determinado empleamos el modificador **DISTINCT**:



```
SELECT DISTINCT campo1 FROM nombre_
tabla
WHERE condición
```

Cláusula CASE

En una select podemos crear campos derivados en base a los obtenidos en la tabla origen. Una opción es utilizar el modificador **CASE**:



```
SELECT CASE campo1 WHEN condicion
THEN valor1
ELSE valor2....FROM nombre_tabla
WHERE condición
```

Filtrar resultados con WHERE

La cláusula **WHERE** permite incluir condiciones para filtrar registros en las operaciones SQL. La construcción de la condición se basa en el uso de operadores lógicos y de comparación.

Operadores lógicos

Permiten unir **condiciones**, los más habituales son:

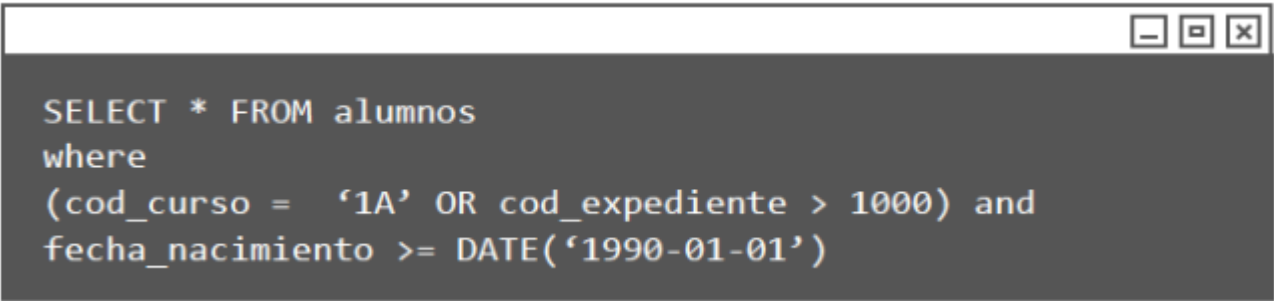
- **AND**: evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
- **OR**: evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
- **NOT**: negación lógica. Devuelve el valor contrario de la expresión.

Operadores de comparación

Permiten comparar valores, los más habituales son:

- **<** : menor.
- **>** : mayor.
- **<=** : menor o igual.
- **>=** : mayor o igual.
- **<>** : distinto.
- **=** : igual.
- **BETWEEN**: especifica un intervalo de valores.
- **LIKE**: compara respecto a un patrón.
- **IN**: especifica una lista de valores.
- **EXISTS**: devuelve TRUE si una subconsulta devuelve algún valor.
- **SOME/ANY**: compara un valor contra un conjunto de valores.

Ejemplo de filtro sobre registros en tabla alumnos:



```
SELECT * FROM alumnos
where
(cod_curso = '1A' OR cod_expediente > 1000) and
fecha_nacimiento >= DATE('1990-01-01')
```

Ordenar registros con ORDER BY

El resultado de una consulta SQL puede ser **ordenado** empleando la cláusula **ORDER BY**. Indicamos en la cláusula **ORDER BY** los campos sobre los cuales vamos a ordenar.

```
SELECT campo1, campo2,....., campoN FROM nombre_tabla ORDER BY campo1, campo2, ...campoM
```

La ordenación puede ser ascendente o descendente **ASC/DESC**. El valor por defecto es **ASC** (ascending).

Ejemplo de ordenación descendente sobre tabla alumnos:

```
SELECT * from alumnos ORDER BY nombre_apellidos desc
```

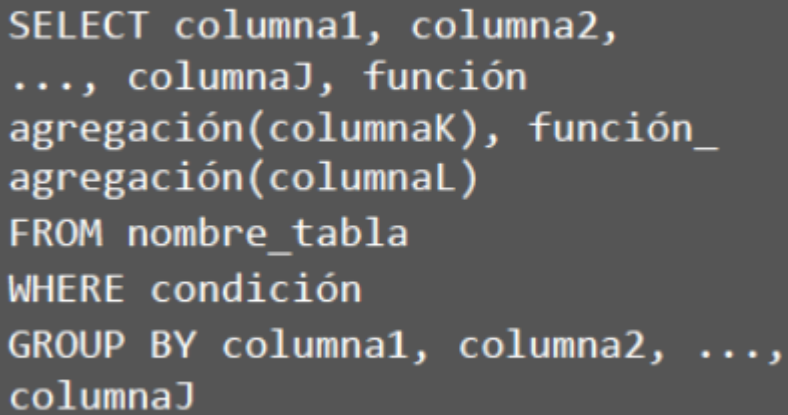
Agregación de resultados con GROUP BY

La sentencia **GROUP BY** permite realizar agregaciones en base a la clave indicada y realizar determinadas operaciones sobre los registros agrupados.

Algunas de las operaciones a realizar son las siguientes:

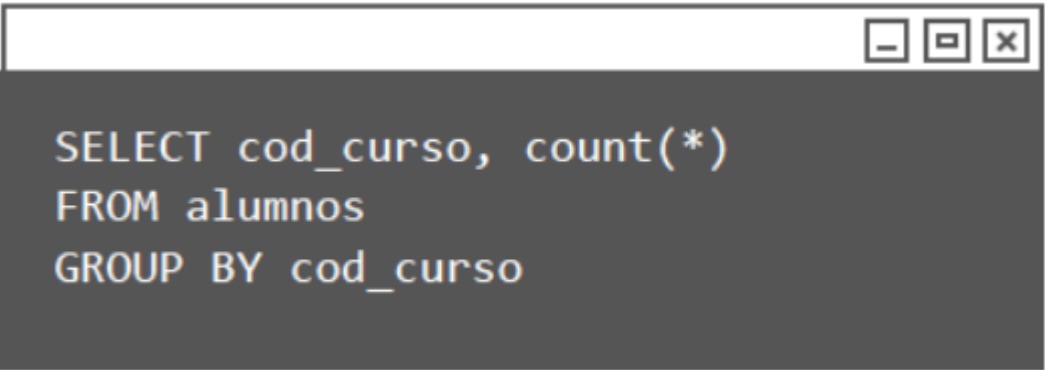
- **COUNT**: cuenta el total de elementos de un grupo.
- **SUM**: suma los valores numéricos de los registros agrupados.
- **MIN**: muestra el valor más pequeño de un grupo.
- **MAX**: muestra el valor máximo de todo el conjunto de registros agrupados.
- **AVERAGE**: calcula la media de todos los valores seleccionados.

La sintaxis es la siguiente:

A screenshot of a SQL query window with a dark background and a light gray title bar containing standard window controls (minimize, maximize, close). The query is written in a light-colored monospace font.

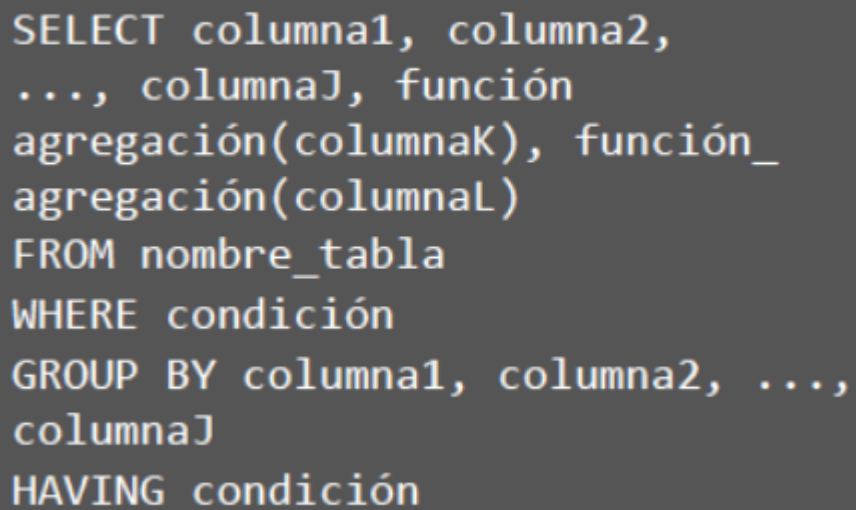
```
SELECT columna1, columna2,  
..., columnaJ, función  
agregación(columnaK), función_  
agregación(columnaL)  
FROM nombre_tabla  
WHERE condición  
GROUP BY columna1, columna2, ...,  
columnaJ
```

Ejemplo sobre la tabla alumnos:

A screenshot of a SQL query window with a dark background and a light gray title bar containing standard window controls (minimize, maximize, close). The query is written in a light-colored monospace font.

```
SELECT cod_curso, count(*)  
FROM alumnos  
GROUP BY cod_curso
```

La sentencia **HAVING** combinada con la sentencia **GROUP BY** nos permite incluir condiciones que filtren la salida sobre la agregación definida en el **GROUP BY**.



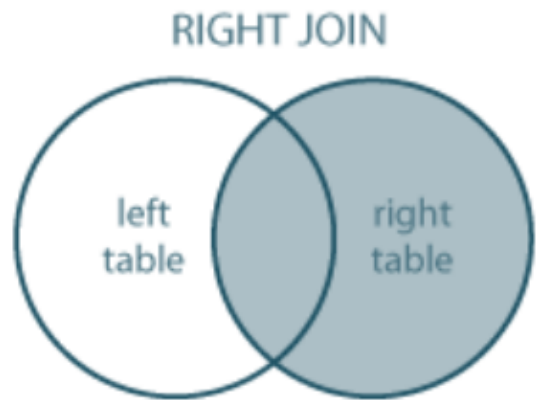
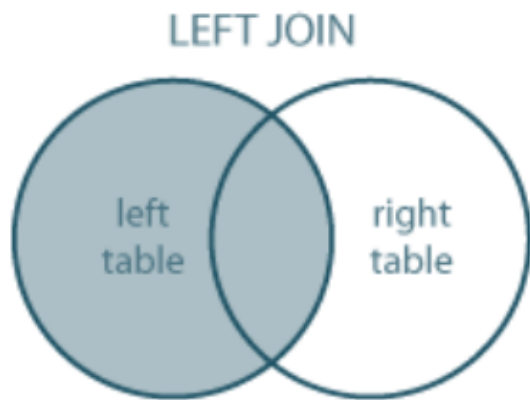
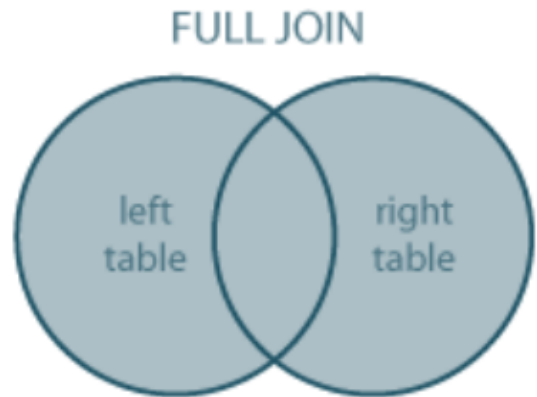
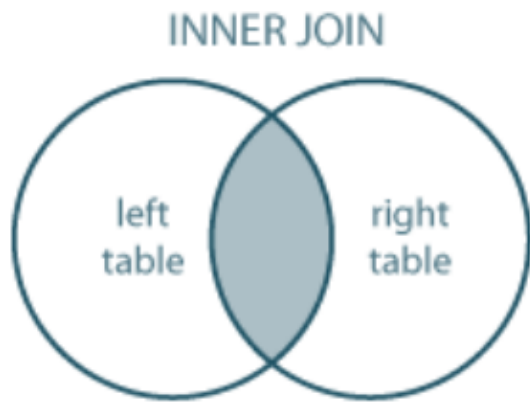
```
SELECT columna1, columna2,  
..., columnaJ, función  
agregación(columnaK), función_  
agregación(columnaL)  
FROM nombre_tabla  
WHERE condición  
GROUP BY columna1, columna2, ...,  
columnaJ  
HAVING condición
```

Cruzar tablas con JOIN

La sentencia **SELECT** permite obtener información de varias tablas.

Para ello se pueden definir cruces entre tablas en los que seleccionamos campos de diferentes tablas.

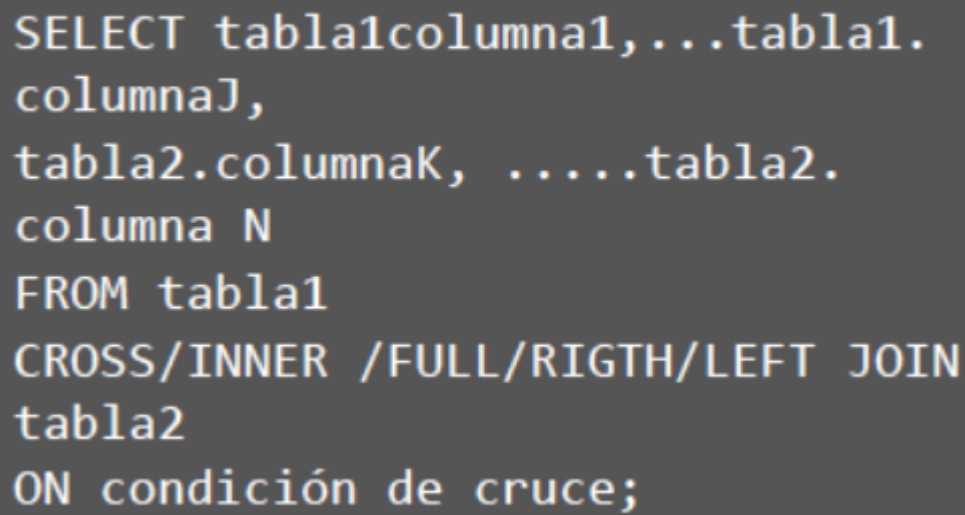
Es necesario definir los campos y las condiciones de cruce.



Tenemos los siguientes tipos de **JOIN**:

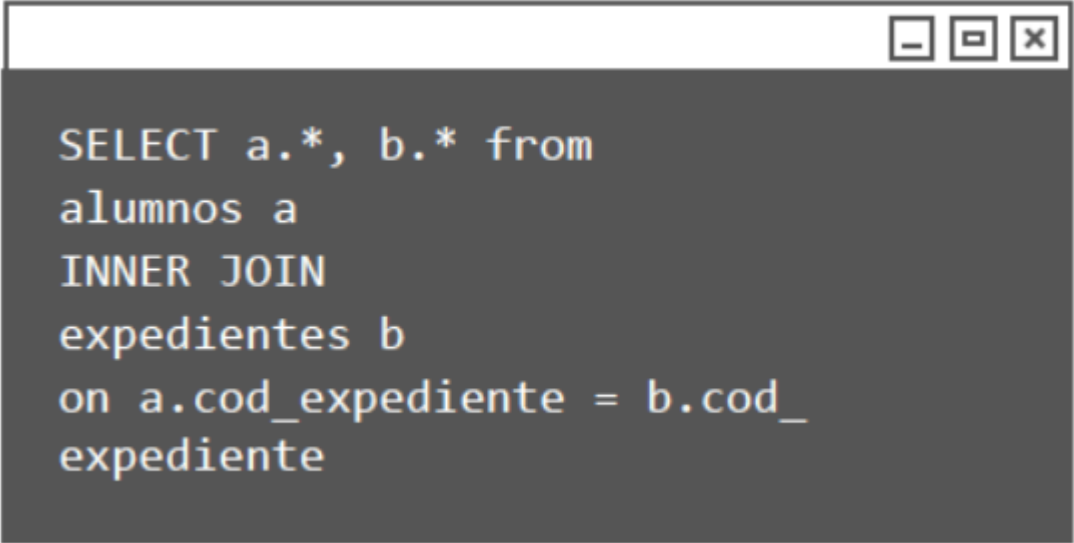
- **CROSS JOIN**: realiza un producto cartesiano entre las dos tablas.
- **INNER JOIN**: cada registro en la tabla **A** es combinado con cada registro de la tabla **B**; pero sólo permanecen aquellos registros en la tabla combinada que satisfacen las condiciones que se especifiquen.
- **FULL JOIN**: esta operación presenta los resultados de tabla izquierda y la tabla derecha, aunque no tengan correspondencia en la otra tabla. La tabla combinada contendrá, entonces, todos los registros de ambas tablas y presentará valores nulos para registros sin pareja.
- **LEFT JOIN**: el resultado de esta operación siempre contiene todos los registros de la tabla de la izquierda (la primera tabla que se menciona en la consulta), aun cuando no exista un registro correspondiente en la tabla de la derecha para uno de la izquierda.
- **RIGHT JOIN**: idéntico al anterior, pero permanecen los registros de la tabla de la derecha.

La sintaxis general es la siguiente. Debemos indicar los campos que obtenemos de cada tabla, el tipo de cruce, así como los campos de cruce para unir ambas tablas.

A screenshot of a SQL query window with a dark background and a light gray title bar containing standard window controls (minimize, maximize, close). The query is written in a light-colored monospace font.

```
SELECT tabla1columna1,...tabla1.
columnaJ,
tabla2.columnaK, .....tabla2.
columna N
FROM tabla1
CROSS/INNER /FULL/RIGTH/LEFT JOIN
tabla2
ON condición de cruce;
```

Ejemplos de cruces entre alumnos y expedientes:

A screenshot of a SQL query window with a dark background and a light gray title bar containing standard window controls (minimize, maximize, close). The query is written in a light-colored monospace font.

```
SELECT a.*, b.* from
alumnos a
INNER JOIN
expedientes b
on a.cod_expediente = b.cod_
expediente
```

Los que están en tabla alumnos y no en expedientes:



```
SELECT a.*, b.* from  
alumnos a  
LEFT JOIN  
expedientes b  
ON a.cod_expediente = b.cod_  
expediente  
WHERE b.cod_expediente is null
```