SELECT

> Permite consultar filas de una o más tablas.

- Al seleccionar consultas de dos tablas, se obtiene el producto cartesiano entre ellos
- Parte del Data Manipulating Language de SQL

```
SELECT col1 <alias>, col2 AS <alias>, ….coln
FROM tabla --Lo primero que hay que definir

-- Optativos
WHERE condicion
GROUP BY col1, col2, …coln
HAVING condicion
ORDER BY col1, col2, …coln
```

WHERE

> Cláusula en la que se identifica la condición que deben cumplir las filas para ser devueltas en la consulta

- Atrás del `WHERE` va un `True` o `False`

```
SELECT col1, col2, ….coln
FROM tabla
WHERE cond1(=,<,>,<>) AND cond2 OR cond3 NOT cond4 LIKE ‘string_%
-- _% son comodines: _ representa una letra, y % varios
```

Funciones (más usadas)

```
YEAR(date)  MONTH(date)  DAY(date) GETDATE()
ISNULL(columna, valor)
LTRIM(string)  RTRIM(string)  TRIM(string)  SPACE(entero)
LEFT(string, n)  RIGHT(string, n)
CHAR(int) LOWER(string) UPPER(string)
```

```
SUBSTR(string, inicio, long)  LEN(string) CONCAT(strin1, string2,
REPLICATE(string, entero)  STR(numero, long, decimal)
```

GROUP BY

> Permite agrupar los resultados por las columnas definidas en ella

- Permite juntar muchas filas en una según la igualdad del valor de una (o varias) columnas
- Símil subtotales excel
- Toda columna del `SELECT` que no sea un campo calculado debe estar en el `GROUP BY`
- Siempre va *después* del `WHERE`

```
SELECT col1, col2, ….coln
FROM tabla
GROUP BY col1, col2
```

→ Funciones de grupo

> Funciones propias del DBMS (Database Management System) que solo pueden utilizarse conjuntamente con la cláusula `GROUP BY`, salvo que no haya nada calculado (por ejemplo al usar un `COUNT(*)`)

- `COUNT()`: `*` cuenta todas las filas, al darle una columna contará a las que sean verdadero
- `SUM()`
- `MIN()`
- `MAX()`
- `AVG()`

```
SELECT COUNT(*)
FROM tabla
GROUP BY col1
```

→ DISTINCT

#todo

→ HAVING

> Cláusula en la que se identifica la condición que deben cumplir las filas para ser devueltas en la consulta tras ser agrupadas en un `GROUP BY`

- Adjetivador del `GROUP BY`
- Símil `WHERE`, pero para `GROUP BY`
- **Filtra a nivel grupo**
- Va seguido de algo que devuelva `True` o `False`

```
SELECT col1, col2, ….coln
FROM tabla
GROUP BY col1
HAVING cond1(=,<,>,<>) AND cond2 OR cond3 NOT cond4 LIKE ‘string’
```

ORDER BY

> Cláusula en la que se detalla el orden en el que se visualizarán las filas obtenidas en la consulta

- Independiente de todos los comandos anteriores
- Va al final de la sentencia
- Es posible poner `ORDER BY 1,2` para ordenar según las primeras dos columnas del `SELECT`, independientemente de su nombre

```
SELECT col1, col2, ….coln
FROM tabla
ORDER BY col3, col1
```

JOIN

> Cláusula que permite unir dos o más tablas para acotar el producto cartesiano del `FROM`.

- `JOIN` es lo mismo que `INNER JOIN`, solo deja aquellas filas en las que exista la igualdad
- `LEFT JOIN` le da prioridad a la tabla a la izquierda; si no está en la tabla derecha lo trae al menos una vez. `RIGHT JOIN` hace lo mismo, pero priorizando la tabla a la derecha

```sql
SELECT * FROM tabla1 JOIN tabla2 ON tabla1.col1 = tabla2.col2

SELECT * FROM tabla1 LEFT JOIN tabla2 ON tabla1.col1 = tabla2.col

SELECT * FROM tabla1 RIGHT JOIN tabla2 ON tabla1.col1 = tabla2.co
```

- Al hacer un `SELECT * FROM tabla1 JOIN tabla2...` aparecerán todas las columnas de la tabla 1 y de la tabla 2
- `<LEFT/RIGHT> JOIN` se usará seguramente en todos los ejercicios que pidan "para todo..."
- Para realizar un `JOIN` con tablas on varias PK puede resultar útil concatenarlas (en el caso que sean tipo `CHAR(n)`)

→ Explicitud

- **Implícito**: condición en el `WHERE` donde se igualan dos columnas de tablas distintas
- **Explícito**: como está especificado debajo mediante el uso de la palabra `JOIN` (es más prolijo de esta manera, pero igual de performante)

CASE

> Permite establecer una selección múltiple dentro de un `SELECT`.

- Se realiza solo la expresión vinculada a la primera condición verdadera, luego se sale del `CASE` y no se validan las siguientes

```sql
-- Con valor de columna
CASE columna WHEN valor1 THEN exp1 WHEN valor2 THEN valor2 WHEN v

-- Con condición (pueden ir columnas distintas en las condiciones
CASE WHEN condicion1 THEN exp1 WHEN condicion2 THEN exp2 WHEN con
```

TOP

> Permite devolver solo una determinada cantidad de filas desde el inicio.

- No es ANSI, es propio de SQL SERVER

```
SELECT TOP n col1, col2, col3 FROM tabla1
ORDER BY col1
```

Subconsultas

> La subconsulta es uno o varios `SELECT` dentro de otro.

Un comando `SELECT` puede colocarse dentro de otro en cualquiera se sus cláusulas `SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY`. En función de donde se incorpore hay condiciones que cumplir.

- Si bien es posible, **no** usar una subconsulta dentro de un `FROM`
- Las usamos para proteger nuestro universo (para que no se vea alterado)
- Deben ser utilizadas como último recurso
- Se conoce como subselect **dinámico** a aquel que cada vez que se ejecuta cambia su resultado (depende del select de afuera)

→ Subconsulta en `SELECT`

```
SELECT col1, (SELECT col5 FROM tabla2 WHERE col3=tablas1.col1)
FROM tabla1
```

- Siempre devolver un **escalar** (conjunto conformado por **una fila y una columna**)
- Si el subselect devuelve más de un valor en cualquiera de sus dimensiones se arroja error
- Se puede igualar algo de dentro con algo de afuera

→ Subconsulta en `WHERE`

La subconsulta debe devolver cualquier expresión que pueda ser utilizada en una condición.

```
SELECT col1, col2 FROM tabla1
WHERE col1 = (SELECT col3 FROM tabla2 WHERE col3 = tablas1.col1)

SELECT col1, col2 FROM tabla1
WHERE col1 IN (SELECT col3 FROM tabla2)
-- IN evalúa intersección: algo intersección "no rows" = True
```

```
SELECT col1, col2 FROM tabla1
WHERE EXISTS (SELECT col3 FROM tabla2 WHERE col1=col3)
-- Evalúa pertenencia: algo EXISTS "no rows" = False (Se usa cuan
```

→ Subconsulta en `GROUP BY`

Siempre debe devolver un escalar

```
SELECT col1, col2 FROM tabla1
GROUP BY (SELECT col5 FROM tabla2 WHERE col3=tablas1.col1)
```

→ Subconsulta en `HAVING`

Puede devolver cualquier expresión que pueda usarse en una condición

```
SELECT col1, col2 FROM tabla1
GROUP BY col1
HAVING col1 = (SELECT col3 FROM tabla2 WHERE col3 = tablas1.col1)

SELECT col1, col2 FROM tabla1
GROUP BY col1
HAVING col1 IN (SELECT col3 FROM tabla2)
```

→ Subconsulta en el `ORDER BY`

Debe siempre devolver un escalar

```
SELECT col1, col2 FORM tabla1
ORDER BY (SELECT col5 FROM talba2 WHERE col3 = tablas1.col1)
```

→ X Subconsulta en el `FROM` X

Puede devolver cualquier conjunto generado por un select

```
SELECT col1, col3 FROM (SELECT col1, col3 FROM tabla1, tabla2
WHERE col3=tablas1.col1)
```

X **No se permite** el uso de un subselect en el `FROM` debido a la pérdida de performance que ocasiona

UNION

> Cláusula que permite unir varios resultados de `SELECT` distintos

- El resultado es uno -> Solo puede llevar un `ORDER BY` al final (y no en cada subselect)
- Deben devolver la misma cantidad de columnas
- Deben devolver columnas del mismo tipo

```
-- Unión disyuntiva
SELECT col1, col2 FROM tabla1
UNION
SELECT col3, col4 FROM tabla2

-- Unión "completa"
SELECT col1, col2 FROM tabla1
UNION ALL
SELECT col3, col4 FROM tabla2
```

- `UNION` representa una **unión disyuntiva** (si dos filas son iguales entre ambos `SELECT`, muestra solo una)
- `UNION ALL` muestra aquellas filas con valores repetidos