Tema 10

Subconsultas

Un consulta dentro de otra.

Clasificación según el resultado:

- Valor escalar: devuelven un único valor. P.ej: `SELECT max(salario) FROM empleado`.
- De fila: una única fila con uno o más valores. P.ej: `SELECT max(salario), min(salario) FROM empleado`.
- De tabla: pueden devolver más de una fila[1] P.Ej: `SELECT apellido, nombre FROM empleado`.

Pueden usarse de varias formas:

1. Dentro de la selección.
2. Dentro de la sección de origen de los datos (`FROM`).
3. Dentro del filtrado:
   - De filas: `WHERE`.
   - De grupos: `HAVING`.

Dentro de la selección tienen menos utilidad y suele hacerse para usar agregados:

[1] No importa cuántas filas devuelvan en realidad con este conjuntos de datos, sino si con otros datos, podrían devolver varias filas.

```
SELECT
   nombre, apellido, salario,
   ROUND(salario -
      (SELECT  AVG(salario) FROM empleados))
         AS 'Variación Salario'
FROM
   empleados;
```

Dentro de `FROM` sirve para realizar una tabla temporal sobre la que obtener los datos requeridos.

Aunque he usado tres alias para mejorar la lectura, el **único obligatorio** es el de la subconsulta para todas las tablas (incluidas las temporales) deben tener un nombre.

```
SELECT
   MAX(media_salario) AS media
FROM
   (SELECT
      AVG(salario) AS media_salario
   FROM
      empleados
   GROUP BY id_departamento) AS media_sal_depart;
```

10.1. Subconsultas escalares.

La forma más sencilla es usarla para comparar:

```
/* Empleados que cobran más que la media */
SELECT * FROM empleados WHERE
   salario > (SELECT AVG(salario) FROM empleados);

/* Obtener el departamento del empleado que más cobra */
SELECT id_departamento FROM empleados WHERE
   salario = (SELECT MAX(salario) FROM empleados);
```

Nota

Los casos que vemos en los ejemplos siguientes pueden aplicarse al resto de apartados de este tema.

Tablas diferentes en consulta y subconsulta:

```
/* Nombre del departamento con id más alto entre
los que tienen empleados. */
SELECT
   nombre_departamento
FROM
   departamentos
WHERE
   id_departamento = (SELECT
         MAX(id_departamento)
      FROM
         empleados);
```

E incluso una subconsulta puede contener otra:

```
/* Nombre del departamento del empleado que menos cobra */
SELECT
   nombre_departamento
FROM
   departamentos
WHERE
   id_departamento = (SELECT
         id_departamento
      FROM
         empleados
      WHERE
         salario = (SELECT
               MIN(salario)
            FROM
               empleados));
```


10.2. Subconsultas de fila.

Podemos igualar **por más de un campo** si queremos obtener datos más concretos.

Por ejemplo si queremos ver que el empleado que más cobra está en el departamento con menor id.

Ten en cuenta que al aplicarse un AND a los dos campos, puede no devolver ninguna fila si no hay un empleado que cumpla ambas condiciones:

```
SELECT
   nombre, apellido
FROM
   empleados
WHERE
   (id_departamento , salario) = (SELECT
      MIN(id_departamento), MAX(salario)
   FROM
      empleados);
```

Nota

Con los operadores `=`, `<>` (o `!=`), `>`, `<`, `>=` y `<=`, `BETWEEN` y `NOT BETWEEN` la subconsulta debe devolver un **único valor** (escalar) o **una única fila**.

10.2.1. BETWEEN y NOT BETWEEN.

```sql
/* Empleados que cobran más que el que más cobra de Marketing y
menos que el que menos cobra de Ejecutivo */
SELECT
   *
FROM
   empleados
WHERE
   salario BETWEEN (SELECT
         MAX(salario)
      FROM
         empleados E,
         departamentos D
      WHERE
         E.id_departamento = D.id_departamento
            AND nombre_departamento = 'Marketing')
   AND (SELECT
         MIN(salario)
      FROM
         empleados E,
         departamentos D
      WHERE
         E.id_departamento = D.id_departamento
            AND nombre_departamento = 'Ejecutivo');
```

Ejercicios 10.1

1. Selecciona todos los empleados que tenga el departamento de marketing.
2. Selecciona la dirección de la localización con el código postal más alto.
3. Selecciona el nombre del trabajo con mayor diferencia entre el salario más alto y el más bajo.
4. Selecciona todos los empleados que tengan como director aquel con id más bajo.
5. Selecciona todos los empleados que pertenezcan a un departamento con un id mayor que la media.
6. Selecciona el nombre del empleado que más cobra del departamento de ventas.
7. Selecciona el nombre del departamento al que pertenezca el empleado con id más alto.
8. Devuelve el nombre del departamento al que pertenece el empleado que más

cobra.

9. Devuelve la dirección y la ciudad de las localizaciones en las que haya al menos dos departamentos.
10. Devuelve la dirección, la ciudad y el nombre del país de las localizaciones en las que haya al menos dos departamentos.
11. Devuelve el identificador de los departamentos cuya media de salarios sea mayor que la media de salarios de la empresa.
12. Devuelve el nombre de los departamentos cuya media de salarios sea mayor que la media de salarios de la empresa.
13. Obtén todos los empleados con salarios entre el máximo y el mínimo para el trabajo IT_PROG.

10.3. Subconsultas que pueden devolver más de una fila (de tabla).

Se utilizan instrucciones especiales entre el operador y la consulta:

- `ANY`.
- `ALL`.
- `IN`.
- `NOT IN`.
- `EXISTS`.
- `NOT EXISTS`.

10.3.1. `ALL` y `ANY`.

```
/* Empleado que menos cobra usando ALL*/
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario <= ALL (SELECT
         salario
      FROM
         empleados);
```

```
/* Todos los empleados excepto el que menos cobra */
SELECT
   nombre, apellido, salario
FROM
   empleados
WHERE
   salario > ANY (SELECT
         salario
      FROM
         empleados);
```

10.3.2. `IN` y `NOT IN`.

```
/* localizaciones en las que no hay departamentos */
SELECT
   direccion, ciudad
FROM
   localizaciones
WHERE
   id_localizacion NOT IN (SELECT
         id_localizacion
      FROM
         departamentos);
```



```sql
/* Obtener el id de trabajo y el de departamento del empleado
que más cobra de cada departamento */
SELECT
   id_trabajo, id_departamento
FROM
   empleados
WHERE
   (id_departamento , salario) IN (SELECT
         id_departamento, MAX(salario)
      FROM
         empleados
      GROUP BY id_departamento);
```

10.3.3. `EXISTS` y `NOT EXISTS`.

```sql
/* Empleados que han tenido trabajos previos en la empresa */
SELECT
   *
FROM
   empleados E
WHERE
   EXISTS( SELECT
         *
      FROM
         historial_trab H
      WHERE
         E.id_empleado = H.id_empleado);
```

Nota

¡Esta consulta es correlacionada!

10.4. Subconsultas correlacionadas.

Subconsultas que contienen una o varias correlaciones entre sus columnas y columnas de la consulta externa.

```
/* Empleados cuyo salario no se encuentre entre los límites definidos
   en la tabla trabajos para su puesto. */
SELECT
   nombre, apellido
FROM
   empleados
WHERE
   salario NOT BETWEEN
      (SELECT min_salario FROM trabajos WHERE
         empleados.id_trabajo = trabajos.id_trabajo)
      AND
      (SELECT max_salario FROM trabajos WHERE
         empleados.id_trabajo = trabajos.id_trabajo);
```

10.5. ¿Dónde usar subconsultas?

Las subconsultas se pueden usar en las siguientes sentencias:

- SELECT.
- INSERT.
- DELETE.
- UPDATE.
- CREATE TABLE AS.
- INSERT INTO.
- SELECT INTO.

10.5.1. Inserción, borrado y modificación usando subconsultas.

Las subconsultas se puede utilizar para modificar datos de tablas.

En estos casos es muy importante probar la subconsulta antes de ejecutar la consulta.

10.5.1.1. Inserción.

Supongamos que los empleados del departamento de ventas se van de viaje y necesitan tener una lista con determinados datos.

NOTA: lo que vamos a hacer aquí se haría con vistas tal y como veremos más adelante porque así estamos introduciendo redundancia de datos.

Para ello creamos una tabla específica.

```
CREATE TABLE viaje (
   id_viajero INT NOT NULL,
   nombre VARCHAR(20),
   apellido VARCHAR(20) NOT NULL,
   correo VARCHAR(25) NOT NULL,
   num_telefono VARCHAR(20),
   id_director INT,
   CONSTRAINT viaje_pk PRIMARY KEY (id_viajero),
   CONSTRAINT viaje_fk1 FOREIGN KEY (id_director)
      REFERENCES empleados(id_empleado)
);
```

Luego deberíamos probar la consulta con la que obtendríamos estos datos:

```
SELECT
   id_empleado, nombre, apellido, email, telefono, id_director
FROM
   empleados
WHERE
   id_departamento = (SELECT
         id_departamento
      FROM
         departamentos
      WHERE
         nombre_Departamento = 'Ventas');
```

Y posteriormente la usamos en el `INSERT` como subconsulta:


```
INSERT INTO viaje (id_viajero, nombre, apellido, correo,
   num_telefono, id_director)
SELECT
   id_empleado, nombre, apellido, email, telefono, id_director
FROM
   empleados
WHERE
   id_departamento = (SELECT
         id_departamento
      FROM
         departamentos
      WHERE
         nombre_Departamento = 'Ventas');
```

10.5.1.2. Modificación.

Por ejemplo, queremos premiar aquellos empleados que han cumplido más de una función en la empresa:

```
UPDATE empleados e
SET
   salario = (salario * 1.25)
WHERE
   EXISTS( SELECT
         *
      FROM
         historial_trab h
      WHERE
         e.id_empleado = h.id_empleado);
```

10.5.1.3. Borrado.

La empresa ha decidido prejubilar a todos los empleados que empezaron a trabajar antes del año 2000 y pertenezcan al departamento *Envíos*.

```sql
DELETE FROM empleados
WHERE
   id_departamento = (SELECT
      id_departamento
   FROM
      departamentos
   WHERE
      nombre_departamento = 'Envíos')
   AND fecha_contratacion < '2000-01-01';
```

En este último caso, la consulta no se puede realizar por las restricciones sobre modificaciones y borrado.

Ejercicios 10.2

1. Selecciona la localización con el código postal más alto. Haz la consulta de, al menos, dos formas diferentes. (`ALL` y `MAX`).
2. Selecciona las localizaciones en las que haya algún departamento usando `ANY`. ¿Puedes hacerlo sin subconsultas?.
3. Obtén los empleados que tengan como jefe a un director cuyo apellido empiece por "C". Usa `IN`.
4. Obtén los empleados cuyo jefe (`id_director`), no trabaja en el mismo departamento que ellos. Usa `EXISTS`.
5. Selecciona aquellos departamentos en los que su director no trabaja en él. Usa `NOT EXISTS`.
6. Crea una tabla con dirección, código postal, ciudad, provincia, nombre del país y nombre de la región y llénala con los datos de las localizaciones de las regiones Europa y Asia usando una subconsulta.
7. Actualiza el salario y la comisión de aquellos empleados que tengan comisión. El salario disminuirá en un 10 % y la comisión subirá un 20 %. Ten en cuenta que la comisión ya es un porcentaje por lo que se pide que si un empleado tiene una comisión de 40 % se el incremente hasta el 60 %.

   Hazlo usando una subconsulta y sin usarla.
8. Elimina los empleados que trabajen en el departamento de *Ventas* y tengan el puesto de *Representante de ventas*.

   Hazlo usando una subconsulta y sin usarla.

10.6. Bibliografía.

- José Luis Comesaña (2011). Curso MEFP-IFCS03-BD. Ministerio de Educación y Formación Profesional.
- Structured Query Language. Wikibook. Wikipedia.
- Curso de MySQL. Con Clase.
- MySQL 8.0 Reference Manual.
- José Juan Sánchez Hernández (2021/2022). Bases de datos / Gestión de Bases de datos
- Iván López Montalbán, Manuel de Castro Vázquez, John Ospino Rivas (2022). Bases de Datos (2ª Edición) . Garceta.
- Subconsultas. IBM PureData System for Analytics, Version 7.1.
- Subconsultas correlacionadas. Amazon Redshift. Guía para desarrolladores de bases de datos.
- How to Use Subqueries in INSERT, UPDATE, and DELETE Statements. Ignacio L. Bisso (LearnSQL).

