Tema 3: *Queries*

SELECT

Como ya vimos en la subcapsula anterior, para consultar los datos de una tabla se utiliza la sentencia SELECT.

Con SELECT podemos obtener los datos de todos los campos de la tabla, por ejemplo:

```
SELECT * FROM profesor;
```

Esto nos devolverá todos los datos que tenga la tabla en todas sus columnas.

Tabla 24: Operación SELECT para datos de todas las columnas

	id	nombre	apellido	fecha_nacimiento	salario
▶	1	Juan	Pérez	1990-06-06	55000
	2	María Emilia	Paz	1984-07-15	72000
	3	Martín	Correa	1987-12-07	63000
	4	Lucía	Díaz	1991-02-24	45000
	5	Raúl	Martínez	1980-10-15	85000
	6	Mabel	Ríos	1982-06-12	83000

Fuente: elaboración propia

En algunos casos, no es necesario o no interesan todas las columnas por una cuestión de visibilidad (tener en cuenta que una tabla puede tener **muchas** columnas), para esos casos, especificamos cuáles son las columnas que nos interesan. Por ejemplo:

```
SELECT apellido, nombre, fecha_nacimiento FROM profesor;
```

Esto retornará en la siguiente tabla.

Tabla 25: Operación SELECT para datos de algunas columnas

	apellido	nombre	fecha_nacimiento
▶	Pérez	Juan	1990-06-06
	Paz	María Emilia	1984-07-15
	Correa	Martín	1987-12-07
	Díaz	Lucía	1991-02-24
	Martínez	Raúl	1980-10-15
	Ríos	Mabel	1982-06-12

Fuente: elaboración propia

También se puede especificar un orden distinto:

```
SELECT nombre, apellido, fecha_nacimiento FROM profesor;
```

Tabla 26: Operación SELECT para especificar un orden distinto

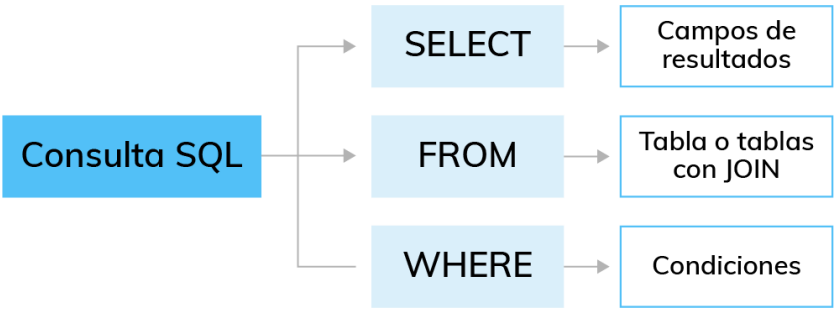
	nombre	apellido	fecha_nacimiento
▶	Juan	Pérez	1990-06-06
	María Emilia	Paz	1984-07-15
	Martín	Correa	1987-12-07
	Lucía	Díaz	1991-02-24
	Raúl	Martínez	1980-10-15
	Mabel	Ríos	1982-06-12

Fuente: elaboración propia

WHERE

En una consulta SQL, podemos distinguir, en principio, tres partes, tal como lo muestra el gráfico a continuación.

Figura 6: Consulta SQL



Fuente: elaboración propia

En las secciones anteriores, vimos un poco de las dos primeras partes. Esta sección está dedicada a la tercera parte de una consulta: el WHERE.

El WHERE sirve para agregar filtros al resultado. Por ejemplo, al ejecutar SELECT * FROM CURSO se obtendría lo que se muestra en la próxima tabla.

Tabla 27: Ejecutar SELECT * FROM CURSO

codigo	nombre	descripcion	cupo	turno	PROFESOR_id
101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1
102	Matemática Discreta	NULL	20	Tarde	2
103	Programación Java	POO en Java	35	Noche	4
104	Programación Web	NULL	35	Noche	5
105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

Fuente: elaboración propia

Esa consulta no tiene WHERE.

Pero puede pasar que solo necesitemos ver los cursos cuyo cupo sea mayor a 25. Eso es posible gracias a los filtros (o condiciones) que se agregan en el WHERE.

```
SELECT *
FROM CURSO
WHERE CUPO > 25;
```

Con esa consulta obtendríamos el resultado deseado.

Tabla 28: Operación WHERE

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	103	Programación Java	POO en Java	35	Noche	4
	104	Programación Web	NULL	35	Noche	5
	105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

Fuente: elaboración propia

Para agregar condiciones, se pueden utilizar los operadores matemáticos: =, <, >, >=, <=, etc. Además, existen otras palabras reservadas para agregar filtros al resultado.

LIKE

LIKE permite filtrar “valores similares” en un campo texto. Es útil cuando solo conocemos una parte del valor que queremos filtrar, pero no sabemos cómo es en su totalidad, es decir, que hay caracteres desconocidos.

Por ejemplo, si queremos obtener todos los cursos que tengan en su nombre la palabra “Programación”, podemos ejecutar:

```
select * from CURSO
where nombre LIKE '%Programación%'
```

Con los porcentajes especificamos que puede haber 0 o más caracteres desconocidos antes (cuando lo agregamos antes de la palabra) o al final (cuando lo agregamos después de la palabra).

El resultado sería el que se muestra a continuación.

Tabla 29: Operación LIKE

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	103	Programación Java	POO en Java	35	Noche	4
	104	Programación Web	NULL	35	Noche	5
	105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

Fuente: elaboración propia

Además de los porcentajes, es posible utilizar guiones bajos.

El guion bajo se utiliza cuando sabemos exactamente la cantidad de caracteres desconocidos que hay adelante (o al final) de la parte conocida de la palabra. Por ejemplo, si ejecutamos la siguiente consulta:

select * from CURSO where nombre LIKE '%Programación _ _ _ _'

Tabla 30: Operación LIKE

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	103	Programación Java	POO en Java	35	Noche	4

Fuente: elaboración propia

Solo obtenemos el curso de Programación en Java porque especificamos 4 guiones bajos después de “Programación “.

BETWEEN

BETWEEN sirve para agregar un rango de valores como filtro. Por ejemplo, si queremos todos los cursos cuyo cupo sea entre 20 y 30 debemos configurar lo siguiente:

SELECT * FROM CURSO **WHERE** cupo **BETWEEN** 20 **AND** 30;

Tabla 31: Operación BETWEEN

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	103	Programación Java	POO en Java	35	Noche	4

Fuente: elaboración propia

AND

El operador lógico AND sirve para unir dos o más condiciones en una misma consulta SQL. Para que un registro sea incluido en el resultado, **todas** las condiciones que estén enlazadas con AND deben ser verdaderas.

```
SELECT *
FROM tabla
WHERE condicion1
AND condicion2
AND ....
```

Por ejemplo:

```
select * from CURSO
where turno = 'Noche' and cupo =35;
```

Resultado:

Tabla 32: Operación AND

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	103	Programación Java	POO en Java	35	Noche	4

Fuente: elaboración propia

OR

Al igual que AND, OR también es un operador lógico para unir condiciones, con la diferencia de que, para que un registro se incluya en el resultado, al menos una de las condiciones enlazadas con OR deben ser verdaderas.

Si tomamos el ejemplo anterior y cambiamos AND por OR:

```
SELECT *
select * from CURSO
where turno = 'Noche' or cupo =35;
```

Queremos todos los cursos que sean del turno noche o que tengan cupo para 35 personas.

Con eso obtendríamos:

Tabla 33: Operación OR

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	103	Programación Java	POO en Java	35	Noche	4
	104	Programación Web	NULL	35	Noche	5
	105	Programación C#	.NET, Visual Studio 2019	30	Noche	6

Fuente: elaboración propia

AS

La palabra reservada AS sirve para darle un alias a una tabla o a una columna de una tabla. En vez de escribir el nombre completo de la tabla (PROFESOR, por ejemplo) o de una columna, definimos un alias con una longitud más corta (P).

El AS es útil, por ejemplo, cuando la consulta obtiene datos de más de una tabla. Si tomamos una de las consultas vistas en la sección JOIN:

```
select profesor.*, curso.nombre
from profesor inner join curso on profesor.id = curso.PROFESOR_id;
```

Podemos simplificarla con el uso de alias, agregando uno para cada tabla (p para la tabla profesor y c para la tabla curso):

```
select p.*, c.nombre
from profesor p inner join curso c on p.id = c.PROFESOR_id;
```

Así nos libramos de la necesidad de tener que escribir el nombre de la tabla cada vez que queramos referirnos a ella.

Utilizar AS también es útil cuando queremos que, en el resultado de la consulta, una columna se muestre con otro nombre.

```
select profesor.*, curso.nombre as 'Curso que dicta'
from profesor p inner join curso c on p.id = c.PROFESOR_id;
```

El resultado de esto es el que se muestra a continuación.

Tabla 34: Operación AS

id	nombre	apellido	fecha_nacimiento	salario	Curso que dicta
1	Juan	Pérez	1990-06-06	55000	Algoritmos
2	María Emilia	Paz	1984-07-15	72000	Matemática Discreta
4	Lucía	Díaz	1991-02-24	45000	Programación Java
5	Raúl	Martínez	1980-10-15	85000	Programación Web
6	Mabel	Ríos	1982-06-12	83000	Programación C#

Fuente: elaboración propia

DISTINCT

Utilizamos SELECT DISTINTC cuando queremos que la consulta devuelva solo registros con valores diferentes.

Por ejemplo, si queremos los turnos que hay para los cursos:

SELECT **distinct** turno FROM CURSO;

Tabla 35: Operación SELECT DISTINTC

	turno
▶	Mañana
	Tarde
	Noche

Fuente: elaboración propia

ORDER BY

Para ordenar el resultado de una consulta en orden ascendente o descendente (teniendo en cuenta uno o más campos) utilizamos ORDER BY.

Por defecto, ORDER BY ordena los datos de manera ascendente.

Por ejemplo:

```
SELECT distinct turno
FROM alkemy.CURSO
order by turno;
```

Esa consulta nos mostrará los diferentes turnos de los cursos, ordenados alfabéticamente desde la A a la Z.

Tabla 36: Operación ORDER BY de manera ascendente

	turno
▶	Mañana
	Noche
	Tarde

Fuente: elaboración propia

Si quisiéramos hacerlo en forma descendente:

```
SELECT distinct turno
FROM alkemy.CURSO
order by turno DESC;
```

Tabla 37: Operación ORDER BY de manera descendente

	turno
▶	Tarde
	Noche
	Mañana

Fuente: elaboración propia

También es posible realizar el orden utilizando un grupo de campos:

```
select * from curso
order by turno desc, cupo asc;
```

De esta forma, ordenará primero los turnos de los cursos de Z a A y, si encuentra turnos repetidos, los ordenará por cupo ascendentemente.

Así, ordenará los turnos, de la tabla curso, en forma descendente primero y luego los cupos en forma ascendente.

Tabla 38: Operación ORDER BY primero descendente y luego ascendente

	codigo	nombre	descripcion	cupo	turno	PROFESOR_id
▶	102	Matemática Discreta	NULL	20	Tarde	2
	105	Programación C#	.NET, Visual Studio 2019	30	Noche	6
	103	Programación Java	POO en Java	35	Noche	4
	104	Programación Web	NULL	35	Noche	5
	101	Algoritmos	Algoritmos y estructuras de datos	20	Mañana	1

Fuente: elaboración propia

LIMIT

La palabra reservada LIMIT sirve para especificar la cantidad máxima de registros que queremos en el resultado de una consulta.

```
SELECT *
FROM tabla
WHERE condiciones
LIMIT n;     Donde n es un número entero positivo
```

Es muy útil cuando se tiene una tabla con un gran volumen de datos y solo queremos ver los primeros registros.

CASE

El CASE va incluido en la parte del SELECT de una consulta. Sirve para evaluar varias condiciones y retornar un valor si son verdaderas. Evalúa las condiciones según el orden en el que aparecen. Si ninguna condición se cumple, se devolverá el valor que aparece después del ELSE.

```
SELECT CASE
    WHEN condicion1 THEN resultado1
    WHEN condicion2 THEN resultado2
    WHEN condicionN THEN resultadoN
    ELSE result
END nombre_columna
FROM tabla;
Por ejemplo:
SELECT nombre,
            CASE
                  WHEN cupo >= 30 THEN 'Cupo Alto'
                    ELSE 'Cupo Bajo'
            END 'Tipo de cupo'
FROM curso;
```

Lo que retornaría:

Tabla 39: Operación CASE

	nombre	Tipo de cupo
▶	Algoritmos	Cupo Bajo
	Matemática Discreta	Cupo Bajo
	Programación Java	Cupo Alto
	Programación Web	Cupo Alto
	Programación C#	Cupo Alto

Fuente: elaboración propia

Documentación

Teoría y ejercicios SQL: https://www.w3schools.com/sql/default.asp

Resumen consultas SQL

Fuente: **Universidad de Sevilla**, (s.f.). Consultas SQL. Recuperado de https://www.cs.us.es/blogs/bd2013/files/2013/09/Consultas-SQL.pdf

Videos

Consultas sencillas

Fuente: **García, F.** [Franklin García]. (s.f.). CONSULTAS en SQL SERVER. Tutorial 2021 [YouTube]. Recuperado de https://www.youtube.com/watch?v=pyzY2JSnPvQ&ab_channel=FranklinGarc%C3%ADa

Fuente: **Sánchez Mendoza, L.** [Limber Sánchez Mnedoza]. (29 de julio de 2011). Consultas simples SQL - La sentencia SELECT (11) [YouTube]. Recuperado de https://www.youtube.com/watch?v=IbafcdsR1YA&ab_channel=LimberSanchezMendoza

Ejercicio 3

Objetivo: en este ejercicio haremos diferentes consultas SELECT aplicando lo visto anteriormente.

Dada la siguiente tabla deberás escribir una consulta SELECT.

Tabla 40: Consulta SELECT para PROFESOR

id	nombre	apellido	fecha_nacimiento	salario
1	Juan	Pérez	1990-06-06	55000
2	María Emilia	Paz	1984-07-15	72000
3	Martín	Correa	1987-12-07	63000
4	Lucía	Díaz	1991-02-24	45000
5	Raúl	Martínez	1980-10-15	85000
6	Mabel	Ríos	1982-06-12	83000

Fuente: elaboración propia

Para cada caso, escribe la consulta correspondiente.

1) Nombre, apellido y fecha de nacimiento de todos los empleados, ordenados por fecha de nacimiento ascendente.

   SELECT nombre, apellido, fecha_nacimiento FROM PROFESOR ORDER BY fecha_nacimiento;

2) Todos los profesores cuyo salario sea mayor o igual a 65 000.

   SELECT * FROM PROFESOR WHERE salario >= 65000;

3) Todos los profesores que nacieron en la década del 80.

   select * from PROFESOR where fecha_nacimiento between '1980-01-01' and '1989-12-31'

4) 5 registros

   SELECT * FROM PROFESOR LIMIT 5;

5) Todos los profesores cuyo apellido inicie con la letra “P”

   SELECT * FROM PROFESOR WHERE apellido LIKE ‘P%’;

6) Los profesores que nacieron en la década del 80 y tienen un salario mayor a 80 000

```
SELECT * FROM PROFESOR WHERE fecha_nacimiento
between '1980-01-01' and '1989-12-31' AND salario >
80000;
```

Referencias

[Imagen sin título sobre tipos de JOIN], (2013). Recuperado de http://biercoff.blogspot.com/2013/10/plain-and-simple-about-sql-types-of.html

Campus MVP, (s.f.). Diseñando una base de datos en el modelo relacional. Recuperado de https://www.campusmvp.es/recursos/post/Disenando-una-base-de-datos-en-el-modelo-relacional.aspx?source=post_page----------

Danisable [Danisable]. (s.f.). Bases de datos desde Cero | Modelo Relacional | Parte 7 [YouTube]. Recuperado de https://www.youtube.com/watch?v=SAp0NFqOQ3w&list=PLAzlSdU-KYwWodXmd8PeJY6zpv0wEwY0J&index=7&ab_channel=Danisable

Digital Guide Ionos, (2018). Normalización: evita las redundancias en las bases de datos. Recuperado de https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/normalizacion/

Edu 4 Java [edu4java]. (13 de noviembre de 2010). Select join con mysql workbench. Video Tutorial 5 SQL en español [YouTube]. Recuperado de https://www.w3schools.com/sql/sql_join.asp

García, F. [Franklin García]. (s.f.). CONSULTAS en SQL SERVER. Tutorial 2021 [YouTube]. Recuperado de https://www.youtube.com/watch?v=pyzY2JSnPvQ&ab_channel=FranklinGarc%C3%ADa

Khan Academy, (s.f.). Desarrollo de SQL. Recuperado de https://es.khanacademy.org/computing/computer-programming/sql-documentation

Oppel, A.; Sheldon, R. (2010). Fundamentos de SQL (3.ra. ed.). México: Pearson Educación.

Píldoras Informáticas [pildorasinformáticas]. (28 de diciembre de 2015). Curso SQL. DDL Creación de tablas. Vídeo 16 [YouTube]. Recuperado de https://www.youtube.com/watch?v=XJb6qflbsx4&ab_channel=pildorasinformaticas

Píldoras Informáticas [pildorasinformáticas]. (11 de enero de 2016). Curso SQL. Agregar, eliminar y modificar campos. Vídeo 17 [YouTube]. Recuperado de https://www.youtube.com/watch?v=XJb6qflbsx4&ab_channel=pildorasinformaticas

Sánchez Mendoza, L. [Limber Sánchez Mnedoza]. (29 de julio de 2011). Consultas simples SQL - La sentencia SELECT (11) [YouTube]. Recuperado de https://www.youtube.com/watch?v=IbafcdsR1YA&ab_channel=LimberSanchezMendoza

Universidad de Sevilla, (s.f.). Consultas SQL. Recuperado de https://www.cs.us.es/blogs/bd2013/files/2013/09/Consultas-SQL.pdf

W3 Schools, (s.f.). SQL Joins. Recuperado de https://www.w3schools.com/sql/sql_join.asp


alkemy