**Alumno:** Jonatan Tajada Rico.

Ejercicios de teoría sobre SQL

### Blanca Calderón

## 

* + 1. **Defina brevemente el significado de las siglas SQL.**
* Lenguaje de consultas estructuradas.
  + 1. **El lenguaje SQL se divide en tres partes DDL, DML y DCL. Defina brevemente en qué consiste cada una de las partes.**
* **DDL:** Lenguaje de Definición de Datos:
* Se utiliza para definir la estructura y organización de la base de datos. Las operaciones DDL incluyen:

1. Creación de tablas
2. Modificación de la estructura de tablas – Agregar o Eliminar col)
3. Definición de restricciones (pk/fk)
4. Y otras acciones relacionadas con la estructurada de bbdd.

* **DML:** Lenguaje de Manipulación de Datos:
* Se utiliza para manipular los datos de la base de datos, pero no la estructura, es decir, lo que contienen las tablas. Las operaciones que incluyen son:

1. Insert
2. Update
3. Delete
4. Select

* **DCL:** Lenguaje de Control de Datos:
* Se utiliza para controlar los aspectos de seguridad y acceso a los datos en la base de datos. Las operaciones DCL incluyen son:

1. Grant (conceder privilegios)
2. Revoke (eliminar privilegios)
   * 1. **Defina brevemente qué es MySQL Server.**

* MySQL Server es un sistema de gestión de bases de datos relacional. MySQL es ampliamente utilizado para administrar y almacenar datos en aplicaciones web y empresariales. Ofrece soporte para el lenguaje de consulta estructurado (SQL) y es conocido por su velocidad, confiabilidad y facilidad de uso. MySQL Server gestiona la creación, modificación y consulta de bases de datos, así como la implementación de medidas de seguridad y control de acceso.
  + 1. **Defina brevemente qué es MySQL WorkBench.**
* Es una herramienta visual de diseño y administración de bases de datos MySQL. Proporciona un entorno gráfico que permite a los desarrolladores y administradores de bases de datos diseñar, modelar, generar y administrar esquemas de bases de datos de manera intuitiva. Además, ofrece características como la visualización de consultas SQL, la administración de conexiones y la ejecución de consultas directamente desde la interfaz.
  + 1. **Defina brevemente qué es phpMyAdmin.**
* Es una aplicación web de código abierto diseñada para administrar bases de datos MySQL a través de un navegador web. Ofrece una interfaz gráfica para realizar tareas como la administración de bases de datos, la ejecución de consultas SQL, la gestión de usuarios y privilegios, la importación y exportación de datos, entre otras funciones. phpMyAdmin facilita la administración de bases de datos MySQL sin la necesidad de utilizar comandos SQL directamente.
  + 1. **Defina brevemente el significado de LAMP Stack.**
* *LAMP:* es un acrónimo que representa un conjunto de software comúnmente utilizado para desarrollar y ejecutar aplicaciones web. LAMP incluye los siguientes componentes:
* *Linux*: El sistema operativo.
* *Apache*: El servidor web.
* *MySQL*: El sistema de gestión de bases de datos.
* *PHP/Python/Perl:* Lenguajes de programación utilizados para el desarrollo de aplicaciones web.
  + 1. **Define para qué sirven las siguientes palabras reservadas de la creación de tablas en MySQL:**
       - AUTO\_INCREMENT
       - CHARACTER SET
       - COLLATION
       - ENGINE

*AUTO\_INCREMENT:*

Se utiliza para asignar automáticamente valores incrementales a una columna, generalmente utilizada para claves primarias, facilitando la creación de identificadores únicos.

*CHARACTER SET:*

Define el conjunto de caracteres que se utilizará para la columna de tipo de texto, especificando cómo se deben interpretar los datos almacenados en esa columna.

*COLLATION*:

Indica el orden y la comparación de los datos almacenados en una columna de texto. Define las reglas para ordenar y comparar caracteres, lo que puede ser relevante en búsquedas y ordenamientos.

*ENGINE*:

Especifica el motor de almacenamiento que se utilizará para la tabla. InnoDB y MyISAM son ejemplos de motores de almacenamiento en MySQL. Cada motor tiene sus propias características y ventajas.

* + 1. **¿Qué diferencias existen entre los motores de almacenamiento InnoDB y MyISAM de MySQL? ¿Cuál de los dos es el que se usa por defecto?**
* *InnoDB*: Transaccional, bloqueo de nivel de fila, soporte para claves foráneas, recuperación mejorada, mayor uso de memoria.
* *MyISAM*: No transaccional, bloqueo de nivel de tabla, sin soporte nativo para claves foráneas, recuperación manual, uso eficiente de memoria en algunos casos.
* *InnoDB:* es el motor por defecto.
  + 1. **¿Qué diferencia existe entre los sets de caracteres utf8 y utf8mb4 en MySQL?**

*Rango de Caracteres:*

***utf8mb4:*** Admite un rango más amplio de caracteres Unicode, incluidos aquellos fuera del Plano Multilingüe Básico (BMP). Esto es esencial cuando necesitas almacenar emojis u otros caracteres que no están en el BMP.

***utf8:*** Originalmente diseñado para admitir solo el BMP, lo que significa que no puede almacenar algunos caracteres fuera de este plano, como algunos emojis y símbolos menos comunes.

*Espacio de Almacenamiento:*

**utf8mb4:** Puede requerir más espacio de almacenamiento por carácter debido a su capacidad para manejar caracteres más amplios.

**utf8:** Al ser más limitado en términos de rango de caracteres, puede requerir menos espacio de almacenamiento por carácter.

*Compatibilidad:*

**utf8mb4:** Se recomienda para aplicaciones que necesitan ser compatibles con una amplia gama de caracteres, especialmente aquellas que incluyen emojis y caracteres fuera del BMP.

**utf8:** Puede ser suficiente en aplicaciones que no necesitan admitir caracteres más allá del BMP y buscan ahorrar espacio de almacenamiento.

* + 1. **¿Qué diferencia hay entre los tipos de datos VARCHAR y CHAR?**
* *VARCHAR:* Almacena cadenas de longitud variable.
* *CHAR:* Almacena cadenas de longitud fija.
  + 1. **¿Qué diferencia hay entre los tipos de datos TEXT y BLOB?**
* *TEXT:* Almacena datos de texto de longitud variable.
* *BLOB:* Almacena datos binarios de longitud variable.
  + 1. **¿Qué diferencia hay entre los tipos de datos DATETIME y TIMESTAMP?**
* *DATETIME:* Rango de años más amplio, no afectado por zona horaria, no se actualiza automáticamente.
* *TIMESTAMP*: Rango de años más limitado, almacenado en UTC, se actualiza automáticamente.
  + 1. **¿Qué diferencia hay entre ENUM y SET?**
* *ENUM*: Almacena un solo valor de una lista predefinida.
* *SET*: Almacena múltiples valores de una lista predefinida.
  + 1. **¿Qué tipo de dato usarías si necesitas almacenar un número real exacto sin redondeo?**
* Para almacenar un número real exacto sin redondeo, se suele utilizar el tipo de dato DECIMAL en MySQL. Este tipo de dato permite almacenar números decimales con una precisión fija, evitando problemas de redondeo inherentes a otros tipos de datos como FLOAT o DOUBLE.
* Numeric.
  + 1. **Explica brevemente para qué sirven las cláusulas ON DELETE y ON UPDATE.**
* *ON DELETE:* Se utiliza al definir una clave foránea para especificar la acción que se llevará a cabo en las filas de la tabla referenciada cuando se elimina una fila en la tabla principal. Las opciones comunes son CASCADE (elimina automáticamente las filas relacionadas), SET NULL (establece a NULL los valores en las filas relacionadas) y RESTRICT (impide la eliminación si hay filas relacionadas).
* *ON UPDATE:* Similar a ON DELETE, pero se aplica cuando se actualiza la clave primaria de la tabla principal. Puede tener opciones como CASCADE, SET NULL, SET DEFAULT o RESTRICT. Estas cláusulas ayudan a mantener la integridad referencial en la base de datos.
  + 1. **Explica brevemente para qué sirven las siguientes opciones que podemos usar junto a las cláusulas ON** **DELETE y ON UPDATE:**

1. \* `RESTRICT`
2. \* `CASCADE`
3. \* `SET NULL`
4. \* `NO ACTION`

Estas opciones se utilizan junto a las cláusulas *ON DELETE* y *ON UPDATE* al definir claves foráneas para especificar cómo se deben manejar las relaciones cuando se eliminan o actualizan filas en la tabla principal. Aquí está una breve explicación de cada opción:

*RESTRICT*:

No se permite la acción (eliminación o actualización) si existen filas relacionadas en la tabla secundaria. Es decir, se restringe la operación si hay dependencias.

CASCADE:

La acción (eliminación o actualización) se propaga automáticamente a las filas relacionadas en la tabla secundaria. Si se elimina o actualiza una fila en la tabla principal, las filas correspondientes en la tabla secundaria también se eliminan o actualizan, manteniendo la integridad referencial.

SET NULL:

Cuando se elimina o actualiza una fila en la tabla principal, los valores de las columnas correspondientes en la tabla secundaria se establecen en NULL. Esto es útil cuando se quiere permitir la eliminación o actualización de filas en la tabla principal sin eliminar las filas relacionadas, pero estableciendo los valores a NULL en lugar de eliminar.

NO ACTION:

Similar a RESTRICT, NO ACTION impide la acción (eliminación o actualización) si existen filas relacionadas en la tabla secundaria. Especifica que no se debe tomar ninguna acción y se debe generar un error si hay dependencias. En la práctica, NO ACTION y RESTRICT suelen tener un comportamiento equivalente en muchos sistemas de gestión de bases de datos.

* + 1. **¿Qué diferencia existe entre TRUNCATE y DELETE?**
* *TRUNCATE*: Se utiliza para eliminar todos los registros de una tabla, pero no se registra cada eliminación individualmente en el registro de transacciones. Es más rápido que DELETE y no utiliza tanto espacio de registro, pero no se puede utilizar con condiciones WHERE.
* *DELETE*: Se utiliza para eliminar registros de una tabla, y se pueden aplicar condiciones WHERE para especificar qué registros eliminar. Cada eliminación individual se registra en el registro de transacciones, lo que puede afectar al rendimiento en operaciones masivas.
  + 1. **¿Para qué se utiliza el atributo UNIQUE?**
* El atributo *UNIQUE* se utiliza para asegurar que los valores en una columna o conjunto de columnas sean únicos en cada fila de una tabla. Garantiza la integridad de los datos al prevenir la duplicación de valores en la columna especificada.
  + 1. **¿Qué diferencia hay entre INDEX y KEY? ¿Para qué se utilizan?**
* En MySQL, *INDEX* y *KEY* se utilizan de manera intercambiable. Ambos se utilizan para crear índices en columnas de una tabla, lo que mejora la velocidad de las consultas. En términos prácticos, no hay diferencia significativa entre ellos. La elección entre INDEX y KEY es más cuestión de semántica y preferencia del desarrollador.
  + 1. **¿Para qué se utiliza el atributo ZEROFILL?**
* El atributo *ZEROFILL* se utiliza en combinación con tipos de datos numéricos (por ejemplo, INT, FLOAT) y rellena los valores con ceros a la izquierda hasta alcanzar la longitud especificada. Esto es útil para representar números con una longitud fija de dígitos, manteniendo la consistencia en la presentación visual de los datos.
  + 1. **¿Qué diferencia existe entre declarar una columna como INT(11) y INT(11)ZEROFILL?**
* *INT(11):* Define un entero con una anchura de visualización de 11 dígitos. La anchura de visualización no afecta el rango o la capacidad de almacenamiento del número.
* *INT(11) ZEROFILL*: Además de la anchura de visualización, rellena con ceros a la izquierda hasta alcanzar la longitud especificada. Esto afecta solo la representación visual y no cambia la capacidad de almacenamiento o el rango del número. Es útil para presentar datos numéricos de manera uniforme.
  + 1. **¿Qué significan las llaves y los corchetes en la siguiente descripción de notación?**

1 **CREATE** {DATABASE | SCHEMA} [IF **NOT EXISTS**] nombre\_base\_datos;

* Llaves {}:

Indican opciones mutuamente excluyentes. En este caso, se debe elegir entre DATABASE o SCHEMA. Puedes utilizar cualquiera de las dos palabras clave, pero no ambas.

* Corchetes []:

Indican que la parte incluida entre corchetes es opcional. En este caso, la cláusula [IF NOT EXISTS] es opcional. Puedes elegir incluir esa parte o dejarla fuera según tus necesidades.

* + 1. **¿Qué sentencia SQL utilizarías para conocer cuál es la estructura de una tabla ya creada?**
* DESCRIBE nombre\_de\_tabla; || SHOW COLUMNS FROM nombre\_de\_tabla;
  + 1. **¿Qué sentencia SQL utilizarías para borrar una tabla con el nombre de examen?**
* DROP TABLE examen;
  + 1. **¿Qué sentencia SQL utilizarías para consultar el listado de todas las bases de datos?**
* SHOW DATABASES;
  + 1. **¿Qué sentencia SQL utilizarías para añadir una nueva columna llamada nombre a la tabla estudiante?**
* ALTER TABLE estudiante

ADD COLUMN nombre VARCHAR(255);

* + 1. **¿Qué sentencia SQL nos permite renombrar y modificar el tipo de dato de una columna y sus atributos? Ponga un ejemplo.**
* ALTER TABLE nombre\_de\_tabla

CHANGE viejo\_nombre nuevo\_nombre INT;

* + 1. **¿Qué tipo de dato de MySQL es el único que podría almacenar valores de fecha y hora mayores que**

**2038-01-09 03:14:07?**

* El tipo de dato de MySQL que puede almacenar valores de fecha y hora mayores que el límite del 2038-01-09 03:14:07, superando el problema del año 2038, es DATETIME.
* A diferencia del tipo de dato TIMESTAMP, que está limitado al rango de fechas entre 1970-01-01 00:00:01 y 2038-01-09 03:14:07 debido a restricciones de almacenamiento en 32 bits, el tipo de dato DATETIME permite fechas y horas en un rango mucho más amplio sin verse afectado por el problema del año 2038.
  + 1. **¿Qué tipo de dato utilizarías si necesitaras almacenar números reales exactos sin redondeo?**
* DECIMAL / NUMERIC
  + 1. **¿Qué representan los valores 7 y 4 cuando declaramos una columna como FLOAT(7,4) ?**
* En la declaración FLOAT(7,4), el 7 representa la longitud total del número, y el4 representa la cantidad de decimales. En este caso, se está indicando que el número puede tener hasta 7 dígitos en total, con 4 de ellos después del punto decimal.
  + 1. **¿Qué representa el valor 11 cuando declaramos una columna como INT(11) ?**
* El valor 11 en INT(11) no afecta el rango ni el tamaño de almacenamiento del tipo de dato INT en MySQL. En MySQL, el número entre paréntesis en la declaración INT es solo una indicación de ancho de visualización y no tiene impacto en el rango o en el tamaño de almacenamiento. El valor 11 es comúnmente utilizado para alinearse con ciertos estándares de visualización o herramientas.
  + 1. **¿Qué puerto es el que utiliza por defecto el servicio de MySQL Server?**
* El puerto por defecto que utiliza el servicio de MySQL Server es el puerto 3306. Este es el puerto estándar en el que MySQL espera conexiones por parte de los clientes.

Codigo SQL: SHOW VARIABLES LIKE 'port';

* + 1. **Cuando instalamos MySQL Server está configurado por defecto para que sólo acepte conexiones locales y** **que no permita conexiones remotas. ¿Qué valor tendríamos que poner en la directiva de configuración** **bind-address para permitir conexiones remotas?**

1. [mysqld]
2. bind-address = 127.0.0.1

* Para permitir conexiones remotas en MySQL Server, debes cambiar el valor de la directiva bind-address en el archivo de configuración. En el ejemplo que proporcionas:

[mysqld]

bind-address = 127.0.0.1

* La dirección IP 127.0.0.1 representa la interfaz de loopback, que se refiere solo a la conexión local. Para permitir conexiones remotas, debes cambiar este valor a la dirección IP del servidor o utilizar 0.0.0.0 para escuchar en todas las interfaces disponibles. Aquí tienes un ejemplo:

[mysqld]

bind-address = 0.0.0.0

* + 1. **Los usuarios de MySQL Server se almacenan en la tabla mysql.user. La clave primaria de esta tabla está formada por los valores user y host, de modo que cada fila vendrá identificada por un nombre de usuario y el host desde el que puede conectarse. Según la siguiente tabla, ¿desde qué máquinas se podrá conectar el usuario root@'%'?**

1 +------------------+--------------+

2 | user

3 +------------------+--------------+

| host |

1. | root
2. | root

| %

| localhost |

|

1. | debian-sys-maint | localhost
2. | mysql.session | localhost

8 | mysql.sys

9 +------------------+--------------+

| localhost

|

|

|

* En MySQL, la tabla mysql.user almacena la información de los usuarios y sus privilegios. La clave primaria de esta tabla está formada por las columnas user y host, lo que significa que cada fila se identifica de manera única por un nombre de usuario y el host desde el que puede conectarse.
* Si tienes una entrada en la tabla mysql.user para el usuario root con el host %, significa que ese usuario puede conectarse desde cualquier máquina, ya que % es un comodín que coincide con cualquier host.
* Entonces, para la entrada root@'%', el usuario root puede conectarse desde cualquier máquina, lo que generalmente se interpreta como una conexión desde cualquier dirección IP. Esto es útil para configurar un usuario con privilegios de administrador que puede conectarse desde cualquier lugar.

## Más consultas de teoría

* + 1. **El lenguaje SQL se divide en tres partes DDL, DML y DCL. Defina brevemente en qué consiste DML, cuáles son las sentencias DML y qué operación podemos realizar con cada una de ellas.**
* **DML:** es la parte del lenguaje SQL que se encarga de la manipulación de los datos almacenados en una base de datos.

1. ***Sentencias DML:***

* *SELECT:* Recupera datos de una o varias tablas.
* *INSERT:* Inserta nuevos registros en una tabla.
* *UPDATE:* Modifica los registros existentes en una tabla.
* *DELETE:* Elimina registros de una tabla.

1. ***Operaciones:***

* *SELECT*: Operación de consulta que recupera datos.
* *INSERT*: Operación de inserción que agrega nuevos datos.
* *UPDATE*: Operación de actualización que modifica datos existentes.
* *DELETE*: Operación de eliminación que borra datos.
  + 1. **Indique cuáles son todas las cláusulas que podemos utilizar en la sentencia SELECT y en qué orden se ejecuta cada una de ellas.**
* Las cláusulas en la sentencia SELECT se ejecutan en el siguiente orden:

1. *SELECT*: Especifica las columnas que se van a recuperar.
2. *FROM*: Indica la tabla o tablas de donde se obtendrán los datos.
3. *WHERE*: Filtra los resultados según una condición.
4. *GROUP* *BY*: Agrupa los resultados según una o más columnas.
5. *HAVING*: Filtra los resultados después de aplicar GROUP BY.
6. *ORDER* *BY*: Ordena los resultados según una o más columnas.
   * 1. **Cuando realizamos consultas con la sentencia SELECT en MySQL podemos usar los modificadores ALL, DISTINCT y DISTINCTROW. Explica brevemente en qué consiste cada uno de ellos.**

* Modificadores en la sentencia SELECT en MySQL:

1. *ALL*: Recupera todos los resultados, incluyendo duplicados.
2. *DISTINCT*: Elimina los duplicados de los resultados.
3. *DISTINCTROW*: Similar a DISTINCT, históricamente utilizado en MySQL, pero ahora se recomienda utilizar DISTINCT.
   * 1. **Defina brevemente para qué sirven los siguientes operadores y ponga un ejemplo de uso para cada uno de ellos:**
        + BETWEEN
        + LIKE
        + IS
        + IS NOT

* *BETWEEN*: Se utiliza para seleccionar valores dentro de un rango.
* *LIKE*: Se utiliza para realizar búsquedas de patrones en datos de texto.
* *IS*: Se utiliza para comparar con un valor nulo.
* *IS NOT*: Se utiliza para comparar con un valor que no es nulo.
  + 1. **¿Qué realiza la función COUNT en cada uno de estos casos? ¿Existe alguna diferencia entre ellas?**
       - COUNT(\*)
       - COUNT(columna)
       - COUNT(DISTINCT columna)
* *COUNT(\*):* Cuenta el número total de filas en una tabla.
* *COUNT(columna):* Cuenta el número de filas donde la columna especificada no es nula.
* *COUNT(DISTINCT columna):* Cuenta el número de valores únicos en la columna especificada.
  + 1. **Describa cuáles son las funciones de agregación más utilizadas, que le permiten realizar operaciones específicas sobre un grupo de filas.**
* Las funciones de agregación realizan operaciones específicas sobre un conjunto de filas y devuelven un único valor. Algunas de las más utilizadas son:

1. *SUM(columna):* Calcula la suma de los valores en una columna.
2. *AVG(columna):* Calcula el promedio de los valores en una columna.
3. *COUNT(columna):* Cuenta el número de filas o valores no nulos en una columna.
4. *MAX(columna):* Devuelve el valor máximo en una columna.
5. *MIN(columna):* Devuelve el valor mínimo en una columna.
6. *GROUP\_CONCAT(columna):* Concatena los valores de una columna para cada grupo.
   * 1. **Explique qué diferencia existe entre las cláusulas WHERE y HAVING. Ponga un ejemplo de uso para cada caso.**

* *WHERE:* Se utiliza en la fase de filtrado de filas antes de que se realice la operación de agregación. Se aplica a filas individuales antes de agruparlas o realizar funciones de agregación.

SELECT categoria, AVG(precio) AS promedio

FROM productos

WHERE precio > 50

GROUP BY categoria;

* + 1. **Defina para qué se utiliza la cláusula LIMIT. Dada la sintaxis de la cláusula LIMIT en MySQL explique qué representan las palabras offset y row\_COUNT. Ponga un ejemplo de uso.**

1 [**LIMIT** {[offset,] row\_COUNT | row\_COUNT OFFSET offset}]

* La cláusula LIMIT se utiliza para limitar el número de filas recuperadas por una consulta. Es especialmente útil cuando se trabaja con grandes conjuntos de datos y se desea obtener solo un subconjunto específico de resultados.
* LIMIT {[offset,] row\_COUNT | row\_COUNT OFFSET offset}
* *row\_COUNT:* Representa el número máximo de filas que se deben devolver.
* *offset:* Representa el número de filas que se deben omitir antes de comenzar a devolver filas.

SELECT \* FROM clientes

LIMIT 10;

* + 1. **¿Para qué se utilizan las operaciones INNER JOIN y NATURAL JOIN?. ¿Qué diferencia hay entre ellas?.** **Ponga un ejemplo de una consulta que afecte a dos tablas indicando la diferencia.**
* *INNER JOIN:* Se utiliza para combinar filas de dos o más tablas basándose en una condición de coincidencia entre las tablas.

SELECT empleados.id, empleados.nombre, departamentos.nombre AS departamento

FROM empleados

INNER JOIN departamentos ON empleados.departamento\_id = departamentos.id;

* *NATURAL JOIN:* Se utiliza para combinar filas de dos o más tablas basándose en columnas que tienen el mismo nombre y tipo de datos.

SELECT empleados.id, empleados.nombre, departamentos.nombre AS departamento

FROM empleados

NATURAL JOIN departamentos;

*Diferencia:*

* La principal diferencia es que INNER JOIN utiliza una condición de coincidencia explícita, mientras que NATURAL JOIN utiliza automáticamente columnas con el mismo nombre y tipo de datos.
  + 1. **¿Qué resultado devuelve la operación CROSS JOIN?. Ponga un ejemplo sencillo. ¿Existe otra forma de obtener el mismo resultado de la operación CROSS JOIN? Ponga un ejemplo.**
* *CROSS JOIN:* Devuelve el producto cartesiano de dos o más tablas, es decir, combina cada fila de la primera tabla con cada fila de la segunda tabla.

SELECT empleados.nombre, departamentos.nombre AS departamento

FROM empleados

CROSS JOIN departamentos;

* *Otra forma:*

La misma operación se puede lograr utilizando la cláusula FROM sin una condición de unión.

SELECT empleados.nombre, departamentos.nombre AS departamento

FROM empleados, departamentos;

* + 1. **¿Para qué se utilizan las operaciones LEFT OUTER JOIN y RIGHT OUTER JOIN?. ¿Qué diferencia hay entre ellas? Ponga un ejemplo de una consulta que afecte a dos tablas indicando la diferencia.**
* *LEFT OUTER JOIN:* Devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha. Si no hay coincidencias, se devuelven NULL en las columnas de la tabla derecha.

SELECT empleados.nombre, departamentos.nombre AS departamento

FROM empleados

LEFT OUTER JOIN departamentos ON empleados.departamento\_id = departamentos.id;

* *RIGHT OUTER JOIN:* Devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. Si no hay coincidencias, se devuelven NULL en las columnas de la tabla izquierda.

SELECT empleados.nombre, departamentos.nombre AS departamento

FROM empleados

RIGHT OUTER JOIN departamentos ON empleados.departamento\_id = departamentos.id;

*Diferencia:*

* La diferencia radica en cuál de las dos tablas se considera la "principal". En LEFT OUTER JOIN, la tabla izquierda es la principal, y en RIGHT OUTER JOIN, la tabla derecha es la principal. En términos prácticos, suelen usarse LEFT OUTER JOIN con más frecuencia que RIGHT OUTER JOIN.
  + 1. **¿Qué resultado devuelve la operación FULL OUTER JOIN?. ¿Cómo se puede implementar esta operación en MySQL?**
* La operación FULL OUTER JOIN devuelve todas las filas cuando hay una coincidencia en cualquiera de las tablas y también devuelve las filas de ambas tablas cuando no hay coincidencia. En otras palabras, incluye todas las filas de ambas tablas, coincidentes o no coincidentes.
* En MySQL, la sintaxis para realizar una operación FULL OUTER JOIN no está directamente soportada. Sin embargo, se puede simular utilizando la combinación de LEFT OUTER JOIN y RIGHT OUTER JOIN, junto con la cláusula UNION.

Ejemplo:

SELECT empleados.nombre, departamentos.nombre AS departamento

FROM empleados

LEFT OUTER JOIN departamentos ON empleados.departamento\_id = departamentos.id

UNION

SELECT empleados.nombre, departamentos.nombre AS departamento

FROM empleados

RIGHT OUTER JOIN departamentos ON empleados.departamento\_id = departamentos.id;

* En este ejemplo, la primera parte de la consulta (LEFT OUTER JOIN) obtiene todas las filas de la tabla izquierda (empleados) y las filas coincidentes de la tabla derecha (departamentos). La segunda parte (RIGHT OUTER JOIN) obtiene todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda. El UNION combina ambos conjuntos de resultados, eliminando duplicados. Este enfoque simula el comportamiento de un FULL OUTER JOIN.
  + 1. **Indica si las siguientes consultas son correctas o incorrectas. En caso de ser incorrectas deberá explicar** **el motivo y cómo podrían resolverse de forma correcta.**

a)

1. **SELECT** \*
2. **FROM** alumno
3. **WHERE** teléfono = **NULL**;

* Incorrecta: La comparación con NULL debe hacerse utilizando IS NULL o IS NOT NULL, ya que NULL no se puede comparar directamente con operadores de igualdad o desigualdad.

SELECT \*

FROM alumno

WHERE teléfono IS NULL;

b)

1. **SELECT** \*
2. **FROM** alumno
3. **WHERE** apellido1 = 'S%';

* Incorrecta: debe usar LIKE

c)

1. **SELECT** \*
2. **FROM** alumno

3 **WHERE** fecha\_nacimiento >= '1999/01/01' **AND** '1999/12/31'

* Incorrecta: La condición en la cláusula WHERE no está correctamente estructurada. Se debe especificar la columna después de WHERE para ambas partes de la condición.

SELECT \*

FROM alumno

WHERE fecha\_nacimiento BETWEEN '1999-01-01' AND '1999-12-31';

d)

1. **SELECT** \*
2. **FROM** producto, fabricante
3. **WHERE** fabricante.nombre = 'Lenovo';

* Incorrecta: La consulta realiza una combinación implícita de las tablas producto y fabricante utilizando la coma en la cláusula FROM, pero no especifica cómo deben relacionarse las tablas.

SELECT \*

FROM producto, fabricante

WHERE producto.fabricante\_id = fabricante.id AND fabricante.nombre = 'Lenovo';

e)

1. **SELECT** \*
2. **FROM** producto **INNER JOIN** fabricante
3. **WHERE** fabricante.nombre = 'Lenovo';

* Incorrecta: Falta la condición de unión en la cláusula ON.

SELECT \*

FROM producto INNER JOIN fabricante

ON producto.fabricante\_id = fabricante.id

WHERE fabricante.nombre = 'Lenovo';

f)

1. **SELECT** \*
2. **FROM** producto **INNER JOIN** fabricante
3. **ON** producto.codigo = fabricante.codigo;

* Correcta: La consulta está correctamente formulada para realizar una unión interna entre las tablas producto y fabricante utilizando la condición ON.

g)

1. **SELECT** \*
2. **FROM** producto
3. **WHERE** precio > **AVG**(precio);

* *Incorrecta*: La función de agregación AVG no puede ser utilizada directamente en la cláusula WHERE.

SELECT \*

FROM producto

WHERE precio > (SELECT AVG(precio) FROM producto);

Las "consultas de resumen" se refieren a aquellas consultas SQL que utilizan funciones de agregación como COUNT, SUM, AVG, etc., para resumir o condensar la información de una o más tablas en una forma más compacta y significativa. Estas consultas se utilizan para obtener información estadística o resumida de los datos almacenados en una base de datos.

En SQL, hay varios tipos de consultas, y pueden clasificarse en categorías como:

Consultas de Selección (SELECT): Se utilizan para recuperar datos de una o más tablas.

Consultas de Filtrado (WHERE): Se utilizan para recuperar un subconjunto de datos que cumplen con ciertos criterios de filtrado.

Consultas de Ordenamiento (ORDER BY): Se utilizan para ordenar los resultados de una consulta en un orden específico.

Consultas de Unión (JOIN): Se utilizan para combinar datos de dos o más tablas basándose en una condición de relación.

Consultas de Resumen: Como se mencionó anteriormente, se utilizan funciones de agregación para resumir datos.

Consultas de Agrupación (GROUP BY): Se utilizan junto con funciones de agregación para agrupar filas basándose en un conjunto de columnas.

Consultas de Filtrado de Grupos (HAVING): Se utilizan para filtrar grupos de filas basándose en condiciones de agregación.

Consultas de Modificación de Datos (INSERT, UPDATE, DELETE): Se utilizan para modificar los datos en las tablas.

Consultas de Definición de Datos (CREATE, ALTER, DROP): Se utilizan para definir, modificar o eliminar estructuras de base de datos.

Cada tipo de consulta tiene su propio propósito y se utiliza en diferentes situaciones según los requisitos específicos del problema que se está abordando.