Investigación Diferentes Sentencias para Consultas

En MySQL, las consultas se realizan principalmente con la sentencia SELECT, que permite obtener información almacenada en una o varias tablas. A partir de esta sentencia, se pueden aplicar diferentes complementos y condiciones que hacen posible personalizar los resultados.

```
SELECT * FROM Usuario;
```

La consulta más básica se hace con SELECT nombre_columna FROM tabla, que recupera solo las columnas indicadas, o con SELECT * FROM tabla, que devuelve todos los campos de la tabla.

```
SELECT Nombre, Apellido, Correo FROM Usuario;
```

Para filtrar registros se utiliza WHERE, junto con operadores de comparación como =, <, >, <=, >=, <> (diferente), y operadores lógicos como AND, OR y NOT. Por ejemplo, se puede obtener únicamente los usuarios de una ciudad determinada o los productos con precios superiores a cierto valor.

```
SELECT * FROM Sensor WHERE TipoSensor = 'Temperatura';
```

Los resultados se pueden organizar con ORDER BY, que permite ordenar en forma ascendente (ASC) o descendente (DESC). También es posible limitar la cantidad de resultados con LIMIT, útil para mostrar solo los primeros registros.

```
SELECT * FROM Sensor ORDER BY ValorActual ASC;
```

```
SELECT * FROM Alerta ORDER BY FechaHora DESC;
```

Cuando se requiere eliminar duplicados, se utiliza DISTINCT, que devuelve solo valores únicos de una columna. Para búsquedas por patrones se emplea LIKE, que junto a los comodines % y _ permite localizar registros que empiecen, terminen o contengan ciertas letras. Otra forma de filtrar es con IN, que compara con una lista de valores, y BETWEEN, que selecciona datos dentro de un rango numérico o de fechas.

```
SELECT DISTINCT TipoCultivo FROM Invernadero;
```

```
SELECT * FROM Invernadero WHERE Ubicacion LIKE 'Calle%';
```

```
SELECT * FROM Invernadero WHERE Ubicacion LIKE 'Calle%';
```

```
SELECT * FROM Invernadero WHERE Tamano BETWEEN 100.00 AND 500.00;
```

Además de estas consultas, MySQL permite realizar operaciones estadísticas con las funciones de agregación: COUNT() para contar registros, SUM() para sumar valores, AVG() para calcular promedios, MIN() para obtener el menor valor y MAX() para el mayor. Estas funciones suelen usarse con GROUP BY, que agrupa registros en categorías, y con HAVING, que filtra esos grupos de manera similar a como WHERE filtra registros individuales.

```
SELECT MAX(ValorActual), MIN(ValorActual) FROM Sensor;
```

```
SELECT COUNT(*) FROM Usuario;
```

La cláusula REFERENCES se usa para definir una **clave foránea**, que asegura que los datos de una columna en una tabla (Sensor en este caso) coincidan con los datos de una clave primaria en otra tabla (Invernadero).

```
postgres=# CREATE TABLE Sensor (
postgres(#     IdSensor INT PRIMARY KEY,
postgres(#     ...
postgres(#     IdInvernadero INT,
postgres(#     FOREIGN KEY (IdInvernadero) REFERENCES Invernadero(IdInvernadero)
postgres(# );
```

```
postgres=# ALTER TABLE Alerta
postgres-# ADD CONSTRAINT FK_Alerta_Sensor
postgres-# FOREIGN KEY (IdSensor)
postgres-# REFERENCES Sensor(IdSensor);
```