In MySQL, a **control structure** is a set of statements that control the flow of execution of a program. There are three types of control structures in MySQL:

- Sequences: A sequence is a set of statements that are executed one after the other.
- Conditional statements: A conditional statement is a statement that executes one set of statements if a certain condition is met, and another set of statements if the condition is not met.
- Loop statements: A loop statement is a statement that repeats a set of statements until a certain condition is met.

A trigger is a set of SQL statements that are executed automatically when a specified change operation (SQL INSERT, UPDATE, or DELETE statement) is performed on a specified table. Triggers are useful for tasks such as enforcing business rules, validating input data, and keeping an audit trail.

A view is a virtual table that is created from a SELECT statement. Views are not stored in the database, but they are dynamically created when they are referenced. Views can be used to simplify complex queries, to hide sensitive data, and to provide a consistent view of data to different users.

Here are some examples of how control structures, triggers, and views can be used in MySQL :-

A sequence can be used to generate a unique ID for each new row that is inserted into a table.
A conditional statement can be used to check if a user is authorized to access a particular table.
A loop statement can be used to iterate through all the rows in a table.
A trigger can be used to automatically update a balance field when a deposit is made to a bank account.
A view can be used to create a simplified view of a complex table.
A view can be used to hide sensitive data from unauthorized users.

In a database management system (DBMS) like MySQL, control structures are used to manage the flow of execution within SQL code. They help you make decisions, repeat actions, and control the logical flow of your queries and scripts. Some common control structures in MySQL include:

1. **IF Statement:**
The IF statement allows you to execute a block of code conditionally based on a specified condition.

```sql
DELIMITER //
CREATE PROCEDURE CheckAge(IN age INT)
BEGIN
   IF age >= 18 THEN
      SELECT 'You are an adult';
   ELSE
      SELECT 'You are a minor';
   END IF;
END //
DELIMITER ;
CALL CheckAge(25);
```

2. **CASE Statement:**
The CASE statement allows you to perform conditional checks within queries.

```sql
SELECT name,
      CASE
         WHEN age >= 18 THEN 'Adult'
         ELSE 'Minor'
      END AS age_group
FROM users;
```

3. **WHILE Loop:**
The WHILE loop repeats a block of code as long as a certain condition is true.

```sql
DELIMITER //
CREATE PROCEDURE CountNumbers()
BEGIN
   DECLARE counter INT DEFAULT 1;
   WHILE counter <= 10 DO
      SELECT counter;
      SET counter = counter + 1;
   END WHILE;
END //
DELIMITER ;
CALL CountNumbers();
```

4. **FOR Loop:**
MySQL doesn't have a native FOR loop, but you can simulate it using a WHILE loop.

```sql
DELIMITER //
```

```
CREATE PROCEDURE CountNumbers()
BEGIN
   DECLARE counter INT DEFAULT 1;
   WHILE counter <= 10 DO
      SELECT counter;
      SET counter = counter + 1;
   END WHILE;
END //
DELIMITER ;

CALL CountNumbers();
```

5. **REPEAT Loop:**
The REPEAT loop repeatedly executes a block of code until a specified condition becomes true.

```sql
DELIMITER //
CREATE PROCEDURE CountNumbers()
BEGIN
   DECLARE counter INT DEFAULT 1;
   REPEAT
      SELECT counter;
      SET counter = counter + 1;
   UNTIL counter > 10 END REPEAT;
END //
DELIMITER ;

CALL CountNumbers();
```

These are just a few examples of control structures in MySQL. Remember to use these structures judiciously and ensure they meet your specific requirements while optimizing performance. The syntax and usage might vary slightly based on the version of MySQL you are using, so always refer to the official MySQL documentation for the most accurate and up-to-date information.

A cursor in MySQL is a database object that allows you to process data returned by an SQL query, one row at a time. It provides you with a way to iterate over the result set and perform operations on each row individually.

To use a cursor in MySQL, you need to first declare it using the `DECLARE` statement. The `DECLARE` statement specifies the name of the cursor, the SELECT statement that will be used to fetch the data, and the mode of the cursor. The mode of the cursor can be either `READ_ONLY` or `READ_WRITE`.

Once the cursor has been declared, you can open it using the `OPEN` statement. The `OPEN` statement tells MySQL to start fetching data from the database.

You can then fetch the data from the cursor using the `FETCH` statement. The `FETCH` statement specifies the name of the cursor and the row number that you want to fetch.

When you are finished with the cursor, you need to close it using the `CLOSE` statement. The `CLOSE` statement tells MySQL to stop fetching data from the cursor.

Here is an example of how to use a cursor in MySQL:

SQL

```
DECLARE cur_customers CURSOR FOR SELECT * FROM customers;

OPEN cur_customers;

FETCH cur_customers INTO @customer_id, @customer_name;

SELECT @customer_id, @customer_name;

CLOSE cur_customers;
```

This code declares a cursor named `cur_customers` that will fetch data from the `customers` table. The `OPEN` statement tells MySQL to start fetching data from the table. The `FETCH` statement fetches the first row from the cursor and stores the values of the `customer_id` and `customer_name` columns in the variables `@customer_id` and `@customer_name`. The `SELECT` statement then prints the values of the `customer_id` and `customer_name` variables. The `CLOSE` statement tells MySQL to stop fetching data from the cursor.

Cursors can be a useful tool for processing data row-by-row. However, they can also be inefficient, so they should only be used when necessary.

Reference:

https://dev.mysql.com/doc/refman/8.1/en/sql-statements.html
https://dev.mysql.com/doc/refman/8.1/en/sql-compound-statements.html