CURSOR

Think of a cursor like a bookmark in a book. When you read a book, you might not read all the pages at once. Instead, you read a few pages, then put a bookmark to remember where you stopped so you can pick up right there later. In a database, a cursor helps you keep track of which row of data you are currently looking at while processing a list of results.

It is essentially a mechanism used to retrieve and manipulate data one row at a time rather than processing an entire set of data at once. A Cursor in DBMS allows programmers to work with individual records,

Allowing the programmer to fetch each row one by one and perform operations on it.

Few scenarios in a (DBMS) where cursors can be particularly useful:

- **1. Generating Reports**
- **Scenario:** A company wants to generate a monthly sales report for each salesperson.
- **Use of Cursor:**
  - You can create a cursor that selects all salespeople from the database.
  - For each salesperson, you can fetch their sales records, calculate totals, and generate a report.
  - This allows you to process each salesperson's data one at a time and create personalized reports.
- **2. Processing Payments**
- **Scenario:** An organization needs to process payments for multiple invoices.
- **Use of Cursor:**
  - You can declare a cursor to select all unpaid invoices.
  - For each invoice, you can fetch the details, update the payment status, and process the payment.
  - This ensures each invoice is handled individually, allowing for error checking and handling.

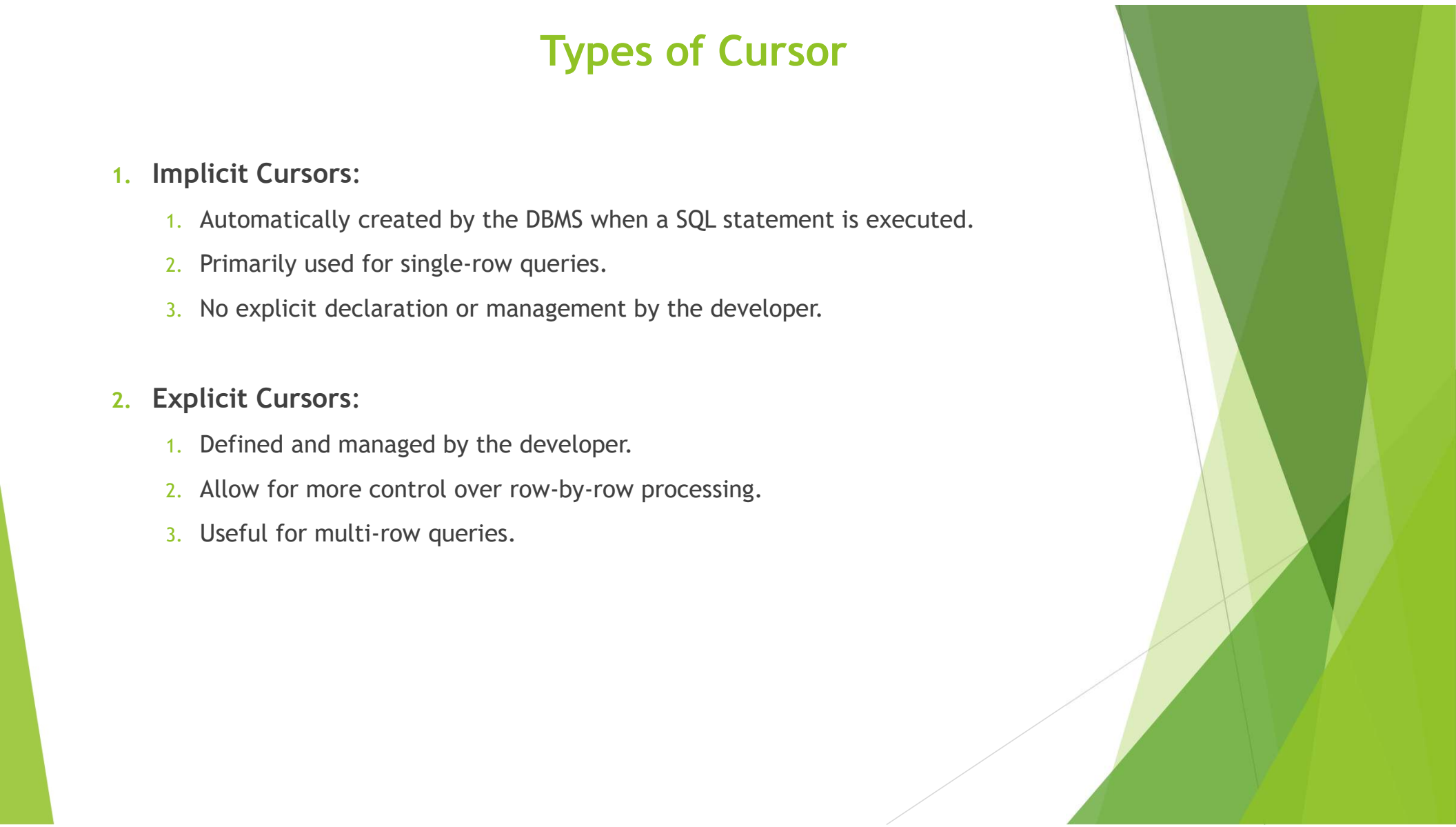
- **3. Complex Business Logic**
- **Scenario**: An application needs to apply complex business rules to a list of customers.
- **Use of Cursor**:
  - You can create a cursor to select all customers.
  - For each customer, you can apply specific rules (like discounts based on purchase history) and update their records accordingly.
  - This allows for detailed and conditional processing of customer data.

- **4. User Notifications**
- **Scenario**: Sending notifications to users based on specific criteria (like subscription renewals).
- **Use of Cursor**:
  - Use a cursor to select users whose subscriptions are about to expire.
  - For each user, fetch their contact details and send them a notification or reminder.
  - This way, you can ensure each user receives a personalized message.

Types of Cursor

1. **Implicit Cursors:**
    1. Automatically created by the DBMS when a SQL statement is executed.
    2. Primarily used for single-row queries.
    3. No explicit declaration or management by the developer.

2. **Explicit Cursors:**
    1. Defined and managed by the developer.
    2. Allow for more control over row-by-row processing.
    3. Useful for multi-row queries.



Cursor Lifecycle:

1. Declaration : *DECLARE my_cursor CURSOR FOR SELECT column_name FROM table_name;*
2. Opening the Cursor : *OPEN my_cursor;*
3. Fetching data : *FETCH my_cursor INTO variable_name;*
4. Handling End of Data : *DECLARE CONTINUE HANDLER FOR NOT FOUND SET done TRUE;*
5. Closing the Cursor : *CLOSE my_cursor;*

Example

```
DELIMITER //
CREATE PROCEDURE example_cursor()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE some_var INT;
    DECLARE my_cursor CURSOR FOR SELECT id FROM my_table;
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN my_cursor;

    read_loop: LOOP
        FETCH my_cursor INTO some_var;
        IF done THEN
            LEAVE read_loop;
        END IF;

        -- Process the row (e.g., print)
        SELECT some_var;
    END LOOP;
    CLOSE my_cursor;
END //
DELIMITER ;
```

- **Advantages of Cursors:**
- **Row-by-Row Processing**: Useful for operations that require processing each row individually.
- **Flexibility**: More control over the processing of query results.

- **Disadvantages of Cursors:**
- **Performance Overhead**: Cursors can be slower than set-based operations because they process rows one at a time.
- **Resource Intensive**: They consume more memory and resources, especially for large datasets.