
ACKNOWLEDGE
HIM
@NAJYWAN

1. Explain E.F. Codd's Rule and its significance in relational database management systems (RDBMS).
Ans.

- E.F. Codd's Rule refers to a set of 12 rules proposed by Edgar F. Codd, the inventor of the relational database model.
- These rules serve as the foundation for ensuring the effectiveness, flexibility, and integrity of relational database management systems (RDBMS).
- They outline the characteristics and requirements that a system must meet to be considered truly relational.
- The significance of E.F. Codd's rules lies in their ability to ensure data integrity, promote data independence, and provide a standardized framework for evaluating the quality of relational database systems.

2. What are the key differences between a Database Management System (DBMS) and a Relational Database Management System (RDBMS)?
Ans.

Aspect	Database Management System (DBMS)	Relational Database Management System (RDBMS)
Data Structure	Hierarchical, Network, or Object-oriented	Relational (Tables consisting of rows and columns)
Data Model	Various models, not necessarily relational	Strictly follows the relational model proposed by E.F. Codd
Data Integrity	Typically less strict	Enforces ACID properties and data integrity through normalization
Query Language	Proprietary or specific to the system	Standardized query language like SQL (Structured Query Language)
Relationships	Not necessarily based on relationships	Data is organized in tables with defined relationships

3. Discuss advanced data types commonly used in RDBMS and provide examples.
Ans.
Advanced data types in RDBMS allow for more complex and specialized data storage and manipulation. Some commonly used advanced data types include:

- Array: Stores an ordered collection of elements, such as integers or strings. Example: An array data type can be used to store a list of phone numbers associated with a customer.
- JSON (JavaScript Object Notation): Stores semi-structured data in JSON format. Example: A JSON data type can be used to store product specifications or user preferences.
- XML (eXtensible Markup Language): Stores hierarchical data in XML format. Example: XML data type can be used to store data with nested structures, such as organizational charts or product categories.

4. Explain the concept of conditional statements in the context of RDBMS. How are they used?
Ans.
Conditional statements in RDBMS are used to control the flow of SQL queries based on specified conditions. They allow for the execution of different SQL commands depending on whether a condition evaluates to true or false. Common conditional statements include:

set if the condition is false.
- CASE: Allows for conditional execution of SQL statements based on multiple conditions.
- WHERE clause: Filters rows based on specified conditions in SELECT, UPDATE, DELETE statements.

5. Describe looping statements in RDBMS and provide examples of their usage.
Ans.
Looping statements in RDBMS allow for the repetition of SQL commands until a specified condition is met. However, SQL typically doesn't have traditional looping constructs like in procedural languages. Instead, iterative operations are often achieved through cursors or recursive queries. For example:

- Cursors: Allow sequential processing of a result set returned by a query.
- Recursive Queries: Enable iterations through self-referencing tables by repeatedly querying the data until a termination condition is met.

Unit 2: Function and Procedure in RDBMS

6. Provide an overview of functions and procedures in RDBMS. How are they different from each other?
Ans.
Overview of Functions and Procedures

- Functions:
  - Functions in RDBMS are named blocks of code that can accept parameters, perform calculations, and return a single value.
  - They are typically used for computations and data transformations within SQL queries.
  - Functions can be used in SQL statements wherever an expression is allowed.
- Procedures:
  - Procedures, also known as stored procedures, are named sets of SQL statements that can accept parameters and perform various operations, including data manipulation, transaction control, and business logic execution.
  - Unlike functions, procedures may or may not return values directly.
  - They are primarily used to encapsulate and execute complex operations within the database server.

Difference: Functions must return a value, whereas procedures may or may not return values. Functions are primarily used to compute and return values, while procedures are used for performing operations without necessarily returning a value.

7. Explain the usage of functions and procedures in the context of RDBMS. Provide examples to illustrate their usage.
Ans.
Functions Example:

```
CREATE FUNCTION calculate_discount(price INT) RETURNS INT
BEGIN
  DECLARE discount INT;
  IF price > 100 THEN
    SET discount = price * 0.1;
```

```
    SET discount = 0;
  END IF;
  RETURN discount;
END;

SELECT calculate_discount(120); -- Output: 12
```

Procedure Example:

```
CREATE PROCEDURE insert_employee(IN emp_name VARCHAR(50), IN emp_salary
DECIMAL(10,2))
BEGIN
  INSERT INTO employees (name, salary) VALUES (emp_name, emp_salary);
END;

CALL insert_employee('John Doe', 50000.00);
```

8. Discuss the process of creating a stored procedure in RDBMS. What are the steps involved?

Ans.

The steps to create a stored procedure in RDBMS typically involve:

- Syntax Declaration: Use CREATE PROCEDURE statement.
- Define Parameters: Specify input and/or output parameters.
- Write Procedure Logic: Write SQL statements and control flow logic within the procedure.
- End Declaration: Use END statement to mark the end of the procedure definition.

9. How do you call stored procedures from other stored procedures in RDBMS? Explain with an example.

Ans.

Stored procedures can be called from other stored procedures using the CALL statement or by invoking them directly within the procedure body.

Example:

```
CREATE PROCEDURE update_employee_salary(IN emp_id INT, IN new_salary
DECIMAL(10,2))
BEGIN
  UPDATE employees SET salary = new_salary WHERE id = emp_id;
END;

CREATE PROCEDURE process_payroll()
BEGIN
  CALL update_employee_salary(101, 55000.00); -- Call another stored procedure
END;
```

10. Compare and contrast functions and procedures in RDBMS. Highlight their key differences and similarities.

Ans.

Aspect	Functions	Procedures
Return Value	Must return a value	May or may not return a value

Aspect	Functions	Procedures
Usage	Primarily used in expressions	Used for executing operations and logic
Transaction Management	Cannot perform transaction control	Can perform transaction control
Input Parameters	Can accept input parameters	Can accept input and output parameters
Execution Context	Executes in the context of SQL statements	Executes independently with its own context

Key Differences:

- Functions must return a value and are primarily used in expressions, while procedures are used for executing operations and can optionally return values.
- Functions execute within the context of SQL statements, while procedures execute independently.
- Both can accept parameters, but functions cannot perform transaction control.

Unit 3: Cursors in RDBMS

11. What is a cursor in the context of RDBMS? Provide an overview of its purpose and functionality.
Ans.

- In the context of RDBMS, a cursor is a database object used to retrieve and manipulate data row by row.
- It provides a mechanism for iterating over a result set returned by a SQL query.
- Cursors are commonly used when it's necessary to process individual rows of a query result set sequentially, especially within stored procedures or application code.

12. Discuss the different types of cursors available in RDBMS. How do they differ from each other?
Ans.
In RDBMS, there are typically three types of cursors:

- Forward-only Cursors: These cursors can only move forward through the result set and do not support scrolling or moving backward.
- Scroll Cursors: These cursors allow movement in both directions within the result set, enabling fetching rows in any order.
- Dynamic Cursors: Dynamic cursors are similar to scroll cursors but allow modifications to the underlying data while the cursor is open.

Differences: The main difference lies in their capabilities for navigating through the result set and the operations they support.

13. Explain the process of declaring a cursor in RDBMS. What are the key components involved?
Ans.
The process of declaring a cursor involves specifying the SQL query whose result set the cursor will traverse and defining the cursor's properties. Key components include:

- Cursor Name: A unique identifier for the cursor.
- SQL Query: The SELECT statement defining the result set.

- Cursor Options: Additional options like locking behavior and sensitivity.

14. How do you open, close, and fetch data using cursors in RDBMS? Provide a step-by-step explanation.
Ans.

- Opening a Cursor: To open a cursor, you use the OPEN statement, specifying the cursor name.
- Fetching Data: Use FETCH statement to retrieve rows from the result set one by one. This can be done in a loop until all rows are fetched.
- Closing a Cursor: Once all rows are fetched or when cursor processing is complete, use the CLOSE statement to release cursor resources.

```
DECLARE cursor_name CURSOR FOR SELECT column1, column2 FROM table_name;
OPEN cursor_name;
FETCH cursor_name INTO variable1, variable2;
CLOSE cursor_name;
```

15. Analyze the advantages and disadvantages of using cursors in RDBMS. When would you choose to use them and when would you avoid them?
Ans.
Advantages:

- Cursors provide fine-grained control over result set traversal.
- They enable row-level processing, allowing for complex data manipulation.
- Useful for scenarios where procedural logic is required within the database.

Disadvantages:

- Cursor operations can be slower compared to set-based operations.
- They may lead to resource contention and increased locking, impacting performance.
- Overuse of cursors can result in inefficient code and potential scalability issues.

Usage Considerations:

- Use cursors when row-level processing is necessary or when set-based operations are not feasible.
- Avoid cursors for simple data retrieval or when dealing with large result sets, as they can degrade performance.

Unit 4 :Triggers and Their Features

16.What is a trigger in the context of RDBMS? Describe its primary features.
Ans.
In the context of RDBMS, a trigger is a database object that automatically executes in response to specified events occurring in the database. Its primary features include:

- Event-driven: Triggers are activated by predefined events such as INSERT, UPDATE, DELETE operations on tables.
- Automatic Execution: Triggers execute automatically when the triggering event occurs.
- Implicit Invocation: Triggers are invoked implicitly by the database system, rather than being called directly by user actions.

maintain data integrity within the database.

17. Discuss the importance of triggers in database management systems.
Ans.
Triggers play a crucial role in database management systems for various reasons:

- They enforce business rules and maintain data integrity.
- Triggers automate repetitive tasks and ensure consistency in data manipulation.
- They provide an additional layer of security by auditing and logging changes to the database.
- Triggers enable complex data validation and constraint enforcement.

18. Explain the concept of a trigger event. Provide examples of trigger events commonly used in RDBMS.
Ans.
A trigger event is an action or operation that occurs in the database, causing a trigger to be executed. Examples of trigger events commonly used in RDBMS include:

- INSERT: Triggered when a new row is inserted into a table.
- UPDATE: Triggered when an existing row is updated in a table.
- DELETE: Triggered when a row is deleted from a table.

19 .Identify and describe the different types of triggers supported in RDBMS.
Ans.
RDBMS typically supports two main types of triggers:

- Row-level Triggers: These triggers execute once for each row affected by the triggering event.
- Statement-level Triggers: These triggers execute once for each triggering event, regardless of the number of rows affected.

20. Compare and contrast row-level triggers and statement-level triggers. When would you use each type?
Ans.

- Row-level Triggers: Executed for each row affected by the triggering event. Useful for validating or modifying data at the row level.
- Statement-level Triggers: Executed once for each triggering event, regardless of the number of rows affected. Suitable for tasks that apply to the entire operation rather than individual rows.

21. List and explain the various trigger events that can activate a trigger in RDBMS.
Ans.
Common trigger events that can activate triggers in RDBMS include:

- BEFORE INSERT, BEFORE UPDATE, BEFORE DELETE
- AFTER INSERT, AFTER UPDATE, AFTER DELETE

22. Discuss scenarios where each trigger event type would be appropriate.
Ans.

- Use BEFORE triggers for data validation, constraint enforcement, or modifying values before they are inserted, updated, or deleted.
- Use AFTER triggers for auditing, logging, or performing actions after the data manipulation operation has been completed.

23. Outline the steps involved in creating a trigger in RDBMS.
Ans.
The general steps for creating a trigger in RDBMS are:

- Define the trigger event (INSERT, UPDATE, DELETE).
- Specify the trigger timing (BEFORE or AFTER).
- Designate the table or tables on which the trigger will operate.
- Define the trigger action or logic to be executed when the trigger fires.

24. Describe the syntax used to define a trigger in SQL.
Ans.
Syntax for Defining a Trigger in SQL :-

```
CREATE [OR REPLACE] TRIGGER trigger_name
{BEFORE | AFTER} {INSERT | UPDATE | DELETE} ON table_name
[FOR EACH ROW]
BEGIN
   -- Trigger logic here
END;
```

25.Implementation of BEFORE and AFTER Triggers
Ans.
Before diving into the implementation details, it's important to understand the concepts of BEFORE and AFTER triggers in RDBMS.

26. Explain the concept of BEFORE and AFTER triggers in RDBMS.
Ans.

- BEFORE Triggers: These triggers execute before the triggering event (e.g., INSERT, UPDATE, DELETE) is processed. They can be used for data validation, constraint enforcement, or modifying data before it is inserted, updated, or deleted.
- AFTER Triggers: These triggers execute after the triggering event has been processed. They are commonly used for auditing, logging, or performing actions after data manipulation operations.

27. Provide examples illustrating the implementation of BEFORE triggers in a database.
Ans.
Let's consider an example of a BEFORE trigger in a database table called employees, where we want to ensure that the salary of an employee being inserted is not negative:

```
CREATE OR REPLACE TRIGGER before_insert_check
BEFORE INSERT ON employees
FOR EACH ROW
BEGIN
   IF :NEW.salary < 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Salary cannot be negative');
   END IF;
END;
```

28. Discuss the use cases for AFTER triggers and provide examples demonstrating their implementation.
Ans.
AFTER triggers can be beneficial for scenarios such as logging changes made to a database table. For example, let's say we have an orders table and we want to log all insertions into this table:

```
CREATE OR REPLACE TRIGGER after_insert_audit
AFTER INSERT ON orders
FOR EACH ROW
BEGIN
  INSERT INTO audit_table (event_type, event_timestamp, affected_row)
  VALUES ('INSERT', SYSTIMESTAMP, :NEW.order_id);
END;
```

29. How do triggers enhance the functionality of a relational database management system? Provide examples to support your answer.
Ans.
Triggers enhance the functionality of RDBMS by automating tasks, enforcing data integrity constraints, and providing auditing capabilities. For example:

- Enforcing referential integrity through triggers ensures that related records are not orphaned or deleted accidentally.
- Using triggers to automatically update denormalized data or maintain summary tables improves query performance and efficiency.
- Auditing changes using triggers helps track data modifications, ensuring accountability and compliance with regulations.

30. Describe a real-world scenario where triggers could be beneficial in ensuring data integrity and consistency within a database.
Ans.
In a banking system, BEFORE triggers can be used to enforce constraints such as ensuring that the withdrawal amount does not exceed the available balance in an account. This ensures data integrity and prevents financial discrepancies.

31 .Discuss the potential drawbacks or limitations of using triggers in RDBMS and how they can be mitigated.
Ans.
Some potential drawbacks of triggers include:

- Increased complexity and maintenance overhead.
- Performance implications, especially for complex triggers or when they are triggered frequently.
- Potential for unintended side effects if not implemented carefully.

These drawbacks can be mitigated by:

- Thorough testing and validation of trigger logic.
- Documentation of trigger functionality and usage.
- Optimizing trigger logic for better performance.
- Limiting the use of triggers to essential functionality and avoiding excessive reliance on them.