CS 306

Recitation 6

Overview

- Check Constraints
- Stored Procedures
- Triggers
- Examples

Check Constraints

- **What is a Check Constraint?**
  - A check constraint is a data validation rule defined within a database table using SQL.
  - It enforces a specific condition that each row in a table must fulfill.
  - The condition is typically expressed as a boolean expression using column values and comparison operators.
  -
- **When to Use Check Constraints:**
  - **Simple Data Validation:** When you need to enforce basic rules on column values within a single table.
  - **Enforce Data Integrity:** To ensure data consistency and accuracy within a table.
  - **Prevent Invalid Data Entry:** To stop invalid values from being inserted into a table.

Check Constraints

- Examples:
  - Ensuring age is greater than 18 (age > 18).
  - Validating price range (price BETWEEN 10 AND 100).
  - Restricting characters in a name field (name REGEXP '^[A-Za-z]+$').

- **Limitations of Check Constraints:**
  - Limited to a single table: Cannot directly reference other tables in the condition.
  - Suitable for basic validation: Complex logic involving joins might not be feasible.

Check Constraints

```
ALTER TABLE customers

ADD CONSTRAINT check_age CHECK (age > 18);




ALTER TABLE products

ADD CONSTRAINT check_price CHECK (price > 10 AND  price < 100);
```

Stored Procedures

- **What is a Stored Procedure?**
  - A stored procedure is a pre-compiled set of SQL statements or a program module stored in a database management system.
  - It acts as a reusable block of code that can be executed by calling its name, similar to a function.
  - Stored procedures can accept input parameters, perform complex operations, and return output values.

- **Long Story Short:** Procedures are functions that you can reuse like the ones you use in programming languages

Stored Procedures

```sql
DELIMITER //

CREATE PROCEDURE calculate_order_total(IN order_id INT, OUT
total_amount DECIMAL(10,2))

BEGIN

  DECLARE subtotal DECIMAL(10,2);

  -- Simulate fetching order details (replace with your actual logic)

  SELECT SUM(price * quantity) INTO subtotal FROM order_items WHERE
order_id = order_id;

  -- Apply any discounts or taxes (replace with your logic)

  SET total_amount = subtotal * 1.08;  -- Assuming 8% tax

  -- Set the output parameter

  SET total_amount = total_amount;

END //

DELIMITER ;
```

```sql
-- Usage example

DECLARE order_total DECIMAL(10,2);


CALL calculate_order_total(123,
@order_total);


SELECT @order_total AS 'Total amount';
```

Triggers

- **What is a Trigger?**
  - A trigger is a stored procedure in a database that automatically executes in response to specific events on a table.
  - These events can be data manipulation language (DML) operations like `INSERT`, `UPDATE`, or `DELETE`.
  - They can happen before or after these operations
  - Triggers can perform a variety of actions, including:
    - Validating data beyond check constraint limitations (involving joins)
    - Performing calculations or updates on other tables based on the triggering event
    - Enforcing complex business logic related to data modifications

Triggers

- **When to Use Triggers:**
    - **Complex Data Validation:** When check constraints are insufficient due to needing to reference other tables or perform more intricate validation logic.
    - **Enforcing Referential Integrity:** To maintain consistency between related tables, especially when foreign key constraints alone might not be enough (e.g., cascading updates/deletes).
    - **Auditing Data Changes:** To track modifications made to tables, such as logging who made the change, when, and what data was affected.
    - **Automating Data Updates:** To perform calculations or updates on other tables based on changes in the triggered table (e.g., updating inventory levels after an order is placed).

Stored Procedures

```
DELIMITER //

CREATE TRIGGER check_customer_on_insert

BEFORE INSERT ON orders

FOR EACH ROW

BEGIN

  DECLARE exists INT;

  SELECT COUNT(*) INTO exists FROM customers WHERE customer_id = NEW.customer_id;

  IF exists = 0 THEN

    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid customer ID.';

  END IF;

END //

DELIMITER ;
```