Database Constraints - Laboratory Work

Objective

Practice implementing and working with database constraints in PostgreSQL including CHECK, NOT NULL, UNIQUE, PRIMARY KEY, and FOREIGN KEY constraints.

Part 1: CHECK Constraints

Task 1.1: Basic CHECK Constraint

Create a table called `employees` with the following structure:

- `employee_id` (integer)
- `first_name` (text)
- `last_name` (text)
- `age` (integer) - must be between 18 and 65
- `salary` (numeric) - must be greater than 0

Write the CREATE TABLE statement with appropriate CHECK constraints.

Task 1.2: Named CHECK Constraint

Create a table called `products_catalog` with:

- `product_id` (integer)
- `product_name` (text)
- `regular_price` (numeric)
- `discount_price` (numeric)

Add a named CHECK constraint called `valid_discount` that ensures:

- `regular_price` is greater than 0
- `discount_price` is greater than 0
- `discount_price` is less than `regular_price`

Task 1.3: Multiple Column CHECK

Create a table called `bookings` with:

- `booking_id` (integer)
- `check_in_date` (date)
- `check_out_date` (date)
- `num_guests` (integer)

Add CHECK constraints to ensure:

- `num_guests` is between 1 and 10
- `check_out_date` is after `check_in_date`

Task 1.4: Testing CHECK Constraints

For each table created above, write INSERT statements that:

1. Successfully insert valid data (at least 2 rows per table)
2. Attempt to insert invalid data that violates each CHECK constraint
3. Document which constraint is violated and why

Part 2: NOT NULL Constraints

Task 2.1: NOT NULL Implementation

Create a table called `customers` with:

- `customer_id` (integer, NOT NULL)
- `email` (text, NOT NULL)
- `phone` (text, can be NULL)
- `registration_date` (date, NOT NULL)

Task 2.2: Combining Constraints

Create a table called `inventory` with:

- `item_id` (integer, NOT NULL)
- `item_name` (text, NOT NULL)
- `quantity` (integer, NOT NULL, must be >= 0)
- `unit_price` (numeric, NOT NULL, must be > 0)
- `last_updated` (timestamp, NOT NULL)

Task 2.3: Testing NOT NULL

Write INSERT statements that:

1. Successfully insert complete records
2. Attempt to insert records with NULL values in NOT NULL columns
3. Insert records with NULL values in nullable columns

Part 3: UNIQUE Constraints

Task 3.1: Single Column UNIQUE

Create a table called `users` with:

- `user_id` (integer)
- `username` (text, must be unique)
- `email` (text, must be unique)
- `created_at` (timestamp)

Task 3.2: Multi-Column UNIQUE

Create a table called `course_enrollments` with:

- `enrollment_id` (integer)
- `student_id` (integer)
- `course_code` (text)
- `semester` (text)
- Add a UNIQUE constraint on the combination of (`student_id`, `course_code`, `semester`)

Task 3.3: Named UNIQUE Constraints

Modify the `users` table from Task 3.1 to:

1. Add a named UNIQUE constraint called `unique_username` on `username`
2. Add a named UNIQUE constraint called `unique_email` on `email`
3. Test by trying to insert duplicate usernames and emails

Part 4: PRIMARY KEY Constraints

Task 4.1: Single Column Primary Key

Create a table called `departments` with:

- `dept_id` (integer, PRIMARY KEY)
- `dept_name` (text, NOT NULL)
- `location` (text)

Insert at least 3 departments and attempt to:

1. Insert a duplicate dept_id
2. Insert a NULL dept_id

Task 4.2: Composite Primary Key

Create a table called `student_courses` with:

- `student_id` (integer)
- `course_id` (integer)
- `enrollment_date` (date)
- `grade` (text)
- PRIMARY KEY should be the combination of (`student_id`, `course_id`)

Task 4.3: Comparison Exercise

Write a document explaining:

1. The difference between UNIQUE and PRIMARY KEY
2. When to use a single-column vs. composite PRIMARY KEY
3. Why a table can have only one PRIMARY KEY but multiple UNIQUE constraints

Part 5: FOREIGN KEY Constraints

Task 5.1: Basic Foreign Key

Given the `departments` table from Task 4.1, create an `employees_dept` table with:

- `emp_id` (integer, PRIMARY KEY)
- `emp_name` (text, NOT NULL)
- `dept_id` (integer, REFERENCES departments)
- `hire_date` (date)

Test by:

1. Inserting employees with valid dept_id
2. Attempting to insert an employee with a non-existent dept_id

Task 5.2: Multiple Foreign Keys

Create a complete database schema for a library system:

Table: authors

- `author_id` (integer, PRIMARY KEY)
- `author_name` (text, NOT NULL)
- `country` (text)

Table: publishers

- `publisher_id` (integer, PRIMARY KEY)
- `publisher_name` (text, NOT NULL)
- `city` (text)

Table: books

- `book_id` (integer, PRIMARY KEY)
- `title` (text, NOT NULL)
- `author_id` (integer, REFERENCES authors)
- `publisher_id` (integer, REFERENCES publishers)
- `publication_year` (integer)
- `isbn` (text, UNIQUE)

Insert sample data into all tables.

Task 5.3: ON DELETE Options

Create a schema demonstrating different ON DELETE behaviors:

Table: categories

- `category_id` (integer, PRIMARY KEY)
- `category_name` (text, NOT NULL)

Table: products_fk

- `product_id` (integer, PRIMARY KEY)
- `product_name` (text, NOT NULL)
- `category_id` (integer, REFERENCES categories ON DELETE RESTRICT)

Table: orders

- `order_id` (integer, PRIMARY KEY)
- `order_date` (date, NOT NULL)

Table: order_items

- `item_id` (integer, PRIMARY KEY)
- `order_id` (integer, REFERENCES orders ON DELETE CASCADE)
- `product_id` (integer, REFERENCES products_fk)
- `quantity` (integer, CHECK (quantity > 0))

Test the following scenarios:

1. Try to delete a category that has products (should fail with RESTRICT)
2. Delete an order and observe that order_items are automatically deleted (CASCADE)
3. Document what happens in each case

Part 6: Practical Application

Task 6.1: E-commerce Database Design

Design and implement a complete database schema for a simple e-commerce system with the following requirements:

Required tables:

1. `customers` (customer_id, name, email, phone, registration_date)
2. `products` (product_id, name, description, price, stock_quantity)
3. `orders` (order_id, customer_id, order_date, total_amount, status)
4. `order_details` (order_detail_id, order_id, product_id, quantity, unit_price)

Constraints to implement:

- All tables must have PRIMARY KEYs
- Appropriate FOREIGN KEYs with correct ON DELETE behaviors
- CHECK constraints to ensure:
  - Price and stock_quantity are non-negative
  - Order status is one of: 'pending', 'processing', 'shipped', 'delivered', 'cancelled'
  - Quantity in order_details is positive
- UNIQUE constraint on customer email
- NOT NULL constraints where appropriate

Deliverables:

1. Complete CREATE TABLE statements
2. At least 5 sample records per table
3. Test queries demonstrating that all constraints work correctly

Submission Requirements

Submit a single SQL file (`.sql`) containing:

1. All CREATE TABLE statements with comments
2. All INSERT statements (successful and failed attempts commented out)
3. Comments explaining what each constraint does
4. Test results documented as SQL comments
5. Your name and student ID at the top of the file