SQL Fundamentals - Travel Application Case Study

Table of Contents

Understanding SQL

SQL (Structured Query Language) is a standard language for managing relational databases. It's divided into several sublanguages based on the type of operations they perform.

Travel Application Database Schema Overview

Our trip application will have the following main entities:

- **Users**: Travelers using the application
- **Destinations**: Places that can be visited
- **Trips**: Planned journeys
- **Bookings**: Hotel/flight reservations
- **Reviews**: User feedback on destinations

DDL (Data Definition Language)

DDL commands are used to define and modify the structure of database objects like tables, indexes, and schemas.

Key DDL Commands:

- `CREATE` - Creates database objects
- `ALTER` - Modifies existing objects
- `DROP` - Deletes objects
- `TRUNCATE` - Removes all data from a table

1. CREATE TABLE Examples

sql

```sql
-- Create Users table
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15),
    date_of_birth DATE,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE
);

-- Create Destinations table
CREATE TABLE destinations (
    destination_id INT PRIMARY KEY AUTO_INCREMENT,
    destination_name VARCHAR(100) NOT NULL,
    country VARCHAR(50) NOT NULL,
    city VARCHAR(50) NOT NULL,
    description TEXT,
    best_time_to_visit VARCHAR(50),
    average_cost_per_day DECIMAL(10,2),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Create Trips table
CREATE TABLE trips (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    destination_id INT NOT NULL,
    trip_name VARCHAR(100) NOT NULL,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    total_budget DECIMAL(12,2),
    actual_cost DECIMAL(12,2),
    trip_status ENUM('planned', 'ongoing', 'completed', 'cancelled') DEFAULT 'planned',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (destination_id) REFERENCES destinations(destination_id)
);

-- Create Bookings table
CREATE TABLE bookings (
    booking_id INT PRIMARY KEY AUTO_INCREMENT,
    trip_id INT NOT NULL,
    booking_type ENUM('flight', 'hotel', 'car_rental', 'activity') NOT NULL,
    booking_reference VARCHAR(50) UNIQUE NOT NULL,
    booking_date DATE NOT NULL,
    check_in_date DATE,
    check_out_date DATE,
    total_amount DECIMAL(10,2) NOT NULL,
    booking_status ENUM('confirmed', 'pending', 'cancelled') DEFAULT 'pending',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (trip_id) REFERENCES trips(trip_id)
);

-- Create Reviews table
CREATE TABLE reviews (
    review_id INT PRIMARY KEY AUTO_INCREMENT,
    user_id INT NOT NULL,
    destination_id INT NOT NULL,
    trip_id INT,
    rating INT NOT NULL CHECK (rating >= 1 AND rating <= 5),
    review_title VARCHAR(100),
    review_text TEXT,
    review_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (destination_id) REFERENCES destinations(destination_id),
    FOREIGN KEY (trip_id) REFERENCES trips(trip_id)
);
```

2. ALTER TABLE Examples

sql

```
-- Add a new column to users table
ALTER TABLE users ADD COLUMN preferred_currency VARCHAR(3) DEFAULT 'USD';

-- Modify column data type
ALTER TABLE destinations ALTER COLUMN average_cost_per_day DECIMAL(12,2);

-- Add index for better performance
ALTER TABLE trips ADD INDEX idx_user_destination (user_id, destination_id);

-- Add constraint
ALTER TABLE bookings ADD CONSTRAINT chk_dates
CHECK (check_out_date >= check_in_date);
```

3. DROP Examples

sql

```sql
-- Drop a column
ALTER TABLE users DROP COLUMN preferred_currency;

-- Drop an index
DROP INDEX idx_user_destination ON trips;

-- Drop a table
DROP TABLE IF EXISTS temp_bookings;
```

4. TRUNCATE Example

sql

```sql
-- Remove all data from reviews table but keep structure
TRUNCATE TABLE reviews;
```

DML (Data Manipulation Language)

DML commands are used to manipulate data within database tables.

Key DML Commands:

- `INSERT` - Adds new data
- `UPDATE` - Modifies existing data
- `DELETE` - Removes data

1. INSERT Examples

sql

```sql
-- Insert users
INSERT INTO users (first_name, last_name, email, phone, date_of_birth) VALUES
('John', 'Doe', 'john.doe@email.com', '+1234567890', '1990-05-15'),
('Jane', 'Smith', 'jane.smith@email.com', '+1234567891', '1985-08-22'),
('Mike', 'Johnson', 'mike.johnson@email.com', '+1234567892', '1992-03-10');

-- Insert destinations
INSERT INTO destinations (destination_name, country, city, description, best_time_to_visit, average_cost_per_day) VALUE
('Eiffel Tower', 'France', 'Paris', 'Iconic iron tower and symbol of France', 'April-October', 150.00),
('Great Wall of China', 'China', 'Beijing', 'Ancient fortification system', 'September-November', 80.00),
('Machu Picchu', 'Peru', 'Cusco', 'Ancient Incan citadel', 'May-September', 120.00);

-- Insert trips
INSERT INTO trips (user_id, destination_id, trip_name, start_date, end_date, total_budget) VALUES
(1, 1, 'Paris Adventure', '2024-06-15', '2024-06-22', 2500.00),
(2, 2, 'China Explorer', '2024-08-01', '2024-08-10', 3000.00),
(3, 3, 'Peru Discovery', '2024-09-05', '2024-09-12', 2800.00);

-- Insert bookings
INSERT INTO bookings (trip_id, booking_type, booking_reference, booking_date, check_in_date, check_out_date, total_a
(1, 'flight', 'FL001234', '2024-06-15', '2024-06-15', '2024-06-22', 800.00),
(1, 'hotel', 'HT001234', '2024-06-15', '2024-06-15', '2024-06-22', 1200.00),
(2, 'flight', 'FL001235', '2024-08-01', '2024-08-01', '2024-08-10', 1200.00);
```

2. UPDATE Examples

sql

```sql
-- Update user information
UPDATE users
SET phone = '+1234567899'
WHERE user_id = 1;

-- Update trip status
UPDATE trips
SET trip_status = 'completed', actual_cost = 2300.00
WHERE trip_id = 1;

-- Update multiple records
UPDATE bookings
SET booking_status = 'confirmed'
WHERE booking_date >= '2024-06-01';

-- Update with JOIN
UPDATE trips t
JOIN users u ON t.user_id = u.user_id
SET t.total_budget = t.total_budget * 1.1
WHERE u.registration_date < '2024-01-01';
```

3. DELETE Examples

```
sql
```

```sql
-- Delete specific booking
DELETE FROM bookings
WHERE booking_id = 3;

-- Delete with condition
DELETE FROM reviews
WHERE rating < 2;

-- Delete with JOIN
DELETE t FROM trips t
JOIN users u ON t.user_id = u.user_id
WHERE u.is_active = FALSE;
```

DQL (Data Query Language)

DQL is used to query and retrieve data from the database.

Key DQL Commands:

- `SELECT` - Retrieves data

1. Basic SELECT Examples

sql

```sql
-- Select all users
SELECT * FROM users;

-- Select specific columns
SELECT first_name, last_name, email FROM users;

-- Select with WHERE clause
SELECT * FROM trips WHERE trip_status = 'completed';

-- Select with multiple conditions
SELECT * FROM destinations
WHERE country = 'France' AND average_cost_per_day < 200;
```

2. Advanced SELECT Examples

sql

```sql
-- JOIN operations
SELECT
    u.first_name,
    u.last_name,
    t.trip_name,
    d.destination_name,
    t.start_date
FROM users u
JOIN trips t ON u.user_id = t.user_id
JOIN destinations d ON t.destination_id = d.destination_id;

-- Aggregate functions
SELECT
    COUNT(*) as total_trips,
    AVG(total_budget) as avg_budget,
    MAX(total_budget) as max_budget,
    MIN(total_budget) as min_budget
FROM trips;

-- GROUP BY with HAVING
SELECT
    destination_id,
    COUNT(*) as trip_count,
    AVG(total_budget) as avg_budget
FROM trips
GROUP BY destination_id
HAVING COUNT(*) > 1;

-- Subqueries
SELECT * FROM users
WHERE user_id IN (
    SELECT user_id FROM trips
    WHERE total_budget > 2500
);

-- Window functions
SELECT
    trip_name,
    total_budget,
    RANK() OVER (ORDER BY total_budget DESC) as budget_rank
FROM trips;
```

DCL (Data Control Language)

DCL commands are used to control access to data in the database.

Key DCL Commands:

- `GRANT` - Gives privileges
- `REVOKE` - Removes privileges

Examples

```sql
-- Create users
CREATE USER 'travel_admin'@'localhost' IDENTIFIED BY 'admin123';
CREATE USER 'travel_user'@'localhost' IDENTIFIED BY 'user123';

-- Grant privileges
GRANT ALL PRIVILEGES ON travel_db.* TO 'travel_admin'@'localhost';
GRANT SELECT, INSERT, UPDATE ON travel_db.users TO 'travel_user'@'localhost';
GRANT SELECT ON travel_db.destinations TO 'travel_user'@'localhost';

-- Grant specific column privileges
GRANT SELECT (trip_name, start_date, end_date) ON travel_db.trips TO 'travel_user'@'localhost';

-- Revoke privileges
REVOKE INSERT ON travel_db.users FROM 'travel_user'@'localhost';
REVOKE ALL PRIVILEGES ON travel_db.* FROM 'travel_user'@'localhost';

-- Show grants
SHOW GRANTS FOR 'travel_user'@'localhost';
```

TCL (Transaction Control Language)

TCL commands are used to manage transactions in the database.

Key TCL Commands:

- `COMMIT` - Saves changes
- `ROLLBACK` - Undoes changes
- `SAVEPOINT` - Sets a point to rollback to

Examples

sql

```
-- Simple transaction
START TRANSACTION;
INSERT INTO users (first_name, last_name, email) VALUES ('Alice', 'Brown', 'alice@email.com');
INSERT INTO trips (user_id, destination_id, trip_name, start_date, end_date, total_budget)
VALUES (LAST_INSERT_ID(), 1, 'Alice Paris Trip', '2024-07-01', '2024-07-07', 2000.00);
COMMIT;

-- Transaction with rollback
START TRANSACTION;
UPDATE trips SET total_budget = total_budget * 1.2 WHERE user_id = 1;
-- If something goes wrong
ROLLBACK;

-- Transaction with savepoints
START TRANSACTION;
INSERT INTO destinations (destination_name, country, city, average_cost_per_day)
VALUES ('Taj Mahal', 'India', 'Agra', 100.00);
SAVEPOINT sp1;

INSERT INTO trips (user_id, destination_id, trip_name, start_date, end_date, total_budget)
VALUES (1, LAST_INSERT_ID(), 'India Trip', '2024-10-01', '2024-10-10', 1500.00);
SAVEPOINT sp2;

-- If we want to rollback to sp1
ROLLBACK TO sp1;
COMMIT;
```

Constraints and Their Types

Constraints are rules enforced on data columns to ensure data integrity.

1. PRIMARY KEY Constraint

sql

```sql
-- Single column primary key
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(100) NOT NULL
);

-- Composite primary key
CREATE TABLE trip_participants (
    trip_id INT,
    user_id INT,
    join_date DATE,
    PRIMARY KEY (trip_id, user_id),
    FOREIGN KEY (trip_id) REFERENCES trips(trip_id),
    FOREIGN KEY (user_id) REFERENCES users(user_id)
);
```

2. FOREIGN KEY Constraint

sql

```sql
-- Basic foreign key
CREATE TABLE bookings (
    booking_id INT PRIMARY KEY AUTO_INCREMENT,
    trip_id INT NOT NULL,
    FOREIGN KEY (trip_id) REFERENCES trips(trip_id)
);

-- Foreign key with actions
CREATE TABLE trip_photos (
    photo_id INT PRIMARY KEY AUTO_INCREMENT,
    trip_id INT NOT NULL,
    photo_url VARCHAR(255),
    FOREIGN KEY (trip_id) REFERENCES trips(trip_id)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

3. UNIQUE Constraint

sql

```sql
-- Column-level unique constraint
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    email VARCHAR(100) UNIQUE NOT NULL,
    phone VARCHAR(15) UNIQUE
);

-- Table-level unique constraint
CREATE TABLE bookings (
    booking_id INT PRIMARY KEY AUTO_INCREMENT,
    booking_reference VARCHAR(50) NOT NULL,
    confirmation_code VARCHAR(20) NOT NULL,
    UNIQUE KEY unique_booking (booking_reference, confirmation_code)
);
```

4. NOT NULL Constraint

sql

```sql
CREATE TABLE destinations (
    destination_id INT PRIMARY KEY AUTO_INCREMENT,
    destination_name VARCHAR(100) NOT NULL,
    country VARCHAR(50) NOT NULL,
    city VARCHAR(50) NOT NULL,
    description TEXT  -- This can be NULL
);
```

5. CHECK Constraint

sql

```sql
CREATE TABLE reviews (
    review_id INT PRIMARY KEY AUTO_INCREMENT,
    rating INT NOT NULL CHECK (rating >= 1 AND rating <= 5),
    review_date DATE CHECK (review_date <= CURDATE())
);

CREATE TABLE trips (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    start_date DATE NOT NULL,
    end_date DATE NOT NULL,
    total_budget DECIMAL(12,2) CHECK (total_budget > 0),
    CHECK (end_date > start_date)
);
```

6. DEFAULT Constraint

sql

```sql
CREATE TABLE users (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(50) NOT NULL,
    registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    is_active BOOLEAN DEFAULT TRUE,
    preferred_currency VARCHAR(3) DEFAULT 'USD'
);

CREATE TABLE trips (
    trip_id INT PRIMARY KEY AUTO_INCREMENT,
    trip_status ENUM('planned', 'ongoing', 'completed', 'cancelled') DEFAULT 'planned',
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

7. Adding Constraints to Existing Tables

sql

```sql
-- Add primary key
ALTER TABLE temp_table ADD PRIMARY KEY (id);

-- Add foreign key
ALTER TABLE bookings ADD CONSTRAINT fk_trip
FOREIGN KEY (trip_id) REFERENCES trips(trip_id);

-- Add unique constraint
ALTER TABLE users ADD CONSTRAINT uk_email UNIQUE (email);

-- Add check constraint
ALTER TABLE reviews ADD CONSTRAINT chk_rating
CHECK (rating >= 1 AND rating <= 5);

-- Add not null constraint
ALTER TABLE destinations MODIFY destination_name VARCHAR(100) NOT NULL;
```

8. Dropping Constraints

sql

```
-- Drop foreign key
ALTER TABLE bookings DROP FOREIGN KEY fk_trip;

-- Drop unique constraint
ALTER TABLE users DROP INDEX uk_email;

-- Drop check constraint
ALTER TABLE reviews DROP CHECK chk_rating;

-- Drop primary key
ALTER TABLE temp_table DROP PRIMARY KEY;
```

Practical Travel Application Scenarios

Scenario 1: User Registration and Trip Planning

```sql
-- Transaction for user registration and first trip
START TRANSACTION;

INSERT INTO users (first_name, last_name, email, phone, date_of_birth)
VALUES ('Sarah', 'Wilson', 'sarah.wilson@email.com', '+1234567893', '1988-12-05');

SET @new_user_id = LAST_INSERT_ID();

INSERT INTO trips (user_id, destination_id, trip_name, start_date, end_date, total_budget)
VALUES (@new_user_id, 1, 'First Paris Trip', '2024-09-15', '2024-09-20', 2200.00);

COMMIT;
```

Scenario 2: Booking Management

sql

```
-- View all bookings for a specific trip
SELECT
    b.booking_reference,
    b.booking_type,
    b.booking_date,
    b.total_amount,
    b.booking_status
FROM bookings b
JOIN trips t ON b.trip_id = t.trip_id
WHERE t.trip_id = 1;

-- Cancel a booking
UPDATE bookings
SET booking_status = 'cancelled'
WHERE booking_reference = 'FL001234';
```

Scenario 3: Trip Analytics

sql

```
-- Most popular destinations
SELECT
    d.destination_name,
    d.country,
    COUNT(t.trip_id) as total_trips,
    AVG(t.total_budget) as avg_budget
FROM destinations d
JOIN trips t ON d.destination_id = t.destination_id
GROUP BY d.destination_id, d.destination_name, d.country
ORDER BY total_trips DESC;

-- User spending analysis
SELECT
    u.first_name,
    u.last_name,
    COUNT(t.trip_id) as total_trips,
    SUM(t.total_budget) as total_planned_budget,
    SUM(t.actual_cost) as total_actual_cost
FROM users u
JOIN trips t ON u.user_id = t.user_id
GROUP BY u.user_id, u.first_name, u.last_name
ORDER BY total_actual_cost DESC;
```

Best Practices

1. **Always use transactions** for related operations
2. **Implement proper constraints** to ensure data integrity
3. **Use indexes** on frequently queried columns
4. **Follow naming conventions** for tables and columns
5. **Regular backups** and maintenance
6. **Use appropriate data types** for optimal storage
7. **Implement proper error handling** in applications
8. **Document your database schema** and relationships

This comprehensive guide covers all the essential SQL concepts using a practical travel application example. Each section builds upon the previous ones, providing a solid foundation for database management and development.