Hitchhiker's Guide Template

GENXXXX

Nakul Bhat

Department of Computer Science and Engineering

Manipal Institute of Technology

Email: nakulbhat034@gmail.com

Phone: +91 8660022842

February 6, 2025

Abstract

Contents

Chapter 1

Basic SQL

SQL (Structured Query Language) is used to interact with relational databases. It helps in retrieving, inserting, updating, and deleting data.

1.1 Basic SQL Commands

Definition 1.1: SQL

SQL is a language for managing relational databases.

1.2 Creating a Database and Table

To create a database and a table:

```sql
CREATE DATABASE mydatabase;

USE mydatabase;

CREATE TABLE students (
    id INT PRIMARY KEY,
    name VARCHAR(100),
    age INT
);
```

Example 1.1: Creating a Students Table

Creates 'mydatabase' and 'students' table with 'id', 'name', and 'age' columns.

1.3 Inserting Data

To insert data into a table:

```sql
INSERT INTO students (id, name, age) VALUES (1, 'Alice', 22);
INSERT INTO students (id, name, age) VALUES (2, 'Bob', 24);
```

Example 1.2: Inserting Data

Adds two student records into the ‘students’ table.

1.4 Retrieving Data

To retrieve data from a table:

```
SELECT * FROM students;
SELECT name, age FROM students WHERE age > 22;
```

Example 1.3: Retrieving Data

Retrieves all records from ‘students’ and selects names and ages of students older than 22.

1.5 Updating Data

To update records in a table:

```
UPDATE students SET age = 23 WHERE id = 1;
```

Example 1.4: Updating Data

Updates Alice’s age to 23.

1.6 Deleting Data

To delete records from a table:

```
DELETE FROM students WHERE id = 2;
```

Example 1.5: Deleting Data

Removes Bob’s record from the ‘students’ table.

1.7 Filtering and Sorting

To filter and sort data:

```
SELECT * FROM students WHERE age > 20 ORDER BY name ASC;
```

Example 1.6: Filtering and Sorting

Retrieves students older than 20 and sorts them alphabetically by name.

Chapter 2

Intermediate and Advanced SQL

2.1 Joins

Joins allow combining data from multiple tables.

```sql
SELECT students.id, students.name, courses.course_name
FROM students
JOIN enrollments ON students.id = enrollments.student_id
JOIN courses ON enrollments.course_id = courses.id;
```

Example 2.1: Using Joins

Retrieves student names and their enrolled courses by joining tables.

2.2 Subqueries

Subqueries allow nesting one query inside another.

```sql
SELECT name FROM students WHERE id IN (SELECT student_id FROM enrollments);
```

Example 2.2: Using Subqueries

Retrieves names of students who are enrolled in at least one course.

2.3 Views

Views are virtual tables based on SQL queries.

```sql
CREATE VIEW student_courses AS
SELECT students.name, courses.course_name
FROM students
```

```
JOIN enrollments ON students.id = enrollments.student_id
JOIN courses ON enrollments.course_id = courses.id;
```

Example 2.3: Creating a View

Creates a view 'student_courses' that stores student names and their enrolled courses.

2.4 Stored Procedures

Stored procedures allow defining reusable SQL code blocks.

```
DELIMITER //
CREATE PROCEDURE GetStudentCourses(IN student_id INT)
BEGIN
    SELECT courses.course_name
    FROM enrollments
    JOIN courses ON enrollments.course_id = courses.id
    WHERE enrollments.student_id = student_id;
END //
DELIMITER ;
```

Example 2.4: Creating a Stored Procedure

Defines a procedure 'GetStudentCourses' that retrieves all courses of a student.