Notes on Booleans and SQL Data Fundamentals

Hia Al Saleh

September 30th, 2024

Contents

1 Introduction

SQL (Structured Query Language) is the standard programming language for creating, updating, and retrieving information from databases. It allows the definition, manipulation, and control of data. In this document, we cover fundamental SQL syntax, Boolean logic, data definition, and constraint management in databases.

2 SQL Overview

SQL is a declarative programming language, meaning you describe what you want, and the database management system (DBMS) determines how to execute it. SQL can be used both interactively or embedded in other programming languages like PHP.

2.1 Key Characteristics of SQL

- **Declarative**: You specify the outcome you want, not how to achieve it.
- **Interactive**: Commands are issued directly to the DBMS and results are displayed.
- **Standardized**: Defined by an international standards group, ensuring wide compatibility.

3 SQL Syntax Conventions

- Each SQL statement begins on a new line.
- Keywords are case insensitive, but generally written in uppercase.
- SQL statements are terminated with a semicolon, though some DBMS like MySQL do not always require this.

3.1 Example of SQL Syntax

```
-- Retrieve authors from New York
SELECT au_fname, au_lname
FROM authors
WHERE state = 'NY'
ORDER BY au_lname;
```

This query selects authors whose state is New York, displaying their first and last names.

4 Unique Identifiers

Unique identifiers in SQL are used to generate primary key values, ensuring each row in a table is unique. SQL supports identity columns, which automatically generate unique values.

4.1 Example: Creating Tables

```
CREATE TABLE titles (
    title_id CHAR(3),
    title_name VARCHAR(40),
    type VARCHAR(10),
    pub_id CHAR(3),
    pages INTEGER,
    price DECIMAL(5,2),
    pubdate DATE
);
```

This SQL statement creates a table called `titles` with columns for the title ID, name, type, publisher ID, pages, price, and publication date.

5 Data Definition Language (DDL)

DDL is used to create, modify, and remove database objects like tables and columns. The primary commands in DDL are `CREATE`, `ALTER`, and `DROP`.

5.1 Table and Column Naming Conventions

- Names must begin with a letter and contain only letters, digits, and underscores.
- No spaces or special characters are allowed.
- Avoid using SQL reserved keywords as names (e.g., `SELECT`, `SUM`).

6 Constraints

Constraints in SQL define rules for the data that can be inserted into a table.

6.1 Common Constraints

- **NOT NULL**: Prevents null values from being inserted into a column.
- **PRIMARY KEY**: Uniquely identifies each row in a table.
- **FOREIGN KEY**: Enforces a relationship between two tables.

- **UNIQUE**: Ensures no duplicate values are inserted into a column.
- **CHECK**: Restricts the values that can be inserted into a column based on a condition.

6.2 Example: Defining Constraints

```
CREATE TABLE publishers (
    pub_id CHAR(3) PRIMARY KEY,
    pub_name VARCHAR(20) NOT NULL,
    city VARCHAR(15),
    state CHAR(2),
    country VARCHAR(15) NOT NULL
);
```

This creates a table with a primary key constraint on the `pub_id` column, and enforces that the `pub_name` and `country` columns cannot be null.

7 Working with Null Values

Null values in SQL represent missing or unknown data. While nulls can simplify representing missing data, they complicate query logic and should be avoided if possible.

7.1 Example: Forbidding Nulls

```
CREATE TABLE authors (
    au_id CHAR(3) NOT NULL,
    au_fname VARCHAR(15) NOT NULL,
    au_lname VARCHAR(15) NOT NULL
);
```

This example ensures that the `au_id`, `au_fname`, and `au_lname` columns cannot contain null values.

8 Primary Keys and Foreign Keys

Primary Keys uniquely identify each row in a table, while **Foreign Keys** create relationships between tables.

8.1 Example: Foreign Key Definition

```
CREATE TABLE royalties (
    title_id CHAR(3) NOT NULL PRIMARY KEY,
    pub_id CHAR(3) NOT NULL,
    CONSTRAINT fk_pub FOREIGN KEY (pub_id) REFERENCES publishers(pub_id)
```

```
);
```

This SQL defines a foreign key relationship between the `royalties` and `publishers` tables, ensuring that the `pub_id` in `royalties` matches a valid `pub_id` in `publishers`.

9 UNIQUE Constraint

A **UNIQUE** constraint ensures that a column contains no duplicate values. It is similar to a primary key, except that a table can have multiple unique constraints and they can contain nulls.

9.1 Example: Unique Constraint

```
CREATE TABLE titles (
    title_id CHAR(3) NOT NULL PRIMARY KEY,
    title_name VARCHAR(40) NOT NULL UNIQUE
);
```

This ensures that each `title_name` in the `titles` table is unique.

10 CHECK Constraint

A **CHECK** constraint limits the values that can be inserted into a column based on a logical condition.

10.1 Example: Check Constraint

```
CREATE TABLE cartitems (
    cart_id INTEGER NOT NULL,
    qty SMALLINT NOT NULL CHECK (qty <= 10)
);
```

This constraint ensures that the quantity (`qty`) cannot exceed 10.

11 Temporary Tables

Temporary Tables are used to store intermediate results for complex queries and are dropped automatically at the end of a session.

11.1 Example: Creating a Temporary Table

```
CREATE TEMPORARY TABLE temp_authors (
    au_id CHAR(3),
    au_fname VARCHAR(15),
    au_lname VARCHAR(15)
);
```

12 Altering Tables

`ALTER TABLE` is used to modify an existing table by adding, modifying, or deleting columns and constraints.

12.1 Example: Altering a Table

```
ALTER TABLE authors ADD COLUMN email VARCHAR(25);
```

13 Dropping Tables

The `DROP TABLE` command is used to permanently remove a table from the database.

13.1 Example: Dropping a Table

```
DROP TABLE authors;
```

14 Conclusion

SQL offers robust functionality for managing database structures and ensuring data integrity through constraints and proper table design. By following SQL syntax rules and utilizing constraints such as `NOT NULL`, `PRIMARY KEY`, `FOREIGN KEY`, and `CHECK`, databases can be effectively managed and maintained.