Schema, Constraints, and Data Types

By VICTORY AROGUNDADE

Database Schema: The Blueprint

A schema is the logical structure of a database, defining how data is organized. It acts as a blueprint, outlining tables, columns, relationships, data types and constraints within the database.

1

Defines Structure

Organizes database components like tables and columns.

2

Ensures Consistency

Maintains a uniform data organization across the database.

3

Facilitates Management

Simplifies data retrieval and manipulation operations.

For example, creating a schema for a school database:

```
CREATE SCHEMA SchoolDB;
CREATE TABLE Students (
 StudentID INT PRIMARY KEY,
 Name VARCHAR(50),
 Age INT
);
```

Constraints: Enforcing Data Integrity

Constraints are rules applied to columns in a table to enforce data integrity and validity. They ensure that data adheres to predefined standards, preventing inconsistencies and errors.

PRIMARY KEY

Uniquely identifies each row in a table.

FOREIGN KEY

Links rows between different tables, maintaining relationships.

UNIQUE

Ensures that all values in a column are distinct.

NOT NULL

Requires a column to always contain a value.

CHECK

Ensures values meet specific conditions (e.g., Age > 18).

Data Types: Defining Data's Nature

Data types define the kind of values a column can store, dictating how data is interpreted and manipulated. Choosing the correct data type is essential for efficient storage and accurate operations. Common data types include:

- **Numeric types:** INT (Integer), DECIMAL or NUMERIC, FLOAT or REAL.
- **Character types:** VARCHAR (Variable Character), CHAR (Character), TEXT.
- **Date and Time types:** DATE, TIME, DATETIME or TIMESTAMP
- **Boolean types:** BOOLEAN or BIT.
- **Binary types:** BINARY or VARBINARY.

These types ensure data is stored and processed correctly, optimizing database performance and reliability.

User-Defined Types: Beyond Built-in Data

User-Defined Types (UDTs) in SQL empower developers to create custom data types beyond the database's built-in options. They represent complex data structures or enforce specific constraints, significantly enhancing data modeling and reusability.

Custom Structures

Model intricate data like custom objects or JSON structures directly within the database.

Code Reusability

Define complex types once and reuse them across various tables, functions, or procedures.

Enhanced Integrity

Embed domain-specific constraints within the type definition for better data validation.

The `CREATE TYPE` statement defines UDTs; its syntax varies slightly across database systems. For instance, in SQL Server, you can create a user-defined table type:

```
CREATE TYPE dbo.OrderDetailsType AS TABLE
(
    ProductID INT,
    Quantity INT,
    UnitPrice DECIMAL(10, 2)
);
```

This example creates a structured table type for order line items, enabling efficient and type-safe handling of complex data sets in stored procedures.

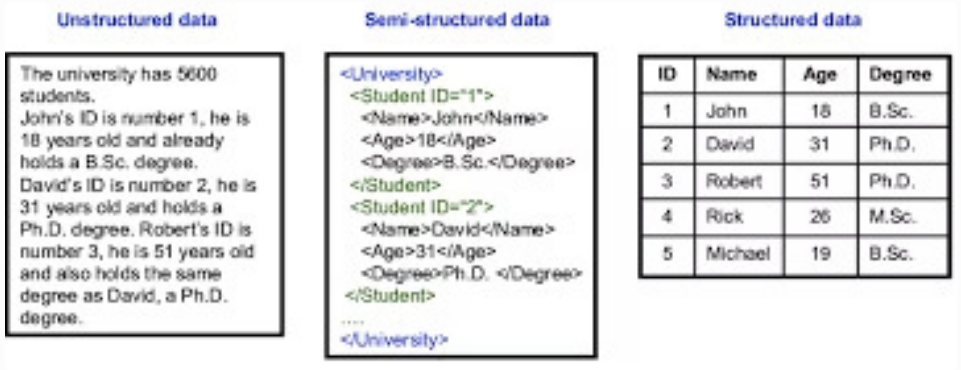
Schema, Constraints, and Data Types: A Comparison

While distinct, schema, constraints, and data types all contribute to data organization, accuracy, and structure within a database.

	Schema	Constraints	Data Types
Similarity	All ensure organization, accuracy, and structure of data in a database.		
Definition	Blueprint of how data is structured.	Rules to ensure validity of data.	Defines nature/format of data stored.
Scope	High-level design of DB.	Applies to individual columns or tables.	Applies at the column level.
Example	Student table structure.	PRIMARY KEY on StudentID.	INT, VARCHAR, DATE.

Data Forms: Structured, Semi-Structured, Unstructured

Data exists in various forms, each with unique characteristics and storage requirements.



Unstructured Data

No predefined format or model, often stored in data lakes. Example: Videos, audio, images, free text, social media posts.

Semi-Structured Data

Has some organization using tags or markers, but not strictly tabular. Example: JSON, XML, NoSQL documents.

Structured Data

Organized into rows and columns, easily stored in relational databases. Example: Customer records (Name, Age, Address).

Comparing Data Types and Their Storage

The format of data dictates its optimal storage method, impacting accessibility and analysis.

Type	Format	Storage	Examples
Structured	Tables (rows & columns)	Relational DB (SQL)	Employee records, sales data
Semi-Structured	Flexible, tagged	NoSQL DB, JSON, XML	Emails, JSON files
Unstructured	Raw, no predefined schema	Data lakes, Hadoop, cloud storage	Images, PDFs, videos, social posts

SQL Command Categories

SQL (Structured Query Language) is the standard language for managing and manipulating relational databases. It's divided into several categories, each serving a distinct purpose in database management. Understanding these categories is crucial for effective database interaction.

DDL – Data Definition Language

DDL commands are used to define and modify the structure of the database. Think of it as laying the foundation and building the framework for your data.

> **Commands**
>
> CREATE, ALTER, DROP, TRUNCATE.

Code Example: CREATE TABLE

```
CREATE TABLE Employees (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(50),
    Salary DECIMAL(10,2)
);
```

For instance, when a company sets up a new HR system, DDL commands are used to create tables for employees, departments, and salaries. If a software update requires adding a new column, ALTER is used. Old test tables are dropped using DDL to clean up the database.

DML – Data Manipulation Language

DML commands handle the actual data within the tables. These are the actions that populate, update, and remove records from your database.

INSERT

Add new records.

UPDATE

Modify existing data.

DELETE

Remove records.

Code Example: INSERT Statement

```
INSERT INTO Employees (EmpID, Name, Salary)
VALUES (101, 'John Doe', 5000.00);
```

When HR adds a new employee record, INSERT is used. Monthly payroll updates salaries with UPDATE. When an employee resigns, their record is deleted from the active employees' table using DELETE.

DQL – Data Query Language

DQL is all about retrieving information. It allows you to ask specific questions of your database and get the data you need for reports and analysis.

Command

SELECT.

Code Example: SELECT Statement

```
SELECT Name, Salary
FROM Employees
WHERE Salary > 4000;
```

A manager might request a report of all employees earning above a certain threshold. Marketing could list customers who purchased products in the last 30 days. Analysts use SELECT queries to analyze spending patterns for decision-making.

DCL – Data Control Language

DCL commands manage user access rights and privileges, ensuring that only authorized individuals can perform certain actions on the database.

GRANT

Give access rights.

REVOKE

Remove access rights.

Code Example: DCL Statements

```
GRANT SELECT ON Employees TO HR_Manager;
```

```
REVOKE UPDATE ON Employees FROM Intern;
```

The Finance team might be granted access to salary tables but not editing rights. An intern could be allowed to view customer names but not contact information. When an employee leaves, their access rights are revoked immediately to maintain security.

TCL – Transaction Control Language

TCL commands manage database transactions to ensure data integrity, making sure that operations are completed reliably or entirely undone.

01

COMMIT

Save changes permanently.

02

ROLLBACK

Undo changes.

03

SAVEPOINT

Set a point to roll back to.

Code Example: TCL Transaction

```
BEGIN;
UPDATE Employees SET Salary = Salary + 500 WHERE EmpID = 101;
COMMIT;
```

When a bank processes a money transfer, TCL ensures that if one part fails, both operations are rolled back. A retail store uses TCL to update inventory after sales, preventing inconsistent stock counts if an error occurs. Developers use savepoints for partial rollbacks during testing.

SQL Command Categories: A Quick Recap

Each SQL command category plays a vital role in managing and interacting with databases, from defining their structure to controlling access and ensuring data integrity.

1. **DDL**
   Defining the database blueprint.

2. **DML**
   Managing data within tables.

3. **DQL**
   Querying data from the database.

4. **DCL**
   Controlling user access and privileges.

5. **TCL**
   Ensuring safe and consistent transactions.