Database Management Basics

by Sophia

WHAT'S COVERED

In this lesson, you will learn about the basic concepts of database storage systems and options. You will be introduced to different types of database storage methods. Lastly, you will learn about the basics of the SQL database command language and how to query data, create new data, and modify and delete existing data within a relational database.

Specifically, this lesson will cover the following:

1. Databases for Web Development

Web applications often need to store data about users and user accounts for use when the visitors return to the application. Writing this data to a basic file may work fine for smaller applications that store nonsensitive information. However, files do not provide much in the way of access controls and protections. Furthermore, as our data storage needs become more complex, text files tend to become less effective and less reliable. This is especially true when we need to host product and service information. This is when we need to consider using a database to store our application data.

Databases can be used to effectively store large amounts of interconnected data. They also use mechanisms to avoid data corruption and allow multiple systems to access data at the same time. Furthermore, **database management systems (DBMS)** provide security, access controls, and redundancy. When web applications and websites evolve and become more complex, a database is often chosen as the method for reliably storing the company, product, and user data.

DID YOU KNOW

Databases are hosted within a database server application. These applications are not like your typical computer application that you interact with, such as Microsoft Word or a web browser. Database servers are background services that run in the background of your system. These servers read and write to and from the system's file storage and listen to the network connection for database connection requests. The way we interact with the database is by establishing a connection to the database server and sending commands through the connection to be executed by the database server.

TERM TO KNOW

Database Management Systems (DBMS)
Software designed to maintain access to some form of data storage, either relational or nonrelational.

2. Database Concepts

Historically, the most common database type was a **relational database**. Relational databases store similar data in tables. A table may be used to store data describing entities such as employees, products, services, categories, orders, line items, and anything else that may be pertinent to the organization or application. Furthermore, these entities are often related to each other in some manner. For example, customers place orders, and multiple line items can be related to a single order. Each customer has a unique ID and each order is related back to a customer by including the customer's ID in the order data. Relational databases use a language called SQL, or Structured Query Language, in order to interact with the database.

However, in recent years, the demand for different database storage models resulted in the development of database models that are referred to as **no-SQL databases**. NoSQL database solutions store data using multiple storage models that include the following:

- Document
- Key–value
- Wide column
- Graph

Document databases store data in data structures referred to as documents and are often written using JSON or XML. Key–value databases store data using the key as a unique identifier, which points to a particular value, much like a variable in programming is used to store a single value. Wide-column databases resemble a table from a relational database but do not include relationships and are used to increase the efficiency of analytical processes. Graph databases store simple objects but focus primarily on storing information about relationships between the different objects.

TERMS TO KNOW

Relational Database
A model of data storage that stores similar data within related tables of data.

No-SQL Database
A model of data storage that uses alternative methods of storing and organizing data.

3. Introduction to SQL

SQL is the primary language used to interact with databases. SQL is used for structured relational databases and enables us to query data from the database, insert new data, as well as update and delete existing data. Additionally, SQL contains the commands for creating and defining the database as well. This includes creating the database, tables and columns, relationships, and more. SQL is also used to manage user accounts and access privileges.

This tutorial section will not focus on the creation of the database since many implementations of database servers include a graphical interface for defining a database structure, also referred to as the **database schema**. However, let us explore some of the basics of retrieving and manipulating data within a database.

TERM TO KNOW

Database Schema
The definition of a database structure.

3a. Querying Data With SELECT

Querying data from a database is the most common and basic operation in SQL. A query consists of multiple parts or clauses, which control a different aspect of the query. For example, if we wanted to retrieve the list of employees from an employee table, we would start with the keyword SELECT and then provide a comma-separated list of column names, or we can use an asterisk to indicate all columns. The SELECT clause not only identifies this as a SELECT operation but also what columns of data we want to include in the query output.

EXAMPLE The SQL SELECT clause

```
SELECT first_name, last_name, emp_id, department
```

The second clause will identify from which table the data will come. We start with the FROM keyword and then the name of the table. Sometimes, we need to specify the schema or database the table belongs to, in which case we can use the dot notation and include the schema name prefixed to the table name.

EXAMPLE The SQL FROM clause

```
FROM employee
or
FROM ABCIncDB.employee
```

If we needed to retrieve a list of all employees in the employee table, we could stop there and issue this query as “SELECT first_name, last_name, emp_id, department FROM employee,” and the database would return these four columns of data from all rows in the employee table.

However, if we wanted to narrow the returned results to the employees assigned to the marketing department or perhaps only one employee, we could add the third clause called the WHERE clause. This clause comes after the FROM clause and includes a condition statement that will filter the results to only those rows that satisfy the condition statement.

EXAMPLE The SQL WHERE clause

```
WHERE department = 'marketing';
```

In the examples above, the first WHERE clause selects only the rows where the department column value equals “marketing.” Furthermore, we can use multiple condition statements by using the keywords AND and OR. Using AND between two condition statements will require that a row satisfy both conditions. Using OR will mean that any row that satisfies at least one condition will be selected.

EXAMPLE SQL multiple conditions with AND

```
SELECT *
FROM employee
WHERE region = 'spain' AND department = 'marketing';
```

Another option is to use the LIKE keyword and the IN keyword when creating condition statements. When we use LIKE, we then add a string that contains fixed characters and the wildcards “%” and “_.” The wildcards can be any character or symbol. The “%” wildcard represents zero, one, or many characters. The “_” (underscore) represents one and only one character. For example, if we want to select all first names that begin with “chris,” we would use the following:

EXAMPLE SQL LIKE and wildcards

```
SELECT *
FROM employee
WHERE first_name LIKE 'chris%';
```

The IN keyword can be used to create a list of possible values that can satisfy the condition.

EXAMPLE The SQL IN WHERE clause

```
SELECT *
FROM employee
WHERE Department IN ('marketing', 'sales', 'accounting');
```

The example above selects all columns from the employee table where the department value is marketing, sales, or accounting. There are many more combinations and options for querying data from a database that are beyond the scope of this course.

3b. Creating Data With INSERT

When we need to add data to a database, we can insert data into a single table using the INSERT SQL command. This command begins with the keyword INSERT and then the optional keyword INTO, followed by the table name.

EXAMPLE The first clause of the INSERT command

```
INSERT INTO employee
```

The next clause is the optional list of comma-separated column names surrounded by parentheses. This specifies which columns we will be inserting data into. This portion can be omitted from the command; however, you then need to be aware of the column names and their order as specified in the table. You will also need to include a value for each column, even if they are null. After the optional list of columns, we provide the VALUES keyword and then a comma-separated list of values to be inserted. This list needs to follow the order of the column names in the SQL command or they need to follow the order of the original table.

EXAMPLE Using SQL INSERT to insert multiple records into an employee table

```
INSERT INTO employee (first_name, last_name, emp_id, department, region)
VALUES ('John', 'Doe', '345', 'Marketing', 'North America'),
    ('Jane', 'Doe', '456', 'Sales', 'South America');
```

The above example inserts two new records into the employee table. We specify the columns that we will be inserting data into, followed by the VALUES keyword, then followed by two sets of values. Each set of values is a comma-separated list of values surrounded by parentheses. The two sets of values are separated by a comma.

3c. Changing Data With UPDATE

When we need to change a value for a record within a table, we will use the UPDATE command. This command begins with UPDATE followed by the table name. Next, we include the SET keyword, followed by the column name and the value it should be updated to.

EXAMPLE The SQL UPDATE command without the WHERE clause

```
UPDATE employee SET department = "Sales";
```

KEY CONCEPT

The important thing to keep in mind with the UPDATE command is that you must specify a WHERE clause. Otherwise, the entire column will be updated and in the example above, all employee departments will be updated to “Sales.” To remedy this, we include the WHERE clause to specify which employee’s data we want to update.

EXAMPLE The SQL UPDATE command with the WHERE clause

```
UPDATE employee SET department = "Sales" WHERE emp_id = 345;
```

3d. Removing Data With DELETE

When we need to remove a record from a table, we can use the DELETE command. However, when using the DELETE command, we have the same issue in that we must include a WHERE clause. Otherwise, all rows will be deleted from the table.

EXAMPLE Using SQL DELETE to delete a record from table with the WHERE clause

```
DELETE FROM employee WHERE emp_id = 456;
```

This will remove the employee with the emp_id value of 456 from the employee table.

SUMMARY

In this lesson, you were introduced to **databases as a storage option for web developers** and the software they develop. You learned about the basic **database concepts** and both relational and nonrelational database models. Lastly, you learned about the basics of the **SQL language** and how to use SQL to **query**, **create**, **change**, and **remove** data from a database.

Source: This Tutorial has been adapted from "The Missing Link: An Introduction to Web Development and Programming " by Michael Mendez. Access for free at https://open.umn.edu/opentextbooks/textbooks/the-missing-link-an-introduction-to-web-development-and-programming. License: Creative Commons attribution: CC BY-NC-SA.

TERMS TO KNOW

Database Management Systems (DBMS)

Software designed to maintain access to some form of data storage, either relational or nonrelational.

Database Schema

The definition of a database structure.

No-SQL Database

A model of data storage that uses alternative methods of storing and organizing data.

Relational Database

A model of data storage that stores similar data within related tables of data.