College Of Engineering Trivandrum

Application Software Development Lab

Abhishek Manoharan
S5 CSE Roll No:2

TVE17CS002

Department of Computer Science

July 30, 2019

Contents

Cycle 1

Exp No 1

INTRODUCTION TO SQL

1 Aim

Understand the basics of SQL

2 Introduction

SQL is a standard language for accessing and manipulating databases.

2.1 What is SQL

- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

2.2 What Can SQL do?

- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

2.3 SQL is a Standard - BUT....

Although SQL is an ANSI/ISO standard, there are different versions of the SQL language.

However, to be compliant with the ANSI standard, they all support at least the major commands (such as SELECT, UPDATE, DELETE, INSERT, WHERE) in a similar manner.

2.4 RDBMS

RDBMS stands for Relational Database Management System.

RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.

3 History of SQL

Dr. E. F. Codd published the paper, "A Relational Model of Data for Large Shared Data Banks", in June 1970 in the Association of Computer Machinery (ACM) journal, Communications of theACM. Codd's model is now accepted as the definitive model for relational database management systems (RDBMS). The language, Structured English Query Language ("SEQUEL") was developed by IBM Corporation, Inc., to use Codd's model. SEQUEL later became SQL (still pronounced "sequel"). In 1979, Relational Software, Inc. (now Oracle Corporation) introduced the first commercially available implementation of SQL. Today, SQL is accepted as the standard RDBMS language.

4 How SQL Works

The strengths of SQL provide benefits for all types of users, including application programmers, database administrators, managers, and end users. Technically speaking, SQL is a data sublanguage. The purpose of SQL is to provide an interface to a relational database such as Oracle, and all SQL statements are instructions to the database. In this SQL differs from general-purpose programming languages like C and BASIC. Among the features of SQL are the following:

1. It processes sets of data as groups rather than as individual units.

2. It provides automatic navigation to the data.

3. It uses statements that are complex and powerful individually, and that therefore stand alone.

Flow-control statements were not part of SQL originally, but they are found in the recently accepted optional part of SQL, ISO/IEC 9075-5: 1996. Flow-control statements are commonly known as "persistent stored modules" (PSM), and Oracle's PL/SQL extension to SQL is similar to PSM.

SQL unifies all of the above tasks in one consistent language. Common Language for All Relational Databases All major relational database management systems support SQL, so you can transfer all skills you have gained with SQL from one database to another. In addition, all programs written in SQL are portable. They can often be moved from one database to another with very little modification. Summary of SQL Statements
SQL statements are divided into these categories:

1.Data Definition Language (DDL) Statements

2.Data Manipulation Language (DML) Statements

3.Transaction Control Statements (TCL)

4.Session Control Statement

5.System Control Statement

5 Managing Tables

A table is the data structure that holds data in a relational database. A table is composed of rows and columns.

A table can represent a single entity that you want to track within your system. This type of a table could represent a list of the employees within your organization, or the orders placed for your company's products.

A table can also represent a relationship between two entities. This type of a table could portray the association between employees and their job skills, or the relationship of products to orders. Within the tables, foreign keys are used to represent relationships.

5.1 Creating Tables

To create a table, use the SQL command CREATETABLE.
Syntax:
CREATE TABLE 'TABLE NAME'('FIELD NAME ' 'DATA TYPE' 'SIZE',.....)

5.2 Altering Tables

1. To add one or more new columns to the table
2. To add one or more integrity constraints to a table
3. To modify an existing column's definition (datatype, length, default value, and NOTNULL integrity constraint)
4. To modify data block space usage parameters (PCTFREE, PCTUSED)
5. To modify transaction entry settings (INITRANS, MAXTRANS)
6. To modify storage parameters (NEXT, PCTINCREASE, etc.)
7. To enable or disable integrity constraints associated with the table
8. To drop integrity constraints associated with the table

6 Data Types

6.1 String Data Types

String Datatypes

The following are the **String Datatypes** in PostgreSQL:

Data Type Syntax	Explanation
char(*size*)	Where ***size*** is the number of characters to store. Fixed-length strings. Space padded on right to equal ***size*** characters.
character(*size*)	Where ***size*** is the number of characters to store. Fixed-length strings. Space padded on right to equal ***size*** characters.
varchar(*size*)	Where ***size*** is the number of characters to store. Variable-length string.
character varying(*size*)	Where ***size*** is the number of characters to store. Variable-length string.
text	Variable-length string.

Figure 1: *String Data Types*

7 Result

7.1 Numerical Data Types

Numeric Datatypes

The following are the **Numeric Datatypes** in PostgreSQL:

Data Type Syntax	Explanation
bit(*size*)	Fixed-length bit string Where ***size*** is the length of the bit string.
varbit(*size*) bit varying(*size*)	Variable-length bit string Where ***size*** is the length of the bit string.
smallint	Equivalent to int2. 2-byte signed integer.
int	Equivalent to int4. 4-byte signed integer.
integer	Equivalent to int4. 4-byte signed integer.
bigint	Big integer value which is equivalent to int8. 8-byte signed integer.
smallserial	Small auto-incrementing integer value which is equivalent to serial2. 2-byte signed integer that is auto-incrementing.
serial	Auto-incrementing integer value which is equivalent to serial4. 4-byte signed integer that is auto-incrementing.
bigserial	Big auto-incrementing integer value which is equivalent to serial8. 8-byte signed integer that is auto-incrementing.
numeric(*m*,*d*)	Where ***m*** is the total digits and ***d*** is the number of digits after the decimal.
double precision	8 byte, double precision, floating-point number
real	4-byte, single precision, floating-point number
money	Currency value.
bool	Logical boolean data type - true or false
boolean	Logical boolean data type - true or false

Figure 2: *Numerical Data Types*

7.2 Date/Time Data Types

Date/Time Datatypes

The following are the **Date/Time Datatypes** in PostgreSQL:

Data Type Syntax	Explanation
date	Displayed as 'YYYY-MM-DD'.
timestamp	Displayed as 'YYYY-MM-DD HH:MM:SS'.
timestamp without time zone	Displayed as 'YYYY-MM-DD HH:MM:SS'.
timestamp with time zone	Displayed as 'YYYY-MM-DD HH:MM:SS-TZ'. Equivalent to timestamptz.
time	Displayed as 'HH:MM:SS' with no time zone.
time without time zone	Displayed as 'HH:MM:SS' with no time zone.
time with time zone	Displayed as 'HH:MM:SS-TZ' with time zone. Equivalent to timetz.

Figure 3: *Date/Time Data Types*

8 Result

Understood the basics of SQL.