SQL PROGRAMMING

FOR BEGINNERS

THE ULTIMATE BEGINNER'S GUIDE TO LEARN SQL PROGRAMMING AND DATABASE MANAGEMENT STEP-BY-STEP, INCLUDING MYSQL, MICROSOFT SQL SERVER, ORACLE AND ACCESS

RICHARD MACHINA

SQL PROGRAMMING FOR BEGINNERS:

THE GUIDE WITH STEP BY STEP PROCESSES ON DATA ANALYSIS, DATA ANALITICS AND PROGRAMMING LANGUAGE. LEARN SQL SERVER TECHNIQUE FOR ANALYZING AND MANIPULATING THE CODES

RICHARD MACHINA

TABLE OF CONTENTS

Introduction

There are a variety of options to choose from when you want to learn a new language. Some of these languages will let you build your very own website and will also help you identify different ways in which you can reach out to your customers. These languages also help you convert potential customers into confirmed customers. Other languages allow you to develop applications and games for your phone. Some of these languages are advanced which makes it difficult for an amateur to start programming immediately. There are other languages like SQL that allow a business owner to keep track of the information they have without any challenges.

Companies often used the Database Management System or DBMS to store the different information about their company. This information includes customer data and sales data. DBMS is the first option that was introduced into the market, which made it easier to work with the data. Over the years, newer methods were developed which helped a business hold onto its information without any difficulty. The simplest DBMS systems designed now are more secure than the ones that were developed a few years ago.

Companies need to hold a lot of information, and some of the information will be personal like the customer's address, credit card information, date of birth, and so on. To maintain the safety of this information, the databases need to be secure. There should also be a way for people to connect the information to analyze the data. This was when the relational database management system was developed. This type of database is like the traditional database system but is more secure. It also uses more technology to ensure that the information is safe.

As a business owner, you will certainly want to look at different options and also pick what type of tool you want to use to manage the database. SQL is the perfect tool for this purpose since it is well designed and easy to use. This language was designed to manage businesses and provides multiple tools that will allow you to keep the information safe.

Benefits of Working with SQL

In addition to a variety of coding languages, you also have some database languages, and each of these languages is different when compared to the other. You may also wonder why you should use SQL over other languages. Therefore, it is important that you know the benefits of SQL.

- High speed

You should use the SQL system if you want to go through volumes of information quickly. The SQL system can find you a lot of information within a few seconds. You will also find the information that you need. If you work with volumes of data, you should use SQL since it is one of the most efficient options.

- Standards that are well defined

If you want to ensure that your database is secure and strong, you should use SQL since it is constantly updated. These updates help to keep the SQL system strong. Other database tools will not have the same standards like SQL, which will affect your analyses. The features in other databases will also make it difficult for you to store all the necessary information.

- You Do Not Need to Code

You do not need to code to use the SQL tool. All you need to do is remember a few syntaxes. You, however, do not have to master coding in SQL if you want to use it to perform any analyses on some data.

- Object-oriented DBMS

Since you will use a database management system when you work with SQL, it makes it easier for you to find all the information you need, store that information and perform the necessary analysis to make informed decisions.

You Can Earn a Lot of Money

You always want to earn more when you work for an organization. You can certainly get a better salary if you know how to use SQL. You can do this by either nurturing your programming skills in SQL or by learning how to maintain a system and keep it running effectively and efficiently. You can also work as an SQL analyst and provide information and insights for a business; it will help the seniors make better decisions. This will help to maximize the profits for any business.

- All Types of Technology Uses SQL

Most businesses use database tools and technologies like MySQL, Microsoft SQL Server, and PostgreSQL. You should also remember that most people use SQL at some point in their lives. If you are not aware, you also use SQL on your smartphone.

- Employers Look for SQL Skills

Most employers actively look for people who know how to use SQL. Yes, an employer is willing to pay you more, but he or she also is aware of the benefits of hiring an individual who is skilled at using SQL. If you want to move jobs or change your area of work, you should learn how to code in SQL. You will be one of the most sought-after candidate for the position

Chapter 1 SQL: Definition (What It Is And What It Is Used For)

Definition

SQL is a programming language that stands for ‘Structured Query Language,’ and it is a simple language to learn considering it will allow interaction to occur between the different databases that are in the same system.

The SQL is a set of instructions that you can use to interact with your relational database. While there are a lot of languages that you can use to do this, SQL is the only language that most databases can understand. Whenever you are ready to interact with one of these databases, the software can go in and translate the commands that you are given, whether you are giving them in form entries or mouse clicks. These will be translated into SQL statements that the database will already be able to interpret.

If you have ever worked with a software program that is database driven, then it is likely that you have used some form of SQL in the past. It is likely that you didn't even know that you were doing this though. For example, there are a lot of dynamic web pages that are database driven. These will take some user input from the forms and clicks that you are making and then will use this information to compose a SQL query. This query will then go through and retrieve the information from the database to perform the action, such as switch over to a new page.

To illustrate how this works, think about a simple online catalog that allows you to search. The search page will often contain a form that will just have a text box. You can enter the name of the item that you would like to search using the form and then you would simply need to click on the search button. As soon as you click on the search button, the web server will go through and search through the database to find anything related to that search term. It will bring those back to create a new web page that will go along with your specific request.

For those who have not spent that much time at all learning a programming language and who would not consider themselves programmers, the commands that you would use in SQL are not too hard to learn. Commands in SQL are all designed with a syntax that fits in with the English language.

At first, this will seem really complicated, and you may be worried about how much work it will be to get it set up. But when you start to work on a few codes, you will find that it is not actually that hard to work with. Often, just reading out the SQL statement will help you to figure out what the command will do.

Uses of SQL

You can use SQL to help out in the following ways:

- SQL helps when you want to create tables based on the data you have.
- SQL can store the data that you collect.
- SQL can look at your database and retrieves the information on there.
- SQL allows you to modify data.
- SQL can take some of the structures in your database and change them up.

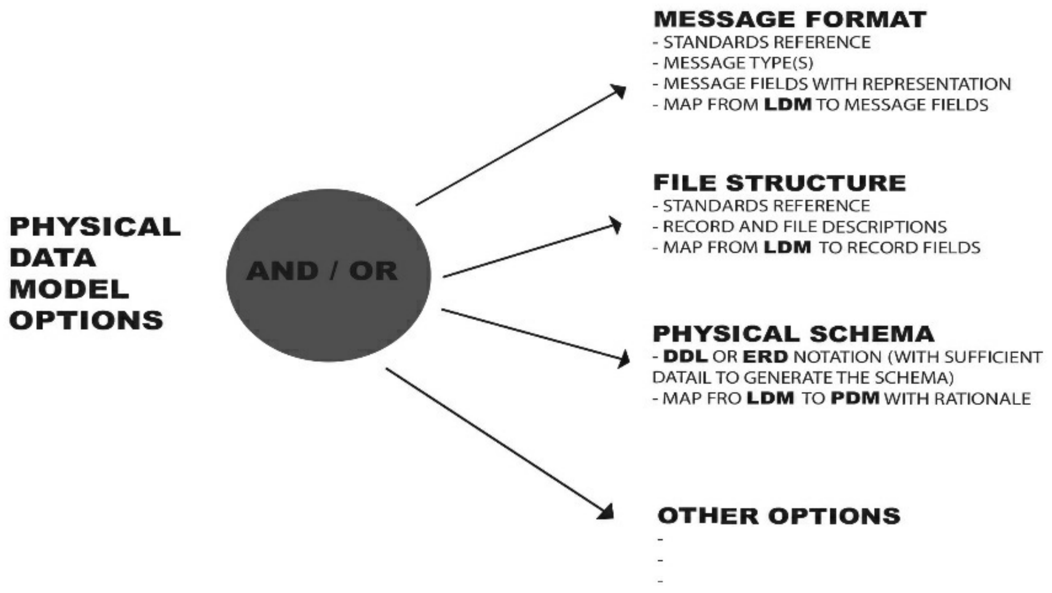
- SQL allows you to combine data.
- SQL allows you to perform calculations.
- SQL allows data protection.

Features of SQL

Relational database

These are ones that are going to be segregated into tables or logical units. These tables can be interconnected inside of the database so that they make sense based on what you are working on at the time. The database is going to also make it easier to break up the data into some smaller units so that you are able to manage them easier and they will be optimized for making things easier on you.

It is important to know about the relational database because it is going to help to keep all your information together, but it does help to split it up so that the pieces are small enough to read through easier. The server will be able to go through all of these parts to see what you need because the smaller pieces are easier to go through compared to the bigger pieces. Because of the optimization and the efficiency that is found in this kind of system, it is common to see a lot of businesses going with this option instead of another one.


PHYSICAL DATA MODEL OPTIONS
AND / OR
MESSAGE FORMAT
- STANDARDS REFERENCE
- MESSAGE TYPE(S)
- MESSAGE FIELDS WITH REPRESENTATION
- MAP FROM LDM TO MESSAGE FIELDS
FILE STRUCTURE
- STANDARDS REFERENCE
- RECORD AND FILE DESCRIPTIONS
- MAP FROM LDM TO RECORD FIELDS
PHYSICAL SCHEMA
- DDL OR ERD NOTATION (WITH SUFFICIENT DATAIL TO GENERATE THE SCHEMA)
- MAP FRO LDM TO PDM WITH RATIONALE
OTHER OPTIONS

Client and Server Technology

When using client-server technology, the application is divided into two parts. The client part provides a convenient graphical interface and is located on the user's computer. The server part provides data management, information sharing, administration and ensures the security of information. The client application generates requests to the database server on which the corresponding commands are executed. Query results are sent to the client.

When developing distributed information systems in the organization of interaction between the client and server parts, the following important tasks in a practical sense are distinguished:

Transfer of a personal database to a server for its subsequent collective use as a corporate database;

Organization of requests from one end of the client over to the company's database will help make sure that the client will receive the right results.

Development of a client application for remote access to a corporate database from a client computer.

The task of transferring a personal database to a server may arise in situations when it is necessary to provide collective access to a database developed using a personal DBMS (FoxPro, Access). To solve this problem, these personal DBMSs have the appropriate tools designed to convert databases to SQL format.

The preparation of queries to the database on the server (in SQL) from the client-side can be performed using a specially designed utility. To provide the user with great opportunities and convenience in preparing and executing requests, client applications are created.

To organize queries to a server database in SQL or using a client application, various methods of interaction are possible that significantly affect efficiency. The main ways of such interaction include:

- Interface DB-LIB (database libraries);
- ODBC technologies (open database compatibility);
- OLE DB interface (linking and embedding database objects);
- DAO technologies (data access objects);
- ADO technologies (data objects).

The DB-LIB interface is an application programming interface specifically designed for SQL. Therefore, it is the least mobile among those considered in the sense of the possibility of transferring applications to another environment. In terms of performance, this method allows the fastest access to information. The reason for this is that it represents an optimized application programming interface and directly uses the SQL system query language.

ODBC technologies are designed to provide the possibility of interconnection between different DBMSs and to receive requests from the application to retrieve information, translate them into the core language of the addressable database to access the information stored in it.

The main purpose of ODBC is to abstract the application from the features of the core of the server database with which it interacts, so the server database becomes as if transparent to any client application.

The advantage of this technology is the simplicity of application development, due to the high level of abstractness of the data access interface of almost any existing DBMS types. Using this technology, it is possible to create client-server applications, and it is advisable to develop the client part of the application using personal DBMS tools and the server part using SQL tools.

The main disadvantage of ODBC technology is the need to translate queries, which reduces the speed of data access. On client-server systems, this drawback is eliminated by moving the request from the client computer to the server computer. This eliminates the intermediate links, which are the main reason for reducing the speed of information processing using the tools of this technology.

When using ODBC tools in a client application, a specific data source is accessed, and through it, to the DBMS that it represents. When installing ODBC tools, the common ODBC subsystem is installed and the “driver-database” pairs are defined, which are the names used to establish the connection with. Corresponding pairs are called named data sources.

Each named data source describes the actual data source and information about access to this data. The data can be databases, spreadsheets and text files. Access information, for example, to a database server, usually includes server location information, database name, account ID and password, as well as various driver parameters that describe how you should establish the right kinds of connections to your source of data.

When processing data on a server using ODBC technology and using a client application, two main stages are distinguished: setting a data source - creating and configuring a connection, as well as actually processing data using queries.

It is usually best if you are able to work with the OLE DB interface to create tools and utilities, or system-level developments that require high performance or access to SQL properties that are not available using ADO technology. Key features of the OLE DB specification provide full data access functionality. In SQL, the server database processor uses this interface for communication: between internal components, such as the storage processor and the relationship processor; Between SQL installations using remote stored procedures as an interface to other data sources for distributed queries.

When using OJSC technology, work with databases and tables is carried out using collections of objects. This provides great convenience in working with database objects.

At present, the technology of JSC is gradually superseded by ADO technology, which allows you to develop Web applications for working with databases. In general, ADO technology can be described as the most advanced application development technology for working with distributed client-server technology databases.

Internet-Based Database Systems

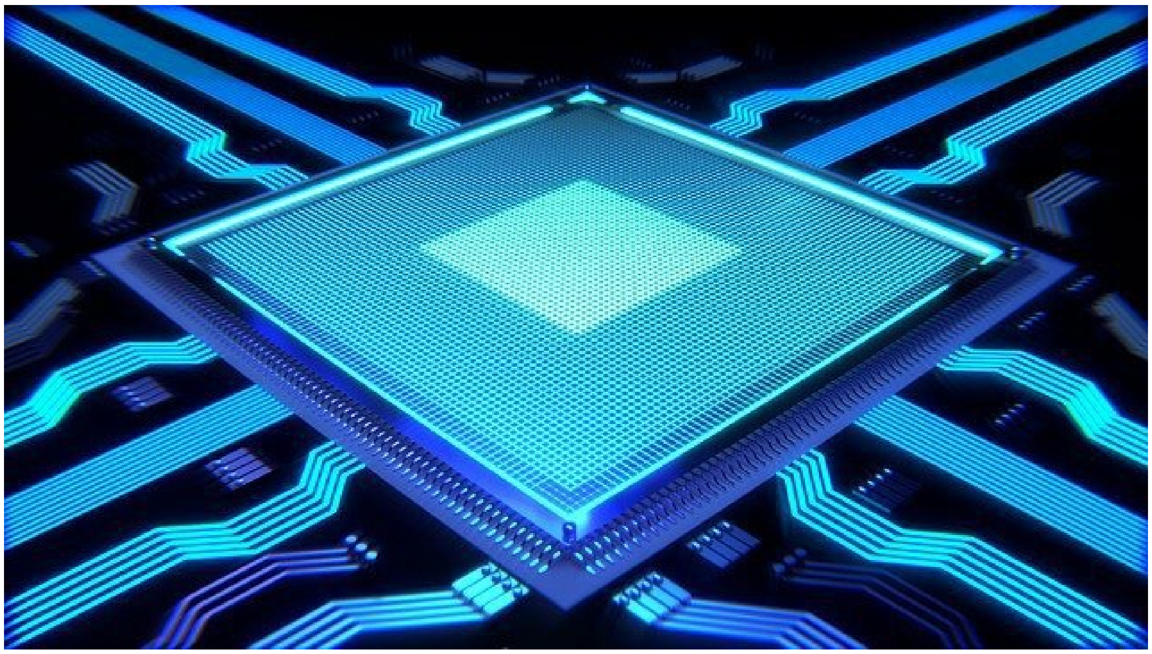
While the client to server kind of technology is widely popular and has worked well for a lot of people to use over time, you will also find that there are some programmers who have decided to add in some databases that are going to have more integration with the internet. This kind of system is going to allow the user to access the database when they go online, so they will be able to use their own personal web browser in order to check this out when they would like.

Along with this, the customer can go and check out the data if they would like, and they can even go online and make some changes to the account, check on the transactions that are there, check out the inventories, and purchase items. They can even make payments online if this is all set up in the right manner. Thanks to acting that we are able to base some of our databases online, we will find that it is possible to go online and let the customer access this database, even from their own home.

To help us work with these databases, we will just need to pick out the kind of web browser that we would like to go with, and then head on over to the website of the company that we go with. At times, you may need to have some credentials to get onto the account, but that is going to depend on the requirements of the company at the time. Then you can work with the search function in order to find some of the information hats you would like on the database.

You may find that a lot of these online databases are going to require us to have a login to have any kind of access, especially if there is some kind of payment requirement that goes with them. This may seem like a pain to work with, but it does add in some of that extra security that you are looking for to make sure your personal and financial information is always safe.

Chapter 2 The Operators



Definition

When we are talking about operators in your database, we are talking about the reserved characters and words that are mostly used in the WHERE clause of your statements.

Operators are used in order to perform operations in your statements, such as mathematical equations and comparisons, and they can even help to specify the parameters you want to set for the statements.

And of course, they are able to connect more than one parameter that is found within the SQL statement.

There are a few different types of operators that you will be able to use within your SQL statement including:

- Arithmetic operators
- Comparison operators
- Logical operators
- Operators for negating conditions

Each of these will work in a slightly different way to help you to get the results that you want.

Logical operators

The logical operators are going to be the keywords that you will use in your statements in order to perform comparisons.

Some of the logical operators you can use include:

- In—this operator will allow you to compare the value of specified literal values that are set. You will get a true if one or more of the specified values is equal to the value that you test.
- Like—the like operator is going to make it so that you can compare a value against others that are similar. You will usually combine it with the "_" or the "%" sign to get more done. The underscore is used to represent a number or a character and the % is used for several characters, zero, or one.
- Unique—with the unique operator, you will be able to take a look at one or more of your data rows and see if they are unique or not.
- Exists—you will need to use this operator to find the data rows that will meet the criteria that you put down. It basically allows you to put down some criteria and see if there are any rows that exist that meet with this.
- Between—you can use this operator in order to find values that will fall into a specific range. You will need to assign the maximum and the minimum values to make this work.
- Is null— the is null operator will allow you to compare the value of your choosing with a NULL one.
- Any and all—any and all values often go together. Any operator is going to compare a value against all of the values on a list. The list of values is to be set up with predetermined conditions. ALL, on the other hand, will compare the values that you select against the values that are in a different value set.

These are important for helping you to look up new data points within your database or to make some comparisons.

You can try out a few of these in your database and see what they come up with.

Comparison operators

The comparison operators are great if you would like to check the single values that are found inside of an SQL statement.

This particular category is going to be composed of some basic mathematical signs, so they are pretty easy to figure out.

Some of the comparison operators that you will be able to use within your SQL database includes:

- Non-equality—if you are testing the non-equality operator, you will use the “<>” signs. The operation is going to give you the result of TRUE if the data does not equal and FALSE if the data does equal. You can also use the “!=” to check for this as well.
- Equality—you will be able to use this operator in order to test out some single values in your statement. You will simply need to use the “=” sign to check for this. You are only going to get a response if the data that you chose has identical values. If the values aren’t equal, you will get a false, and if they are equal, you will get a true.
- Greater-than values and Less-than values—these two are going to rely on the “<” and “>” signs to help you out. They will work as stand-alone operators if you would like, but often they are more effective when you combine them together with some of the other operators.

Arithmetic operations

These operators are great if you are looking to do some mathematical operations in your SQL language.

There are four main type types that you will be able to use in your equations:

- Addition—you will just need to use the “+” sign to get started on using addition. For this example, we are going to add up the values for the overhead column and the materials column. The statement that works for this is: SELECT MATERIALS + OVERHEAD FROM PRODUCTION_COST_TBL.

- Subtraction—you can also do some basic subtraction when it comes to your SQL statements. You will simply need to use the "-" sign to do this.
- Multiplication—it is possible to do multiplication from your tables as well. You would need to use the "*" sign to do this.
- Division—this one will just need the "/" sign to get started.

You can combine a few of the arithmetic operations as well if this is what your process needs.

If you are going to use any combination of arithmetic operators, you need to remember that the principles of precedence are going to come into play.

This means that the syntax is going to take care of all the things that need to be multiplied first, and then the division, and then addition, and then subtraction.

It will not go from left to right, but it will go around to work with the symbols that you have up for the arithmetic operations, so keep this in mind when you are writing out your syntax.

Operators to use when negating conditions

In some cases, you will want to negate the operators in your database when you want to change the viewpoint of the condition.

You will want to use the NOT command in order to cancel the operator that it is used for.

Some of the techniques that you can use to do this include:

- Not equal—you can use this when you want to find the results that are not equal to something. For example, if you did use the "<>" or the "!=" symbols, you would get anything that was not equal to the value that you placed into the equation.
- Not between—you can also negate with the not between and the between operators. For example, if you are using a price, you could decide that you want to find the values that are not between 500 and 1000 or any value. This means that all the values that come up need to be either 499 and below or 1001 and above.
- Not in—if you will be able to use the not in command in order to negate the in operator. You will be able to pick some values that you don't want to have listed, and then the not in command will return you anything that doesn't fit in that amount.
- Not like—this is the term that you will use to negate the operator like. When you are using this, you are only going to see the values that are different from the one that you placed into the equation.
- Is not null—this one will work to negate the is null command. You will want to use this in order to check for the data that isn't null.

These are going to help you to set up some of the conditions of your search, so you are able to get a wide variety of information back without having to go through every little part of your database.

Conjunctive operators

There are going to be times when you will need to use several different criteria to make something happen. For example, if you are getting some results that are confusing from the database searches, you may decide to add in a few different criteria to see how it will turn up.

You will be able to combine together some different criteria in the statement in order to make the conjunctive operator.

Some of the conjunctive operators that you can use include:

- OR—you will use this statement to combine the conditions of the WHERE clause. Before this statement can take action, the criteria need to be separated by the OR, or it should be TRUE.
- AND—this operator is going to make it easier to include multiple criteria into the WHERE section of your statement. The statement will then only take action when the criteria have been segregated by the AND, and they are all true.

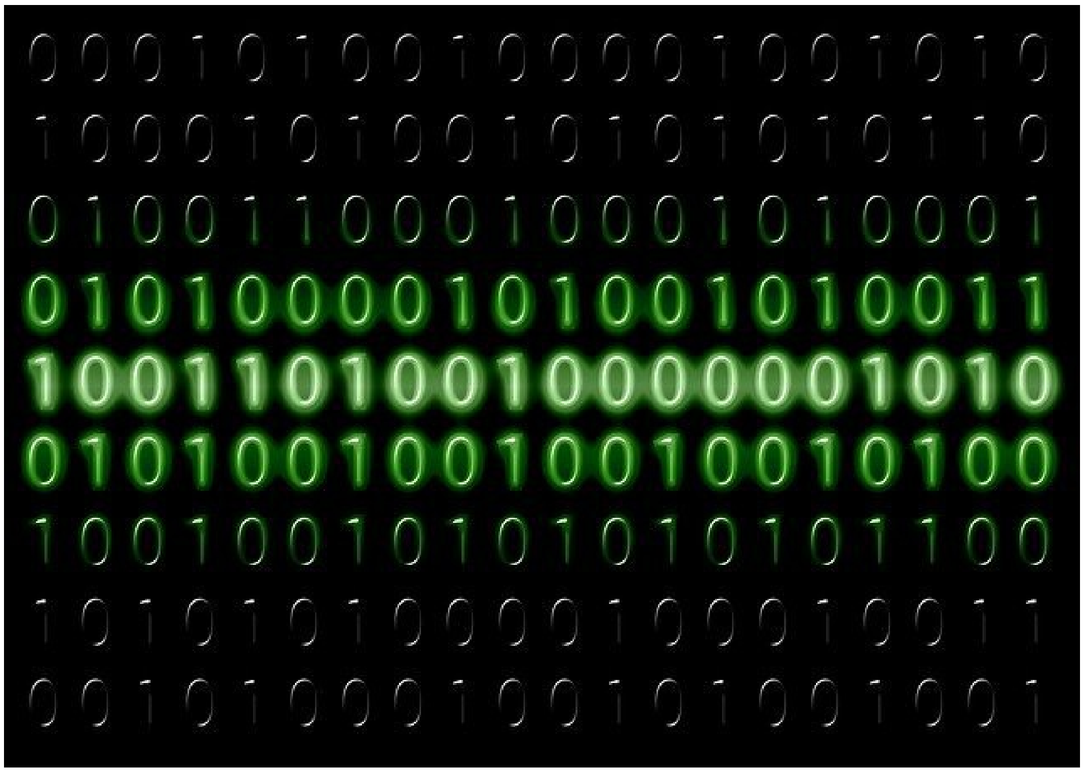
Keep in mind there are times when you will want to combine these operators together inside of the statement.

Just make sure that you are adding in parentheses to improve readability.

Concatenation of Strings

Using the addition operator "+", we can concatenate Strings of all types, such as declared variables with variable character types and strings defined at run time.

Chapter 3 The Data Definition Language



Data definition language statements are used to create database objects in the database system. They can be used to create table entities such as columns, views according to the specifications given. Data definition language statements are also responsible for the removal or deletion of columns, tables, and views. Data definition statements are user-specific and depend upon the parameters provided by the user.

SQL essential objects

SQL objects are an easy way to understand the essence of the language we are dealing with. Just like any programming language SQL too depends on different objects to make a sense for the programmers. Here are some of those for your better understanding of the subject.

a) Constants

Constants in SQL are also called as literal values and describes a constant value of any data. Constants are usually alphanumerical or a hexadecimal value.

b) Identifiers

Identifiers are a special feature developed in SQL for easily identifying entities, indices, and tables in the databases. Some of the well-known identifiers are @, #.

c) Comments

Comments in SQL just like all other programming languages are used to note down valuable information for programmers. Most database administrators use two hyphens side by side to write down comments.

d) Reserved keywords

These are the prescribed keywords defined by SQL software fundamentally and are not allowed to use by any user. Reserved keywords should be referred by SQL programmers from the documentation.

e) Data types

All the values that database tables and columns possess are usually curated into different types known as Data types. SQL offers various data types such as Numeric, Character, and Temporal, Binary and Bit data types.

f) File storage options

All SQL server data is stored using storage options such as File stream. It uses advanced tactics such as VARBINARY to store the information that is being dealt with. A lot of versions also use sparse columns to store data.

g) Functions

Functions are special characteristic operations that are destined to give results. SQL provides various aggregate and scalar functions for its users.

DDL for the database and Table creation

Usually, database objects that are present are divided into two types namely physical and logical. All physical objects such as data that are sent to a disk should be carefully observed. Logical objects are responsible for the creation of entities such as tables, columns, and views.

The primary DDL object that needed to be created is starting a database. Without creating a database there is no chance to create other entities.

Create a database

Here is the DDL statement for creating a Database:

CREATE DATABASE {enter parameters here}

The parameters that are used for database creation are as follows:

1. Name of the database - This parameter is used to create a valid name for the database. You can't use the same database name that is already present in the instance. Reserved keywords are also not allowed as the name of the database. All database names are stored in system files and can be easily changed if there is root access.
2. Attach - With this parameter, you can specify whether this database needs to be attached to the files of other databases or not.
3. File specifications - This parameter defines a lot of information such as name, size and internal information associated with the database.
4. Logging - You can use this parameter to define the files that can be used for the log management of the system files.

Using these parameters, you can easily create a database according to your requirements.

Create a database snapshot

Snapshot refers to the complete visual access of an entity. Database snapshot refers to a physical copy of all the information of a database. If a database consists of huge data it may take more time to create a database snapshot.

Here is the SQL statement for creating a database snapshot:

The parameters involving this DDL statement are.

1. Database snapshot name - Using this parameter can result in creating a name for the snapshot that you are going to create.
2. AS SNAPSHOT - Using this parameter you can enter the database name that you are willing to create a snapshot for. Remember that the database should be in the same instance for things to work fine.
3. Location - Using this parameter you can give a location to store

the snapshot. In default, the snapshot is stored in the temp directory of the system.

Attaching and detaching databases

There are different databases in a SQL server and are often required to use information from other databases. SQL provides options to attach data by using FOR ATTACH statement. Remember that if you are willing to attach data then everything present in that database will be imported. This works exactly in the process of merging.

SQL also provides detaching the data for its users. However, you need to look at the database configuration file ' detach_db' for making this work.

Note: Deal carefully with the system database files as they sometimes may result in catastrophic system failures. Always refer to documentation before detaching any data.

Create a table

Tables are logical entities present in databases. Usually, tables consist of rows and columns and deals with different data types.

Here is the SQL statement for creating a table:

CREATE TABLE {Name of the table} {Enter parameters here}

The parameters involved in this DDL statement are as follows:

1. Name of the table - By using this parameter you can give a name to the table. Advanced SQL server operations also provide options to change column names and data types.

- Schemas and Constraints - Schemas and constraints are complex topics. But for now, imagine them as an easy way to group objects and provide certain values.
- NULL - This is a special parameter that helps us to define whether null values are allowed or not in the columns. These type of user specifications are an example of constraint.
- Tempdb - This is a parameter where all the data, tables and columns created are automatically destroyed at the end of the

session.

Unique constraint DDL in tables

Constraints are a set of rules defined for maintaining the integrity of the database. These are therefore known as integrity constraints. By using constraints there is a lot of influence on databases and can result in good reliable data. With constraints, it is also easy to maintain a database.

Constraints are usually handled by DBMS but according to SQL standards, database programmers and administrators need to learn about constraints and define them for faster results.

Types of constraints:

There are primarily two types of constraints namely declarative and procedural constraints. We will mainly discuss declarative constraints such as Primary and Foreign keys using DDL statements.

- DEFAULT

This is a special constraint that is used to select columns or a group of columns that have the utmost same or identical values. Unique in the statement defines that the selected columns have updatable and unique values that can be easily recognized.

Note:

Remember that before defining any constraint you need to start with defining its column name.

Here is the SQL statement:

CONSTRAINT {Name of the column} UNIQUE [Enter parameters] {Enter values}

The parameters possessed by the following DDL SQL statement are:

1. Constraint - This will define the constraint we are going to use in this statement. The constraint will usually follow the column name that we are going to apply this on.
2. Type of Index - This parameter on a whole will explain whether this index is clustered or not. Clustering usually refers to canonically arranging index values.
3. Null - You can also use a parameter to define whether your constraint column can accept null values or not. By using this parameter, you are giving additional query information for any further Join operators.

- PRIMARY KEY

The primary key is a unique value that can be identified or used to query with other tables. Primary keys can be a foreign key for other databases. The primary key is usually essential for any database that is ready for query operations.

The primary key as remaining constraints can be used in both Creating and Altering database objects. This is the reason why they are explicitly used to create DDL and DML statements for changing the structure of the data.

Here is the SQL statement for Primary key:

CONSTRAINT {Name of the column} PRIMARY KEY {Enter the parameter values}

The parameters possessed by the following DDL SQL statement are:

1. Constraint - As said before the primary key is a constraint and it is important to define a primary key as a constraint to give expected results. You can also use the primary key without defining constraint but it may often result in corrupted data operations.

2. Column name - In the SQL operational statement you also need to define the column name of the table where the operational procedure is going to take place.
3. Null and Clustering - We will also define whether the DDL statement is willing to take Null values or not. Some statements will also give slight information about the clustering format.

Note:

Remember that dropping Primary key columns will not delete the created key values but will not result in successful retrieving.

- CHECK

This is one of the most important constraints available because it acts like a schematic operation and will restrict values to be entered. The check clause will explicitly check the constraints that are available and produce results.

Here is the SQL statement:

CONSTRAINT {Name of the column} CHECK {Logical expression}

The parameters possessed by the following DDL SQL statement are:

1. Check - This statement will evaluate the expression given to do a thorough check. Before creating a table or column the database system will look at this DDL statement and start checking the resources.
2. Replication - This is a statement that defines whether the database procedure should be continued or halted after the evaluation of an expression. A lot of database operations give errors if there is any hiccup during the process.
3. Logical expression - Logical expression is usually a combining factor of conditional and loop statements along with operators to undergo logical equality or operation. A lot of SQL programmers use this to counter-attack brute-force attacks coming from hackers. When the entities don't satisfy the logical expression then it will result in a halt.

Foreign key DDL in tables

Foreign keys are explicitly used to join two tables and can also be used to successfully modify the databases. It is an integral constraint and is used to actively rearrange the columns according to a logical expression.

Here is the SQL statement:

CONSTRAINT {Name of the column} FOREIGN KEY (Enter parameters here} REFERENCES {Logical values} {CHECK}

The parameters possessed by the following DDL SQL statement are:

1. Define constraint - First of all, in the statement, it is necessary to describe foreign keys as a constraint for maintaining the integrity of the database.
2. Define column - Now it is necessary to look down the column that can be worked as a foreign key. You might have already understood that a foreign key is a primary key to other tables.
3. Reference values - In this parameter you need to enter all the details that will result in successfully determining the foreign key automatically. It will check for the available primary keys and compare them with the later.
4. Logical Values - This constraint parameter is used to determine the logical entity of the foreign key. You can use it to check with any foreign expression values.
5. CHECK - A check can be used to look at any logical or equivalent values provided by the database administrator.

Chapter 4 Data Control Language (Dcl)

Definition

The DCL is another component of SQL that you should learn to use, and it is the commands that the user works with any time that they want to control who is allowed to get on the database. If you are dealing with personal information like credit card information, it is a good idea to have some limitations on who can get onto the system and get the information. This DCL command is used to help generate the objects that are related to who can access the information in the database, including who will be able to distribute the information. There are a few commands that are helpful when you are working on DCL including:

- Create synonym
- Grand
- Alter password

- Revoke

Roles and Privilege

You can identify one of the users by their authorization identifier, which is represented by the user name. However, this isn't the only way to identify someone who can perform certain operations on a table or a database. For instance, in a large company there are many users and setting up the privileges for each one can be time consuming and expensive. You can easily solve this problem by applying roles as the identifier.

This is an SQL feature that sets a role name to a certain user. It comes with a number of privileges attached, or none at all, and can be easily granted to anyone. You can even set the role to a group of users at once and save even more time. For instance, if the company has 20 salesmen, you can grant them all the privileges that fall into that category.

Keep in mind that not all of these functions are available in all SQL versions, or they might differ from the way we describe them. No matter which implementation or database management system you use, you should always read the documentation that comes with it. With that being said, let's see the syntax used to create a role:

```
CREATE ROLE SalesMan l
```

That's it. Now that we have the role, we can grant it to a number of people with the following syntax:

```
GRANT SalesMan to John;
```

Next, you can grant the privileges you want to the role. The syntax is the same. Now, let's see how to set a role to have the privilege of inserting data into a table. Type the following statement:

```
GRANT INSERT
        ON CLIENT
        TO SalesMan;
```

Now all the salesmen in the company can insert client information into the client table. Next, let's see how to allow the users to view data with the following lines:

```
GRANT SELECT
        ON ITEM
        TO PUBLIC;
```

You may notice we used the public keyword this time. This means that anyone who can use the system can now see the information inside the “item” table.

As you already know, tables change all the time. They need to be updated, new information needs to be inserted or deleted, and so on. This means that you need to give certain people the right to make such changes. However, you don’t want everyone to have this level of access. Here’s how to give a role the right to update a table:

```
GRANT UPDATE (Salary)
        ON SALARYRATE
        TO Manager;
```

Now the manager has the power to change the numbers in the salary column in order to adjust the income of the salesmen. However, this is the only column he has access to at the moment. He or she should also be able to update the Minimum and Maximum columns that represent the range of promotions. In order to enable the update privilege for every single column, you have two options. You either mention both columns in your syntax, or none of them. Both solutions lead to the same result.

Now, what if all of the businessmen in these examples close up shop, nobody is paying for their products or services, and the employees walk away or retire? Things always change and databases need to change with them because that’s life. Some of the data items, or even tables, become useless in these scenarios because the information they hold no longer reflects reality. Therefore we must delete these old records from the database and preserve only what is still accurate.

Make sure you are always aware of your actions and what you are deleting or you might cause some irreparable damage. With that in mind, let’s see the syntax:

```
GRANT DELETE
```

```
    ON EMPLOYEE
    TO MANAGER;
```

Now that manager is granted privileges over removing data from the employee table.

Referencing Tables

In SQL, you have the possibility to set the primary key of one table as the foreign key of another table. This means that the data from the first table can be accessed by anyone who has user privileges over the second table. This leads to a potential security problem because it creates a back door which anyone with malicious intent could access. Due to the referencing function, all it takes the unauthorized user is to find a table which references his target table.

Imagine a business having a table with everyone who will be fired in a month. Only certain people who occupy a management position have user privilege over this table. However, any employee could make an educated guess if the primary key of that table is named EmployeeID, or EmpID, or anything else along those lines. All he needs to do now is create his own table which uses the EmployeeID as its foreign key. Now he has access to view the table and see who will get fired. Here’s how the code looks:

```
CREATE TABLE SNEAKY (
EmployeeID INTEGER REFERENCES FIRING_LIST) ;
```

The next step this user needs to take is to use the insert statement in order to add a number of rows that match the employee ID’s. The new table called “sneaky” will only accept the data for those who are found on the firing list. The data that is rejected contains the names of those who aren’t to be fired.

To address this potential data breach situation, most database management systems include a solution and you should always implement it for this very reason. Make sure you extend reference privileges only to trustworthy users. Here’s how:

```
GRANT REFERENCES (EmployeeID)
ON FIRING_LIST TO MANAGER ;
```

Domains

Some security breaches are caused by domains, especially created domains. The user who creates the domain becomes its owner, just like in the case of tables and databases. When you create one, you can define it to hold the same data type and share identical restrictions with a set of table columns. Keep in mind that the columns that are part of the domain statement will inherit every characteristic that belongs to the domain. These features can be removed for certain columns, however, domains give you the ability to apply them with a single expression.

Domains are great to have when you are working with a large number of tables which hold columns with identical features. For instance, the database that belongs to a company can have multiple tables. Every one of them is likely to contain a “cost” column that holds a decimal data type that ranges anywhere between zero and 10,000. The first thing you need to do is create the domain that will wrap around the tables. It is recommended to take this step before you even create the tables. Furthermore, you need to mention the features of the columns when setting up the domain. Here’s what it looks like:

```
CREATE DOMAIN CostDomain DECIMAL (10, 2)
        CHECK (Cost >= 0 and Cost <= 10000) ;
```

Revoking Privileges

Sometimes you will have to take away these privileges from users who no longer fit the required criteria. Perhaps they leave the company, move to another department, and so on. You don’t want to allow someone who gets a job at a competing company to still have access to your data.

Revoking privileges is the easiest thing you can do under any of these circumstances. The best approach is to probably remove all of their privileges at once, unless they simply start performing other functions that require more access. In SQL this action uses the revoke statement. Essentially, it works exactly like the grant statement but in reverse.

Here’s the syntax:

```
REVOKE [GRANT OPTION FOR] privileges
ON object
FROM users [RESTRICT | CASCADE] ;
```

You can use the same syntax to remove all rights to access, or only the specific privileges instead. Take note of one major difference that does exist between REVOKE and GRANT. In this example, you need to apply a “restrict or cascade” line to your instructions. The purpose of cascade and restrict is to also revoke the privileges of any other user who received them from the person you are initially removing from your list of privileges.

Furthermore, you can revoke someone’s access with the addition of “grant option for” in order to remove some specific privileges that were granted by the main recipient to anyone else. However, he will keep those privileges himself. If you use this statement together with the cascade clause, then you will remove the access privileges from the main user, the authorizations he provided for anyone else, as well as the right to give anyone else such access in the future.

Whatever you choose to do, just make sure that those who have access to your database are responsible, trustworthy, and they have a reason to have them. Do not take the risk of security breaches lightly because all it takes is one mistake and someone takes advantage of your information or deletes it by mistake or in order to cause damage.

Chapter 5 Data Manipulation Language (Dml)

This is going to be the aspect of SQL that we are able to use when we would like to be able to modify some of the information about the objects that are found in the database that you are using. This is going to make it easier to delete the objects, update these objects, or insert something new inside of the database. This is going to give us a lot of freedom when we want to make sure that the right changes to our information in the database are added in without having to make a new part of the table for this database.

The DML that we are working within here, or the Data Manipulation Language, is going to come together with the statement of SELECT in order to get the right information from the database and then include other statements that change the state of the data. These operators are:

- INSERT - Adding records (rows) to the database table

- UPDATE - Updating data in a column of a table in our database.
- DELETE - Delete some of the records that are found in the table of our database.

INSERT Statement

When we are talking about the statement for INSERT, we are looking at the command that is able to add in some new rows to any table that we are working with. When we take a look at our current cause, the values of the columns have the potential to be literal constants, but we can also see that they are just a subquery result in some cases. With the first situation, we will work with a separate statement of INSERT to insert each row. The second case, as many rows will be inserted as the number returned by the subquery.

Operator Syntax

INSERT INTO < table name [(< column name ...)]

{VALUES (< column value ...)}

| < query expression

| {DEFAULT VALUES};

As you can see from the syntax presented, a list of columns is optional. If this is a part that is not present in our work, then the list of all values that are inserted must be done. This means that the programmer needs to be able to provide for us the values for each column that shows up in the table.

When we look back at this particular case, the order that we see the values fall in need to correspond to the specific column order that the statement for CREATE TABLE said along the way.

In addition, each of these values has to come from the same type of data, and it needs to also be the same type that is defined in the column or columns that we chose with our statement for CREATE TABLE. This may not seem important now but it is going to make a difference in some of the coding that we are able to do with this.

Insert Rows into a Table Containing an Auto-Increment Field

Many commercial products allow the use of auto-incrementing columns in tables, i.e. fields whose value is generated in an automatic manner when we want to put in the new records. These types of columns are going to turn into the primary keys of the table because they automatically provide uniqueness.

A good example of this type of column is going to be known as a sequential counter. With this one, when you do go through and insert a row, it is going to generate out a value of one that is greater compared to the previous value. This previous value is going to be obtained when inserting the previous row.

An auto-incrementing field is defined using the IDENTITY (1, 1) construct. In this case, the first parameter of the IDENTITY (1) property determines which value starts the count, and the second - which step will be used to increment the value. Thus, in our example, the first inserted record will have a value of 1 in the code column, the second - 2, etc.

Since the value is generated automatically in the code field, the operator

```
INSERT INTO Printer_Inc VALUES (15, 3111, 'y', 'laser', 2599);
```

will lead to an error even if you are not working with any row that is in your table that relies on this kind of code field equal to 15. Therefore, to insert a row into the table, we simply will not indicate this field in the same way as in the case of using the default value, i.e.

UPDATE Statement

There are going to be a number of times when we need to work with updating our table. If someone else is added to the database we work with, or we are trying to handle some changes to the table for our customers, for example, then handling the UPDATE command is one of the best ways to get this under control. The UPDATE statement modifies the data in the table. The command has the following syntax

```
UPDATE
SET {column name = {expression to calculate the column value
| Null
| DEFAULT}, ...}
[{WHERE}];
```

Using one operator, values can be set for any number of columns. However, in the same UPDATE statement, you can make changes to each column of the specified table only once. If there is no WHERE clause, all rows in the table will be updated.

If a column allows a NULL value, then it can be specified explicitly. In addition, you can replace the existing value with the default value that we want to work within our column.

If there is a reference to expression here, then we may find that this command is going to refer to all of the values that are currently in the table that is being edited.

It is also allowed to assign values of some columns to other columns. Suppose, for example, you want to replace hard drives of less than 10 GB in PC notebooks. In this case, the capacity of new disks should be half the amount of RAM available on these devices. This problem can be solved as follows:

```
 UPDATE Laptop SET hd=ram/2 WHERE hd<10
```

Naturally, we are going to find that the types of data we use for the ram and the hd columns have to be compatible. If you want to help cast these types, then you would work with the command of CAST to get it done. If you would like to change the data based on the contents that we are seeing in the columns that you want to work with, then you would want to work with the statement of CASE.

If, say, you need to put 20 GB hard drives on PC notebooks with memory less than 128 MB and 40-gigabyte ones on other PC notebooks, then you can write this request:

You can also use subqueries to calculate column values. For example, you need to equip all PC notebooks with the fastest processors available. Then you can write:

```
 UPDATE Laptop
SET speed = (SELECT MAX(speed) FROM Laptop)
```

In this case, the code will not be executed because the auto-incrementing field does not allow updates, and we will receive the corresponding error message. To accomplish this task nevertheless, one can proceed as follows.

In Transact-SQL, the UPDATE statement ng> extends the standard by using the optional FROM clause. This proposal specifies a table that provides the criteria for the update operation. Additional flexibility is provided by the use of table join operations.

DELETE Statement

This is a good statement to learn a little bit about. The statement of DELETE is going to be there to help us get rid of statements from any kind of table or cursor or even view that we want, and in a few of them, the operator's action extends to those base tables from which data was extracted to these views or cursors. The delete operator has a simple syntax:

```
DELETE FROM [WHERE];
```

If you are working with the WHERE command here, and it is missing, then all of your rows in the view or the table will be gone. This is true if the table is something we can update. You can also perform this operation (deleting all rows from a table) in Transact-SQL more quickly using the command

TRUNCATE TABLE

However, there are a number of differences in the implementation of the TRUNCATE TABLE command compared to using the DELETE statement, which should be kept in mind:

1. It is not journalized to delete individual rows of a table. Only the release of pages that were busy with table data is written to the log.

2. Do not work out triggers. As a result, this command is not applicable if there is a foreign key reference to this table.

3. The counter value (IDENTITY) is reset to the initial value.

An example. You want to remove all notebooks with a screen size less than

12 inches from the Laptop table.

```
DELETE FROM Laptop
WHERE screen<12
```

All notebooks can be deleted using the operator.

```
DELETE FROM Laptop
```

Or

FROM

Using a table type source, you can specify the data that is deleted from the table in the first FROM clause.

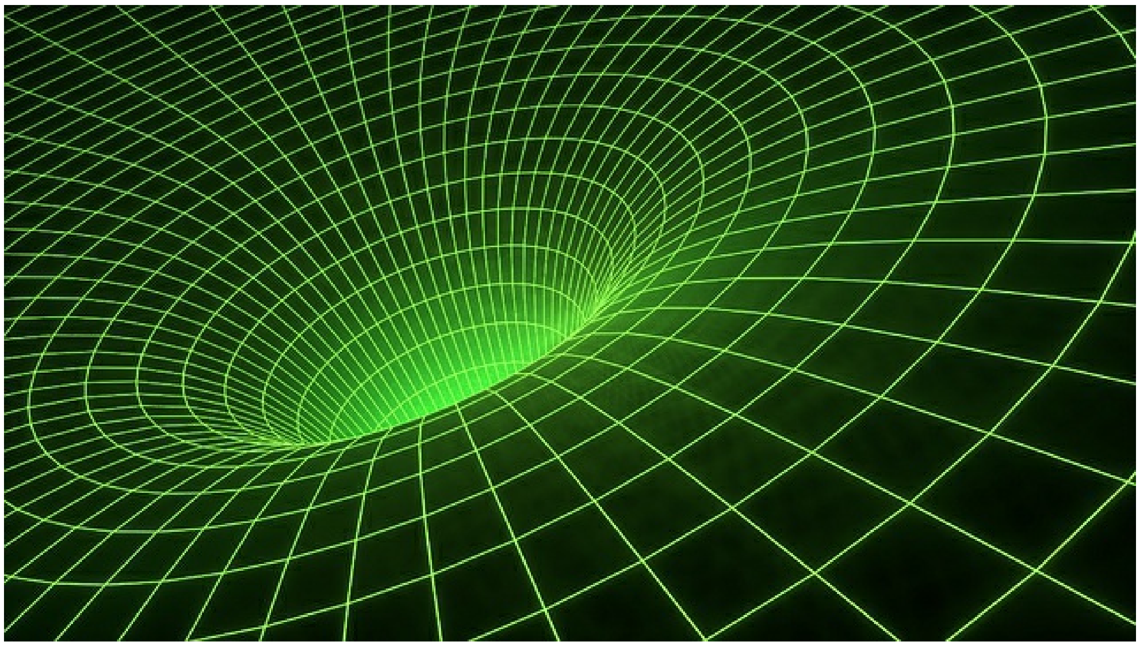
Using this clause, you can join tables, which logically replaces the use of subqueries in the WHERE clause to identify deleted rows.

Let us explain what was said by an example. Suppose you want to remove those PC models from the Product table for which there are no rows that would correspond to it in the PC table either.

When we stick with the syntax that is considered standard in this one, this is something that we are able to solve with the help of the DELETE FROM code if we would like.

An external join is used here, as a result of which the pc.model column for PC models that are not in the PC table will contain a NULL value, which is used to identify the rows to be deleted.

Chapter 6 Data Query Language (Dql)



A query, is, simply put, a set of instructions you give in order to change a table within a database. The ones we will be looking at are primarily the UPDATE and DELETE queries.

Both of these queries are very self-explanatory. The UPDATE query will take the information currently within a table, and change it to whatever you desire. The DELETE query is quite like an UPDATE query just with “null” instead of what you wanted to change it to. It will delete any entries that you wish.

It is important to note that while these queries are extremely important, they’re also inefficient. You’ll learn that there are much more efficient ways to do what these queries do, and at a much larger scale.

With that being said, they are still a must-learn for budding developers. They help you learn the fundamental blocks that advanced SQL is based on. After all, every business owner has heard horror stories of developers that only know higher-level material, and low-level techniques become their downfall.

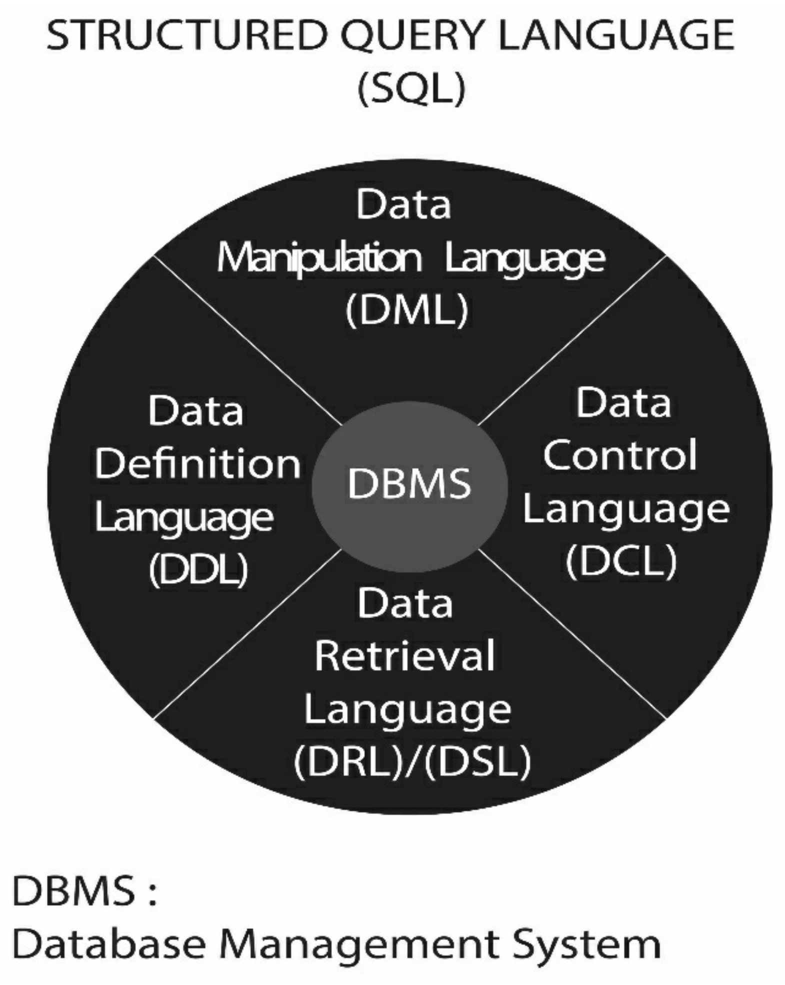
These queries will primarily be useful in debugging and lower-level positions. Otherwise, they’re only useful for manually editing smaller tables, at which point you might as well use Excel instead of SQL.

On the other side of the coin, we have the TOP query. The TOP query, rather than actually changing the information inside a table, shows you only specific entries from a table. To be precise, the TOP query will show you the topmost N or topmost N % of a given table.

This is especially useful if you’re using SQL for math or science, in which case it can make recurring certain functions very easy.

The LIKE clause is meant to compare different objects/strings while the ORDER BY clause will sort a table in ascending or descending order, as you feel fit.

These are two extremely powerful tools that you'll use throughout your career as a SQL developer, so let’s dive right in!


STRUCTURED QUERY LANGUAGE (SQL)
Data Manipulation Language (DML)
Data Definition Language (DDL)
DBMS
Data Control Language (DCL)
Data Retrieval Language (DRL)/(DSL)
DBMS : Database Management System

UPDATE & DELETE Query

The UPDATE query in SQL is mainly intended to be used when modifying the records that are already in a table. What’s worth noting here is that if you use the WHERE clause together with the UPDATE query, only the rows you selected with the WHERE clause will be updated. If you don’t do this, then every row inside the table will be equally affected.

The Syntax for an UPDATE query within a WHERE clause is:

```
UPDATE name_of_table

SET column01 = value01, column02=value02…, columnN = valueN

WHERE[your condition];
```

When using the UPDATE query, you’ll be able to combine any N number of conditions by using the two operators you should be familiar with already: the AND OR operators.

Take this for example:

In the following table, there are customer records listed, and you’re trying to update the record.

You can combine N number of conditions using the AND or the OR operators.

```
+--+----------+-----+-----------+----------+

| ID | NAME     | AGE | ADDRESS   | SALARY   |

+--+----------+-----+-----------+----------+

| 1 | Ilia         | 19 | Uruguay     | 1500.00 |

| 2 | Frank      | 52 | France        | 200.00 |

| 3 | Jim         | 53 | Serbia         | 8040.00 |

| 4 | Martini| 54 | Amsterdam| 9410.00 |

| 5 | Jaffa     | 66 | Podgorica    | 55200.00 |

| 6 | Tim        | 33 | Prune          | 1200.00 |

| 7 | Kit          | 24 | England       | 700.00 |

+--+----------+-----+-----------+----------+
```

Let’s say you want to update the address of the customer with the ID number

2, you would do it as such:

```
SQL> UPDATE CUSTOMER
SET ADDRESS = ‘Toms’
WHERE ID = 2;
```

Now, the CUSTOMERS table would have the following records −

```
+--+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+--+----------+-----+-----------+----------+
| 1 | Ilia         | 19 | Uruguay     | 1500.00 |
| 2 | Frank      | 52 | Toms        | 200.00 |
| 3 | Jim         | 53 | Serbia          | 8040.00 |
| 4 | Martini| 54 | Amsterdam| 9410.00 |
| 5 | Jaffa     | 66 | Podgorica    | 55200.00 |
| 6 | Tim        | 33 | Prune           | 1200.00 |
| 7 | Kit          | 24 | England       | 700.00 |
+--+----------+-----+-----------+----------+
```

Now, if instead, let’s say you want to change the salaries and addresses of all your customers, then you won’t need to use the WHERE clause. The UPDATE query will handle it all by itself. This can sometimes save quite a bit of time, let’s look at the following example:

```
SQL> UPDATE CUSTOMER
SET ADDRESS = 'Toms', SALARY = 1000.00;
```

Now, CUSTOMERS table would have the following records −

```
+--+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+--+----------+-----+-----------+----------+
| 1 | Ilia         | 19 | Toms    | 1000.00 |
| 2 | Frank      | 52 | Toms        | 1000.00 |
| 3 | Jim        | 53 | Toms        | 1000.00 |
| 4 | Martini| 54 | Toms | 1000.00 |
| 5 | Jaffa     | 66 | Toms   | 1000.00 |
| 6 | Tim       | 33 | Toms         | 1000.00 |
| 7 | Kit         | 24 | Toms      | 1000.00 |
+--+----------+-----+-----------+----------+
```

If you can't really tell where this would be useful, don't worry. There are countless examples from around the corporate world. This will let you replace any given thing in a matter of minutes. While it might not seem practical in a table with 7 people in it, imagine you're Microsoft and instead of 4 columns and 7 rows, you have 50 columns and 7000 rows, that isn't very practical to do by hand now is it?

Now let's take a look at the DELETE query. You can probably imagine what it does. It helps you delete certain records from a table. Now, obviously you don't want your whole table gone, so you should probably use the WHERE clause together with it, so you don't accidentally end up deleting, well, everything. When you use the WHERE clause with the DELETE query, only what you've selected will be deleted. Kind of like clicking on a file and pressing delete on your keyboard.

Let's turn our eyes to the syntax a bit, let's use the customers example again for it.

```
DELETE FROM name_of_table
WHERE [your condition];
```

Similarly to the UPDATE query, you can use this in conjunction with the OR and operators to get more complex and precise results.

Let's look at the past example we used:

```
+--+----------+-----+-----------+----------+
| ID | NAME | AGE | ADDRESS   | SALARY   |
+--+----------+-----+-----------+----------+
| 1 | Ilia         | 19 | Toms    | 1000.00 |
| 2 | Frank      | 52 | Toms        | 1000.00 |
| 3 | Jim         | 53 | Toms         | 1000.00 |
| 4 | Martini| 54 | Toms | 1000.00 |
| 5 | Jaffa     | 66 | Toms   | 1000.00 |
| 6 | Tim        | 33 | Toms          | 1000.00 |
| 7 | Kit          | 24 | Toms      | 1000.00 |
+--+----------+-----+-----------+----------+
```

Let's say you want to erase a customer. Maybe they stopped shopping at your locale? Moved to a different state? Whatever reason it may be, this is how you could do it, let’s say the customer’s number is 7.

```
SQL> DELETE FROM CUSTOMER
WHERE ID = 7;
```

As you can see, this is quite similar to the UPDATE query, and they really are similar. If you need some help in thinking about the DELETE query, think about it as an UPDATE query that updates with empty spaces (this isn’t entirely accurate, but it helps).

Now the customers table would look like this:

```
+--+----------+-----+-----------+----------+
| ID | NAME | AGE | ADDRESS   | SALARY   |
+--+----------+-----+-----------+----------+
| 1 | Ilia         | 19 | Toms    | 1000.00 |
| 2 | Frank      | 52 | Toms        | 1000.00 |
| 3 | Jim      | 53 | Toms      | 1000.00 |
| 4 | Martini| 54 | Toms | 1000.00 |
| 5 | Jaffa    | 66 | Toms   | 1000.00 |
| 6 | Tim      | 33 | Toms      | 1000.00 |
+--+----------+-----+-----------+----------+
```

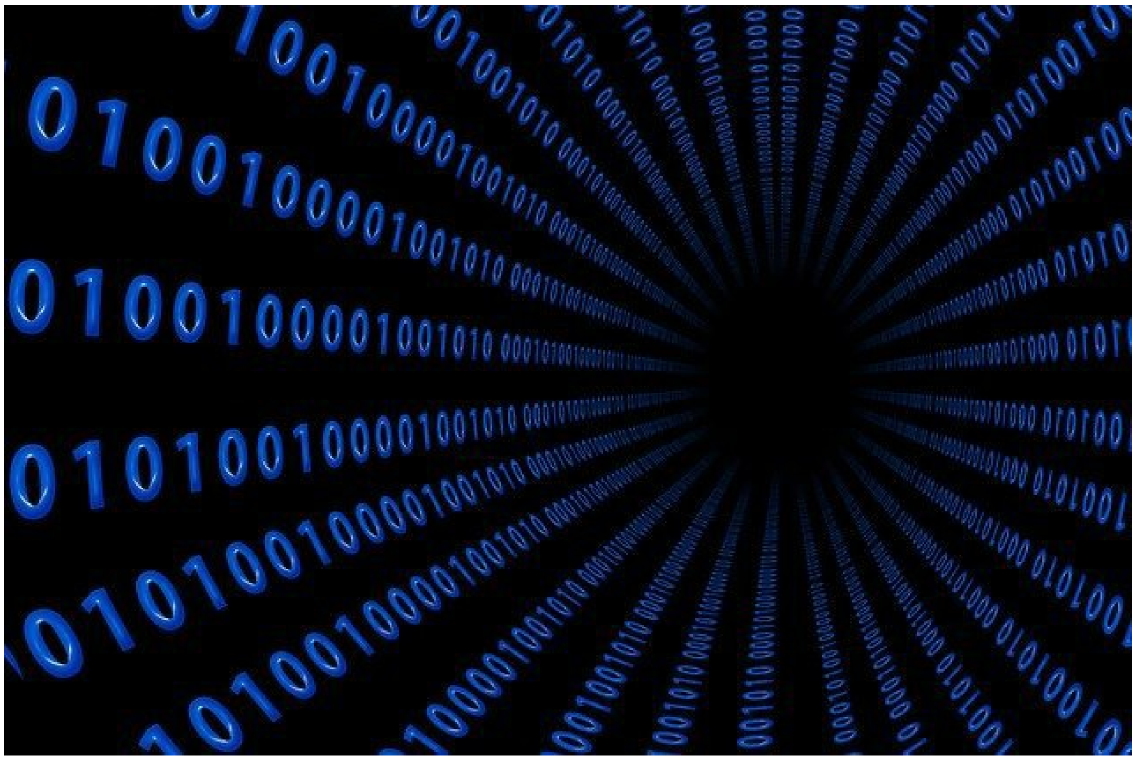
As you can see, all that changed is the 7th column is now empty. If you’ve been wondering how it would look if you hadn’t used the WHERE operator, this is how:

```
+--+----------+-----+-----------+----------+
+--+----------+-----+-----------+----------+
```

That’s right! Using the DELETE query without a WHERE operator results in an empty table.

Now, hopefully you won’t be using the DELETE query too much wherever you end up working, but it can be useful for a variety of things. For example, when your company is moving servers, or simply purging outdated entries.

Chapter 7 Data Retrieval Language (Drl)/Data Selection Language (Dsl)



These are the good commands that you can use in SQL any time that you want to keep track of as well as manage the transactions that will happen inside the database. If you sell a product of any kind on your website, for example, you will need to use the transactional control commands to help keep track of these commands, keep track of the amount you are making, and to keep track of all the other things that you will need when working with those transactions. There are a few things that you will be able to do with these transactional control commands including:

- Commit—this one is going to save all the information that you have about the transactions inside the database.
- Savepoint—this is going to generate different points inside the groups of transactions. You should use this along with the Rollback command.
- Rollback—this is the command that you will use if you want to

go through the database and undo one or more of the transactions.

- Set transaction—this is the command that will assign names to the transactions in your database. You can use it to help add in some organization to your database system.

As we can see as we go through this process, all of these commands are going to be important to what we want to get done within the code that we are writing out. These can help us to get some of the specific results that we are looking for inside of our database. And we will be able to take a bit more time to explore these and learn more about them as we go through this guidebook so that we can get a deeper look at what they are all about and how we are able to use them for some of our own needs as well. But this is a great way for us to get an introduction to what they are all about and how we can use them for some of our own needs as well.

Selecting rows and columns

Selecting a datum from your database can be done through the SELECT key. You only have to specify the data you want to select.

Step #1–Choose the SELECT statement

Choose SELECT to identify your SQL command.

Step #2– Choose the column

Choose the specific column where you want to retrieve the data.

Example: SELECT“column_name”

Step #3–Use the asterisk * to select all columns

If you want to select all columns use *, or you can also choose as many columns as you want.

Example: SELECT“column_name1”

[“column_name2”, “column_name3”]

Step #4–Add FROM and the table name, where the data will come from

You can enclose the identified columns and where conditions with open and close square brackets [ ], but this is optional.

```
Example:  SELECT“column_name”
                       [“column name”, “column name”]
                         FROM ‘table name”
                       WHERE “colum_name”;
```

You can also write the above example in this way:

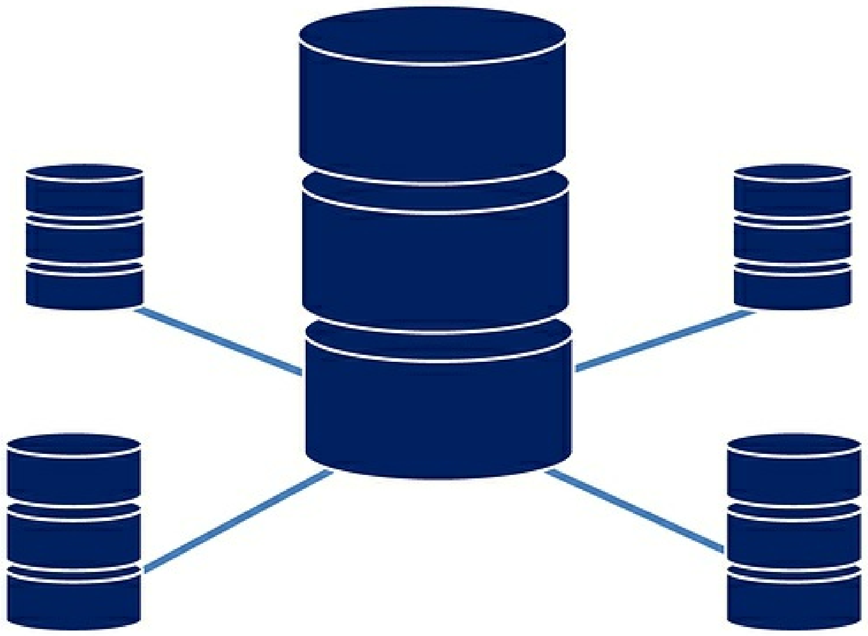
```
Example: select column name, column name, column name
                        From table name
                        Where column name;
```

Step #5–Specify the “CONDITION”

You can specify the condition through the common operators

```
Example #1:  SELECT“column_name”
                       [“column_name”, “column name”]
                         FROM ‘table name”
                       [where“colum_name” “condition”];
```

Chapter 8 Stored Program



Definition

The Stored Procedure is a program written in the SQL language that can realize specified functions. Secondly, this program is compiled by SQL Server and stored in the SQL Server database. Users can call these stored procedures with specified functions by passing their names and parameters. Stored procedures are also database objects. People usually use stored procedures to improve the security of databases and reduce the amount of network communication data.

Advantages of Stored Procedures

a) Fast execution speed and high efficiency:

Because SQL Server 2008 compiles stored procedures into binary executable code in advance, SQL Server 2008 does not need to compile stored procedures when running stored procedures, which can accelerate the execution speed.

b) Modular programming:

After the stored procedure is created, it can be called many times in the program without rewriting the T-SQL statement. After the stored procedure is created, the stored procedure can also be modified, and after one modification, the results obtained by all programs calling the stored procedure will be modified, thus improving the portability of the programs.

c) Reduce network traffic:

Since stored procedures are a set of Transact-SQL stored on the database server, only one stored procedure name and parameters are needed when the client calls them, and the traffic transmitted on the network is much smaller than that of this set of complete Transact-SQL programs, thus reducing network traffic and improving operation speed.

d) Security:

Stored procedures can be used as a security mechanism. When users want to access one or more data tables but do not have access rights, they can design a stored procedure to access the data in these data tables. However, when a data table does not have permission and permission control is required for the operation of the data table, stored procedures can also be used as an access channel to use different stored procedures for users with different permissions.

In SQL Server, stored procedures can be divided into three categories.

a) Default active procedures:

Default active procedures are generally prefixed with "sp_" and are special stored procedures created, managed and used by SQL Server itself. Do not modify or delete them. Physically, the Default active procedures are stored in the Resource database, but logically, the Default active procedures appear in the sys architecture of the system database and user-defined database.

b) Extended Stored Procedures:

Extended stored procedures are usually prefixed with "xp_". Extended stored procedures allow you to create your external stored procedures using other editing languages (such as C#) and their contents do not exist in SQL Server but exist separately in the form of DLL. However, this function may be abolished in future versions of SQL Server, so try not to use it.

c) User-defined Stored Procedures:

Stored procedures created by the user, which can input parameters, return tables or results, messages, etc. to the client, or return output parameters. In SQL Server, user-defined stored procedures are divided into Transact-SQL stored procedures and CLR stored procedures.

Transact-SQL stored procedures, which store a set of Transact-SQL statements and can accept and return user-provided parameters; CLR stored procedures, which are references to Microsoft's. NET Framework Common Language Runtime (CLR) methods, can accept and return user-supplied parameters. CLR stored procedures are implemented as public static methods in .NET Framework programs.

In SQL Server, you can use SQL Server Management Studio and Transact-SQL languages to create stored procedures. When creating stored procedures, you must determine the three components of the stored procedures.

- Input parameters and output parameters.
- Transact-SQL statements executed in stored procedures.
- The returned status value indicating whether the execution of the stored procedure was successful or failed.

How to create stored procedures

Use CREATE PROCEDURE statement to create stored procedures

You can create stored procedures using either SQL statements or SQL Server Management Studio. However, the key to creating stored procedures lies in writing Transact-SQL programs. Therefore, only the method of creating stored procedures using SQL will be introduced here.

The specific syntax format is as follows:

CREATE {Enter the logic here} PROCEDURE

Create Stored Procedures with Input Parameters

Most stored procedures used in databases have parameters. The function of these parameters is to transfer data between stored procedures and calling programs (or calling statements). When transferring data from the calling program to the stored procedure, it will be received by the input parameters in the procedure, while when transferring data from the stored procedure to the calling program, it will be transferred through the output parameters.

Creating Stored Procedures with Output Parameters

If you want to pass the data in the stored procedure to the caller, you should use the output parameters in the stored procedure. The OUTPUT parameter is specified with the output keyword.

Create Stored Procedures with Multiple SQL Statements

Stored procedures can have multiple SQL statements and programming statements provided by DBMS. At this time, after calling the stored procedure, multiple query result sets will be returned.

Modify Stored Procedures

When using stored procedures, once it is found that the stored procedures cannot complete the required functions or the functional requirements change, the originally stored procedures need to be modified.

Modify stored procedures with ALTER PROCEDURE statement

ALTER PROCEDURE statement is provided in the Transact-SQL language to modify stored procedures.

Its syntax code is as follows:

ALTER {Enter the logic here} PROCEDURE

How to delete a stored routine?

A stored routine should be deleted when it is no longer needed. There are also two deletion methods, one is using SSMS interface operation and the other is using the deletion statement provided by Transact-SQL language.

Use the DROP PROCEDURE statement to delete stored procedures

To delete a stored procedure, use the DROP PROCEDURE statement, which has the following syntax format.

DROP PROCEDURE {Enter the name of the routine}

However, understand that you can easily backup all the stored routines that you have created.

Default active procedures

Many management activities in SQL Server are performed through a special procedure called a system stored procedure. Default active procedures with the prefix "sp_" can easily be executed. You can execute a system stored procedure from any database without using the master database name to fully qualify the name of the stored procedure.

Stored functions

There is a small difference between stored procedures and stored functions that is stored functions can be used in expressions to execute statements whereas stored procedures are an individual entity. Stored functions help other entities to make the task get done.

Here is the syntax:

CREATE FUNCTION {Enter the name here} {Enter parameters}

By using this syntax, you can create your stored functions. Parameters consist of data types and index information.

Chapter 9 Join

Definition

A Structured Query Language (SQL) join is an instruction that you can use to combine the information from two different tables. Before we look at the details of an SQL join, let us look at why you would want to use it. Let us look at two tables that provide information about customers and orders.

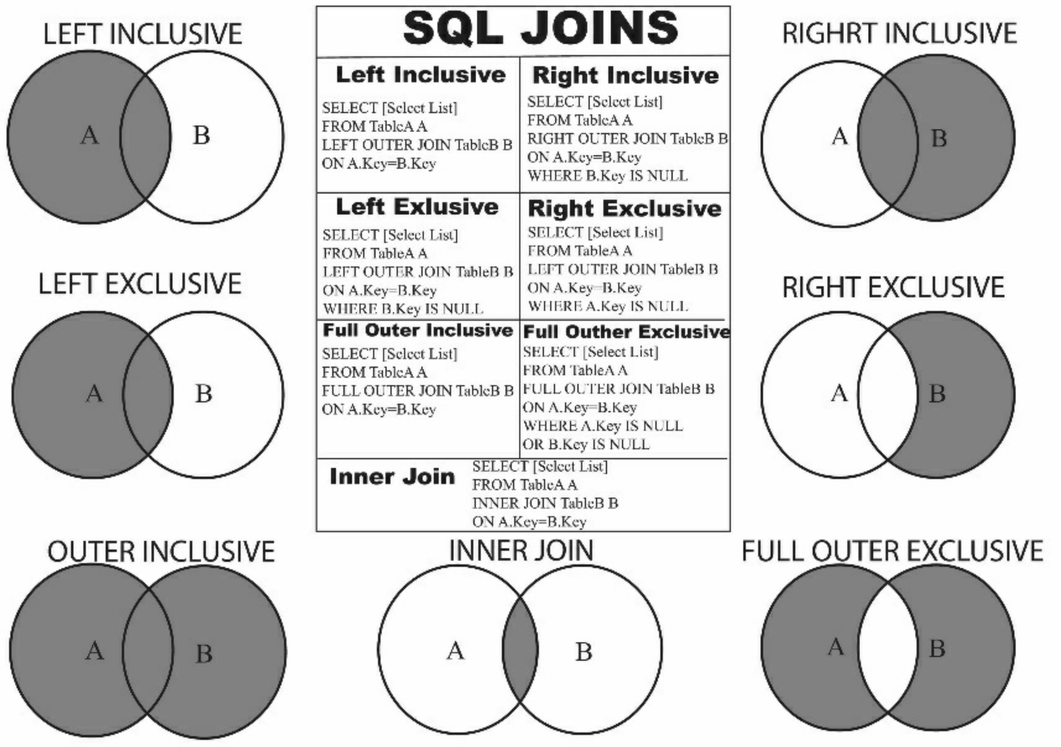
Basics of SQL Joins

The joins will let you collate or merge more than one table with the use of primary identifiers. To understand this, let us look at the two tables we created earlier: all of them share and an id column. You might be wondering what the purpose of joining in SQL is?

Given the constraints associated with normalization, you might miss out on some of the vital information in one table. The process of normalization isn't only recommended but also a must to ensure consistency, bring down redundancy levels as well as preventing various insertions and updating issues

For you two to join two tables, they must be an everyday thing between those two tables. How can this be achieved? Having the two tables share a column with the same name? Or what does that statement exactly mean?

The tables that are supposed to be joined might not share a column that has the same name, but in a real sense, the two tables should have a common name. For instance, each set of data should have a thing in common. In this case, you cannot merge two or more tables that have a column with a similar name but with a completely different set of data types.



Left Inclusive

```
SELECT [Select List]
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key=B.Key
```

Right Inclusive

```
SELECT [Select List]
FROM TableA A
RIGHT OUTER JOIN TableB B
ON A.Key=B.Key
WHERE B.Key IS NULL
```

Left Exlusive

```
SELECT [Select List]
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key=B.Key
WHERE B.Key IS NULL
```

Right Exclusive

```
SELECT [Select List]
FROM TableA A
LEFT OUTER JOIN TableB B
ON A.Key=B.Key
WHERE A.Key IS NULL
```

Full Outer Inclusive

```
SELECT [Select List]
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key=B.Key
```

Full Outher Exclusive

```
SELECT [Select List]
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key=B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```

Inner Join

```
SELECT [Select List]
FROM TableA A
INNER JOIN TableB B
ON A.Key=B.Key
```

Types of Joins

JOINs are SQL’s way to combine together multiple data points from N tables and put it all into a database. A JOIN is a way through which you combine fields from at least two tables by utilizing the values that they have in common.

An excellent example of the way JOIN’s function is:

-- Table One

```
+--+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+--+----------+-----+-----------+----------+
| 1 | Ilia         | 19 | Uruguay     | 1500.00 |
| 2 | Frank      | 52 | Toms        | 200.00 |
| 3 | Jim         | 53 | Serbia         | 8040.00 |
| 4 | Martini| 54 | Amsterdam| 9410.00 |
| 5 | Jaffa     | 66 | Podgorica    | 55200.00 |
| 6 | Tim        | 33 | Prune          | 1200.00 |
| 7 | Kit          | 24 | England       | 700.00 |
+--+----------+-----+-----------+----------+
-- Table two
+-----+--------------------+-------------+--------+
|OID | TIME              | CUSTOMER_ID | NUMBER|
+-----+--------------------+-------------+--------+
| 102 | 2004-20-08 00:00:00 |           3 |   3000 |
| 100 | 2004-20-08 00:00:00 |           3 |   1500 |
| 101 | 2004-21-08 00:00:00 |           2 |   1560 |
| 103 | 2004-19-1 00:00:00 |           4 |   2060 |
+-----+--------------------+-------------+--------+

-- You can join these two together through the following syntax:
SQL>SELECT IDS, NAMES, AGES, AMOUNTS
FROM CUSTOMER, BUYS
WHERE CUSTOMER.ID=BUYS.CUSTOMER_ID;
-- This produces the following output:
+----+----------+-----+--------+
| ID | NAME     | AGE | AMOUNT |
+----+----------+-----+--------+
| 1 | Ilia | 23 |   3000 |
| 1 | Ilia | 23 |   1500 |
| 2 | Jaffa | 25 |   1560 |
| 4 | Jaffa | 25 |   2060 |
+----+----------+-----+--------+
```

That was quite simple, no? JOINs in SQL can be performed in many ways. You can use a multitude of operators, ranging from =, too!= and the greater and lesser symbols. You can also use some SQL keywords like BETWEEN and NOT to accomplish this. With that being said, the most common operator you'll be using is the = sign.

Now, let's take a look at the multitude of JOINs in SQL, shall we?

Let's start off with the most used join, which is the INNER JOIN. It is used in almost every larger SQL database, and is often referred to as an SQUIJOIN because it treats both of the tables that are being joined equally.

This JOIN will make a new table by simply combining the values found within the rows and columns of the tables you're joining. The JOIN will take every row of table 1 and compare and contrast to every row of table 2. It'll find all rows that can be JOIN-ed, and once it's done with that, it's going to do the same with the columns. The output of a JOIN is simply the product of these values.

The syntax of an INNER JOIN is:

```
SELECT table01.column01, table02.column02...
FROM table01
INNER JOIN table02
ON table01.common_field = table02.common_field;
```

If you're interested in how this works, and what it would give out, try it out! Make two tables and check what the script outputs.

Now let's take a look at the 2nd join you'll be learning about- the LEFT JOIN. The LEFT JOIN is well, precisely what it sounds like...left. It serves to output every row from the table on the left. This means that if the table on the left has members, even if there are no ON clause matches in the right one, this JOIN ill still have an output. This means that you can essentially combine the left table with an empty table, and still get a valid output.

This is one of the harder joins to debug, due to its lack of a fail-safe. The SQL standard describes the LEFT JOIN as follows:

SELECT table01.column01,table02.column02...table0N.column0N

FROM table01

LEFT JOIN table02

ON table01.common_field = table02.common_field ;

Here, the ON condition that needs to be fulfilled would be filled in by you, in accordance to what you need to be done.

The RIGHT JOIN in SQL, is the exact opposite of the LEFT JOIN. It will output all the rows from the right table, regardless of the presence of matches in the left one. This makes it just as difficult to debug as the LEFT JOIN, because even if the ON clause has 0 matches, you will still get an output. This will just give you a NULL value for every column in the table on the left.

What this means is that the return of the RIGHT JOIN is the values from the right side, together with the matches with the left table.

Its syntax is:

SELECT table01.column01,table02.column02...table0N.column0N

FROM table01

RIGHT JOIN table02

ON table01.common_field = table02.common_field ;

Now, the FULL JOIN in SQL is merely the combination of these two. It will give you all records from both the left and the right table. It is the hardest JOIN to debug, because none of the tables need to have actually been filled. You can literally JOIN two tables with no elements in either of them, and your only hint to anything being wrong would be the fact that it returns full NULL values. The syntax for this type of JOIN is:

```
SELECT table01.column01,table02.column02...table0N.column0N
FROM table01
FULL JOIN table02
ON table01.common_field = table02.common_field ;
```

Now, there are some databases such as MySQL which don’t actually support the FULL JOIN. If you find yourself working with one of these heretic machines, then you can use the UNION ALL clause to bind together the LEFT JOIN and RIGHT JOIN to obtain the equivalent of a FULL JOIN. The syntax for this is:

```
SQL> SELECT  A, B
   FROM A
   LEFT JOIN B
   ON A.ID = B.A_ID
UNION ALL
   SELECT  A, B, C, D
   FROM A
   RIGHT JOIN B
   ON A.ID = B.A_ID
```

The SQL SELF JOIN is meant to connect a given table to itself, pretending as if it were 2 tables. This serves the purpose of changing the name of at least 1, but up to N tables in a single SQL statement.

The basic syntax for a SELF JOIN to join table a to itself is:

```
SELECT a.name_of_column, b.name_of_column...
FROM table01 a, table01 b
WHERE a.common_field = b.common_field;
```

-- In this case, the WHERE clause can be assigned any type of SQL expression. It is whatever you want to join based on, a condition, think of it as a temporary restriction on the values.

```
SQL> SELECT  a.A, b.B, a.A
```

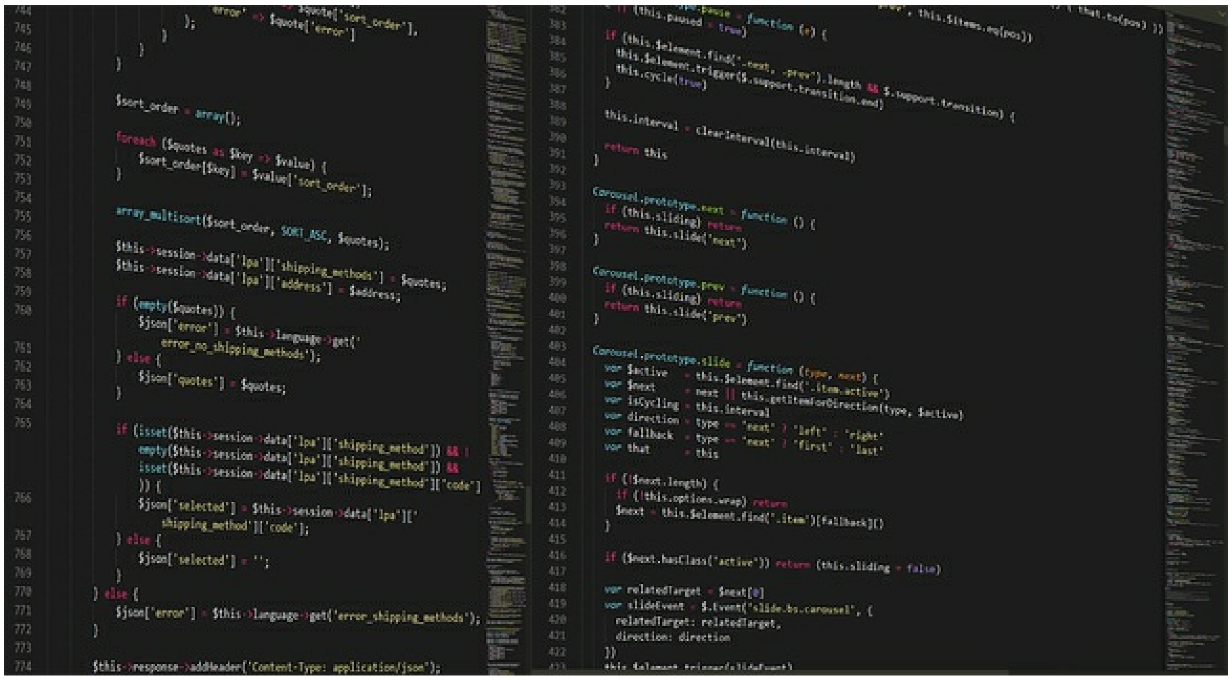
```
    FROM A a, A b
    WHERE a.B < b.B;
```

Next, let’s check out the CARTESIAN or CROSS JOIN. This is a kind of JOIN that will output the cross product of two sets of data points. These must be obtained from at least 2, but up to N tables. This makes it equivalent to an INNER type of JOIN where its condition for being joined is always true, or when there is absolutely no JOIN condition in the first place. The syntax for it is very simple:

```
SELECT table01.column01,table02.column02...column0N
FROM table01,table02,table03...table0N
```

And that’s it! The CARTESIAN JOIN really sounds a lot more complex than it actually is.

Chapter 10 Subquery



Definition

In SQL, subqueries are queries within queries. The subqueries usually use the WHERE clause. Also called nested query or internal query, subqueries can also restrict the data being retrieved.

Creating subqueries are more complex than creating simple queries. You have to use essential key words such as, SELECT, DELETE, UPDATE, INSERT and the operators such as, BETWEEN (used only WITHIN a subquery and not WITH a subquery), IN, =, < =, > =, >, <, < >, and similar symbols.

In composing a subquery, you have to remember these pointers.

- It must be enclosed with an open and close parenthesis.
- It can be used in several ways.
- It is recommended that a subquery can run on its own.
- It can ascribe column values for files.
- It can be found anywhere in the main query. You can identify it

because it is enclosed in parentheses.

- If it displays more than one row in response to an SQL command, this can only be accepted when there are multiple value operators. Example is the IN operator.
- In a subquery, the GROUP BY is used instead of the ORDER BY, which is used in the main statement or query.
- When creating subqueries, do not enclose it immediately in a set function.
- To create subqueries, it is easier to start with a FROM statement.
- Subqueries should also have names for easy identification.
- When using the SELECT key word, only one column should be included in your subquery. The exception is when the main query has selected many columns for comparison.
- Values that refer to a National Character Large Object (NCLOB), Binary Large Object (BLOB), Character Large Object (CLOB) and an Array, which is part of a collection data in variable sizes, should NOT be included in your SELECT list.

Working with the Queries

When you do set up the query that you would like to use, you will find that you are basically sending out an inquiry to the database that you already set up. You will find that there are a few methods to do this, but the SELECT command is going to be one of the best options to make this happen, and can instantly bring back the information that we need from there, based on our search.

For example, if you are working with a table that is going to hold onto all of the products that you offer for sale, then you would be able to use the command of SELECT in order to find the best selling products or ones that will meet another criterion that you have at that time. The request is going to be good on any of the information on the product that is stored in the database, and you will see that this is done pretty normally when we are talking about work in a relational database.

Working with the SELECT Command

Any time that you have a plan to go through and query your database, you will find that the command of SELECT is going to be the best option to make this happen. This command is important because it is going to be in charge of starting and then executing the queries that you would like to send out. In many cases, you will have to add something to the statement as just sending out SELECT is not going to help us to get some of the results that you want. You can choose the product that you would like to find along with the command, or even work with some of the features that show up as well.

Whenever you work with the SELECT command on one of your databases inside of the SQL language, you will find that there are four main keywords that we are able to focus on. These are going to be known as the four classes that we need to have present in order to make sure that we are able to complete the command that we want and see some good results. These four commands are going to include:

- SELECT—this command will be combined with the FROM command in order to obtain the necessary data in a format that is readable and organized. You will use this to help determine the data that is going to show up. The SELECT clause is going to introduce the columns that you would like to see out of the search results and then you can use the FROM in order to find the exact point that you need.
- FROM—the SELECT and the FROM commands often go together. It is mandatory because it takes your search from everything in the database, down to just the things that you would like. You will need to have at least one FROM clause for this to work. A good syntax that would use both the SELECT and the FROM properly includes:
- WHERE—this is what you will use when there are multiple conditions within the clause. For example, it is the element in the query that will display the selective data after the user puts in the information that they want to find. If you are using this feature, the right conditions to have along with it are the AND OR operators. The syntax that you should use for the WHERE command includes:

```
SELEC [ * | ALL | DISTINCT COLUMN1, COLUMN2 ]
FROM TABLE1 [ , TABLE2];
WHERE [ CONDITION1 | EXPRESSION 1 ]
[ AND CONDITION2 | EXPRESSION 2 ]
```

- ORDER BY—you are able to use this clause in order to arrange the output of your query. The server will be able to decide the order and the format that the different information comes up for the user after they do their basic query. The default for this query is going to be organizing the output going from A to Z, but you can make changes that you would like. The syntax that you can use for this will be the same as the one above, but add in the following line at the end:

```
ORDER BY COLUMN 1 | INTEGER [ ASC/DESC ]
```

You will quickly see that all of these are helpful and you can easily use them instead of the SELECT command if you would like. They can sometimes pull out the information that you need from the database you are working in a more efficient manner than you will see with just the SELECT command. But there are going to be many times when you will find that the SELECT command will be plenty to help you get things done when it is time to search your database as well.

A Look at Case Sensitivity

Unlike some of the other coding languages that are out there and that you may be tempted to use on your database searches, you may find that the case sensitivity in SQL is not going to be as important as it is in some of those other ones. You are able to use uppercase or lowercase words as you would like, and you can use either typing of the word and still get the part that you need out of the database. It is even possible for us to go through and enter in some clauses and statements in uppercase or lowercase, without having to worry too much about how these commands are going to work for our needs.

However, there are a few exceptions to this, which means there are going to be times when we need to worry about the case sensitivity that is going to show up in this language a bit more than we may want to. One of the main times for this is when we are looking at the data objects. For the most part, the data that you are storing should be done with uppercase letters. This is going to be helpful because it ensures that there is some consistency in the work that you are doing and can make it easier for us to get the results that we want.

For example, you could run into some issues down the road if one of the users is going through the database and typing in JOHN, but then the next person is typing in John, and then the third person is going through and typing in john to get the results. If you make sure that there is some consistency present, you will find that it is easier for all of the users to get the information that they want, and then you can make sure that you are able to provide the relevant information back when it is all done.

In this case, working with letters in uppercase is often one of the easiest ways to work with this because it is going to make it easier and the user is going to see that this is the norm in order options as well. If you choose to not go with uppercase in this, then you should try to find some other method that is going to keep the consistency that you are looking for during the whole thing. This allows the user a chance to figure out what you are doing, and will help them to find what they need with what is inside of their queries.

Chapter 11 Built-In Functions & Calculations

In regard to SQL, a built-in function can be defined as a portion of programming that accepts zero or any other input and returns an answer. There are different roles of built-in functions. These include the performance of calculations, obtaining of the system value, and application in textual data manipulation. This part aims at examining the various SQL built-in functions, categories of functions, pros and cons of functions and types of built-in functions.

Types of SQL Functions

The SQL functions are grouped into two groups: aggregate and scalar function. Working on Group by, the aggregate functions run on different records and deliver a summary. Scalar functions, on the other hand, run on different records independently.

These are as follows:

1. Single Row Functions-They provide a one-row return for any queried table. They are found in select lists, START WIH, WHERE CLAUSE and others. Examples of single-row functions include numeric_, datamining, Date time_,

conversion_ and XML_functions.

2. Aggregate Function-When you apply this kind of function, you see a single row returns based on different rows. The aggregate function exists in Select lists, ORDER BY and HAVING CLAUSE. They go hand in hand with Group by Clause and SELECT statements. Many of them do not take attention to null values. Those that are usually used include AVG, EANK, MIN, SUM and others.
3. Analytic Function-They are used to compute an aggregate value that are found on specific groups of rows. When you run this function, it delivers many rows for every group. The analytic functions are the last one to be run in a query. Examples of analytic functions include analytic-_clause and Order-by-Clause.
4. Model Functions-These are found in SELECT statements. Examples of model functions include CV, present and previous.
5. User-Defined Function-They are used in PL/SQL to offer functions that are not found in SQL. They mostly used in sections where expressions occur. For instance, you can find them in select list of Select statement.
6. SQL COUNT Function-It is used to provide the number of rows in a table.

Categories of Functions

Functions are classified according to the role they play on the SQL database. The following are some of the function categories available:

1. Aggregate Functions-They do a calculation on a specific set of values and deliver a single value. The aggregate values are used in the SELECT LIST and HAVING clause. The aggregate functions are referred to as being deterministic. This means that they return the same value when running on the same input value.
2. Analytic Function-They calculate an aggregate value according to a group of rows. They return many rows for different groups. The analytic functions can be used to perform different

computations like running totals, percentages and others.

3. Ranking Functions-They provide a ranking value for each portioned row. These kinds of functions are regarded as nondeterministic.
4. Rowset Functions- They're used to return an object that can be applied.
5. Scalar Functions-They work on a single value to return the same. There are various kinds of scalar values. These include configuration function, conversion function and others.

   a. Configuration Functions-They offer information about the present configuration.
   b. Conversion Functions-They support data changing.
   c. Cursor Functions-They provide information concerning cursors.
   d. Date and Time Data Type-They are concerned with operations as regards date and time.

6. Function Determinism-Functions that are found in SQL servers are either deterministic or nondeterministic. Deterministic functions provide the same answers when they are used. Nondeterministic functions, on the other hand, offer different results when they are applied.

SQL Calculations

There are various mathematical functions build-in the SQL server. The functions can be classified into 4 main groups, including Scientific and Trig Functions, rounding functions, signs and random numbers. Although there are numerous mathematical functions within each class, not all of them are used regularly. The various classes are highlighted and explained below:

1. Scientific and Trig Functions-Under this category, there are various subclasses found under it. These include P1, SQRT, and SQUARE. P1 function is used to compute the circumference and area in circles. How it works: SELECT 2 *10. SQRT connotes

square root. This function is used most of the time. The function recognizes any number that can be changed to float datatype. Example: SELECT SQET (36) Returns 6.SQUARE means that you multiply any number by itself. The concept of Pythagoras theorem is useful here. This means that Asquared+Bsquared=Csquared. This can be performed as SELECT SQRT (SQUARE (A) + SQUARE(B)) as C.

2. Rounding Functions- Under this class, there are various subcategories which include the CEILING and FLOOR. The ceiling function is helpful when dealing with a float or decimal number. Your interest is to find out the highest or lowest integer. Whereas the CEILING is the best highest integer, the floor represents the lowest integer. The ROUND function is applied when you want to round a number to the nearest specific decimal place. This is expressed as ROUND (value, number of decimal places).
3. Signs- There are occasions that require that you obtain an absolute figure of a number. For instance, the absolute value of -7 is 7. The absolute number doesn't contain any sign. To assist you with this task, it's essential to utilize the ABS function.
4. COS (X)-This function recognizes an angle expressed as radian as the parameter. After an operation, you get a cosine value.
5. SIN (X)-This function notices a radian angle. After computation, it gives back a sine value.
6. Sign-You can use a sign function when you want a negative, positive, or zero value.

The Importance of SQL Built-In Functions and Mathematical Applications

The build-in functions are sub-programs that help users to achieve different results when handling SQL database. These applications are used several times when you want to manipulate or process data. The SQL functions provide tools that are applied when creating, processing, and manipulating data. The benefits of SQL in-build and math functions are as follows:

1. Manipulation of Data-The in-built tools and math functions play a significant role in data manipulation. Manipulating massive data may be difficult if you do it manually. This is especially when the data is massive. Therefore, these functions play a significant role in ensuring that your data is manipulated fast as per your demands.
2. Assist in The Processing of Data-To benefit from data; you must process it. You may never have the ability to process big data manually. Therefore, the built-in SQL functions and math applications assist you in processing your database.
3. Simplifies Tasks-In case you're a programmer, you can attest to the fact that these functions and formulas make your work ease. You can work fast when you apply these in-build functions and formulas. Due to these tools, you'll accomplish many projects within a short time.
4. Increase Your Productivity-Using the in-built functions enhance your productivity as a programmer. This is because the functions enable you to work quickly on different projects. In case you were to handle data manually, you may take much time before you accomplish a task which ultimately injures your productivity. However, the built-in functions and calculations allow quick execution of tasks.
5. Time Saving-Because functions are written once and used several times, and they save much time. Besides time-saving, the functions offer support to modular programming.
6. They Enhance Performance- When you apply functions; you enhance the performance of your database. This is because the functions are prepared and inserted prior to usage.
7. Enhances Understanding of Complicated Logic-Handling of SQL database is a complex tax. Therefore, functions enable you to decompose data into smooth and manageable functions. In this way, you find it easy to maintain your database.
8. Cost Effective-Because the functions are in-build in the SQL database; you can use them many times without the need to

invest in new ones. In this connection, therefore, they reduce the cost of operating and maintaining your SQL database.

Downsides of In-Built Functions

The SQL in-built functions have various limitations, including:

1. Testability- When using the in-built functions, it's challenging to test their business philosophy. This is a big challenge, especially when you want the functions to support your business philosophy. You may never understand whether the functions are in line with your business vision and mission.
2. Versioning-It's challenging to establish the kind of version that is used in the SQL build-functions. You need to understand whether there're any new versions that can probably provide the best service.
3. Errors-In case there are errors within the in-build functions which you don't know, they may disrupt the program. This may prove costly and time-wasting.
4. Fear of Change-In case there is change; you may not understand how it will affect your SQL built-in functions. The world of technology keeps changes, and this change may affect the in-built functions.

The SQL in-built functions and calculations are essential as they enable a programmer to execute a task fast with minimal errors. The calculations in the in-built database makes it possible to process and manipulate data.

Chapter 12 Tips And Tricks Of SQL

SQL stands for structured query language. This language is a domain specific language that you are going to use if you are programming or trying to manage data inside of a RDBMS (relational database management system).

SQL was started with math, both tuple relational calculus and relational algebra. There is a lot of data definitions and manipulations along with control language that is going to be inside of SQL. SQL involves the use of things such as delete, update, insert, and query.

In essence, you are going to be able to update, delete, insert, and search for the things that you are going to be putting into the program. It is very common for SQL to be described as a declarative language, however, the program also allows for procedural elements.

This is one of the first languages that was able to use the relational model that was created by Edgar F Codd. Although it is not going to work with all of the rules that are set forth for this model, it is one of the most widely used languages for data bases.

In ’86, SQL became part of the ANSI. Then, in ’87 it became part of the ISO. However, there have been updates since then that have made it to where the language can include larger sets. Just keep in mind that the code for SQL is not going to be one hundred percent portable between data bases unless there are some adjustments to the code so that it fits the requirements for that data base.

Learning SQL can be one of the better decisions that you make about your career because you can push yourself forward with it that way that you can rely on using your own knowledge rather than having to go to someone else for their knowledge. In fact, people are going to be coming to you to learn what it is that you know about the program.

By learning SQL, you are going to be able to do more than you may have been able to before. Here are a few things that are going to give you a good reason as to why you should learn SQL.

Money

Learning SQL makes it to where you have the opportunity to earn some extra money. Developers that work with SQL earn around $92,000 a year! An administrator for an SQL data base is going to make about $97,000 a year. So, just learning SQL makes it to where you are able to earn around twice as much as what the average American household is going to make in a year.

Highly sought after

Employers are wanting people who know SQL! The more knowledge that you have about SQL the more sought after you are going to by employers. Knowing SQL is not only going to benefit you but your employer as well because they are not going to have to pay for you to learn the program. The interviewing process is going to be better than any other process that you have gone through and you may find that they are going to be willing to give you more money just for knowing SQL over the other person. With SQL knowledge, you are going to be opening yourself up for more careers than you might have been able to apply for before.

Get answers

SQL is going to give you the answers that you are looking for to any questions that you may have about business or data that is being stored inside of your data base. Therefore, you are going to be more self-sufficient and not as dependent on others when it comes to business. If you are able to answer questions on your own that you so that you are not stopping someone else from doing their job, then an employer is going to be able to save money by hiring you because you are going to be able to answer questions on hiring someone else to answer those questions. Knowing SQL is going to even help you if you are wanting to start your own business or push your business that you have already started to the next step that has just been out of your reach.

More stable than Excel

When you are using Excel for large amounts of data, you may notice that it is too much for the program and therefore the program tends to crash. A crash leads to lost data and extra time that you are going to have to go in and fix anything that may be wrong or entering data that was not saved. SQL is going to be much more reliable for you to use when you are trying to work with large amounts of data and it is going to save you some time working with it because it is not going to require too much for you to work with the processes that SQL offers.

Making reports

Searching in SQL is relatively easy and you can reuse that search when you have to double check to make sure that the data in the data base is accurate. Excel does not give you the proper processes that you need to get ahold of the data that you are wanting to get ahold of.

SQL coding only has to be written once and saved and then it is going to run each time that you need it to. This is yet another way that SQL makes it to where your life is easier because you are not having to take up as much time trying to get the data that you need.

Do not think that SQL is going to be simple, it is complicated and is going to take a lot of time to learn, but the more effort that you put into it, the more it is going to pay off for you. You are going to not only be saving money by learning SQL, but you are going to be increasing what you will be able to make with SQL.

Four Tips That Make Using SQL Easier!

1. Changing the language on the user interface: close out the program if you have it open and then go to the installation

folder. You will right click on the short cut that is on your desk top and open the file location. From there you will open the SQL developer folder and then the first folder that is listed will need to be opened next. The next thing that you are going to click on is the SQL developer.conf. You are going to be adding in a new setting inside of the text that is already there to change the language to what it is that you are wanting to see. You can put this new setting anywhere. Putting a comment in the code is going to be a good idea so that you know what you have done if you have to get back into it at a later date. You will AddVMOption before adding in the user language and you can set it to any language that you are wanting. Now reopen your SQL developer and it will be in the language that you want it in.

2. Construct data base connections: right click on the connection on the left of the screen and click on new connection. You will need to title the connection whatever it is that you want. You will need to enter the user title and password for it. You should change the color if you are going to be working with multiple connections at once. In the role you are going to change the role if you are using a system connection title. You can leave the home host alone if you are using your home computer. However, if you are using a different location, you will need to input the IP address for where the system is going to be running. Leave your part alone and xe should be left alone as well unless you are not working with an express edition of SQL. You can test the connection and if it is successful, you can close the connection down and you have created your connection. If everything is correct it is going to open with no errors and you are going to be able to put in SQL code.
3. Disabling features: there are a lot of features that SQL offers and if you do not use them, then you should disable them so that they are not slowing down the developer. You will go to the tools menu and go down to the features option. Each feature has different folders, it is up to you to decide which features you want to keep running and which ones you want to disable. You

can expand each folder down so that you are able to see what each folder contains. All you are going to do is uncheck the feature and it will turn that feature off and cause the system to start to run faster. Be sure that you are going to apply the changes so that they are not turning themselves back on without you turning them on yourself.

4. Executing commands and scripts: use the tool bar that is at the top of the developer and press the play button. Make sure that you have added in your semi colon. You can also use ctrl and enter so that you are not having to pull your hand off the keyboard. To run a script, you are going to you can use the toolbar again just select run scripts so you run both commands. Or, press the F5 key if that is easier for you. Should your file be external use the at sign and the path file to import it and run it.

Conclusion

Since its introduction in the computing world, SQL has played a significant role in revolutionizing data storage in a systematic manner as well as direct retrieval. As the digital world continues to grow steadily, the amount of data stored quickly escalates, making organizations and personal data piling up. Therefore, SQL acts as a platform where these data are stored while offering direct access without the emergence of limitations. As such, SQL is used in different sectors, including telecommunication, finance, trade, manufacturing, institutional, and transport. Its presence primarily deals with data but also accompanies other significant benefits on its application.

Data Integration

Across the sectors mentioned above, one of its main applications of SQL is the creation of data integration scripts commonly done by administrators and developers. SQL databases comprise of several tables which may contain different data. When these data are integrated, it creates a new experience essential for the provision of meaningful information, therefore, increasing productivity. Data integration scripts are crucial in any given organization, including the government, as it offers trusted data which can be utilized to promote the achievement of the set goals.

Analytical Queries

Data analysts regularly utilize Structured Query Language to smoothen their operations more so when establishing and executing queries. That is SQL comprises multiple tables that consist of different datasets. When these data are combined, it brings out more comprehensive information critical for any individual or organization. The same is also applicable for data analysts as they use a similar aspect. As they use an analytical query structure, queries, and tables from SQL are fed into the structure to deliver crucial results from varying sources. In this case, data analysts can readily acquire different queries and customize them to have a more comprehensive data to depend on as solutions.

Data Retrieval

This is another important application of SQL to retrieve data from different subsets within a database with big data. This is essential in financial sectors and analytics as to the use of numerical values typically consists of mixed statistical data. The most commonly used SQL elements are create, select, delete, and update, among others. The technique is made possible when the user quickly can search the database and acquire the needed data as SQL sieves the information to bring out the desired data. In some cases, the language may deliver similar or related data when the required data is missing. This is crucial as one can compare the results as well as make changes where the need arises.

MySQL system has a slot that allows you to announce more than one set of variables that has a common type of data. Again, most of the technicians have had issues with how this command relays information to related databases. In the various methods of storing variances and variables, this one has proven to be more secure than others. Consequently, it has been known to be the most popular of them all.

Variables can be applied in mathematical expressions, for example, adding values altogether or combining and holding texts. This can be used as a section of the general information. For your information, variables are also applied in storing information so as one can participate in a kind of calculations. Additionally, variables can be part of the parameters and are used in procedural assessments. This is two in one method that not only lets you declare a variable but also setting it up with values that have a similar data type. Going back to the examples we gave earlier, we can affirm that varchar is a kind of data that lets you sustain more than one kind of character in just a single string.

Up to this point, you should be in a position to understand the kind of SQL Exercises and Programs as well as the various types in existence. This will not only let you be in an excellent place to tackle errors in case they occur and prevent them from happening as well. When Mark Suaman, a renowned data scientist and a graduate of Havard University, first used varchar, he recommended it for being efficient and accurate. He rated it among the best types of data set in the market today. It does not have an allocation for potential occurrences of errors. It is hard to interfere with such a highly secure kind of data type.

There is plenty to learn when it comes to SQL, but with the use of practice and good knowledge, you can be as successful as you decide to be in any database. Just how the English language has many rules to be followed, the same applies with SQL. By taking the time to thoroughly learn the language, many things are achievable with the use of a database. Refer back to any of the information any time you are stumped on something you are working on.

Although it can be a complex challenge, patience and practice will help you successfully learn SQL. By remembering the basic commands and rules to SQL, you will avoid any issues that can come across most individuals that practice the use of it. It is a lot of information to take in, but instead, take it as it comes. Go to the practical tools that you may need for whatever you are trying to achieve through the database. When presented with an obstacle or complex assignment, refer to the tools that will clear up what you need. Take time to fully analyze what is before you while also trying to focus on one thing at a time.

Keep an open and simple mind when moving forward and you will keep any issues from becoming more complicated than what they need to be. As mention, SQL can be a simple thing to learn. You just need to take the time to understand what everything fully means in depth. If something doesn't turn out as expected, retrace your tracks to find where you might have inappropriately added a formula and some of the information. By building and maintaining successful problem-solving skills, you will not limit your success.

Thank you again for purchasing this book! I hope this book was able to help you thoroughly understand how SQL works. The next step is to put the tools and knowledge to use in your SQL database. Finally, if you enjoyed this book, please take the time to share your thoughts and post a review on Amazon. It'd be greatly appreciated!

Thank you and good luck!