SQL

Crash Course - The Ultimate Beginner's Course to Learning SQL Programming in a weekend

Chris Sheridan

Learn SQL Database Programming in a Weekend

Chris Sheridan

opening

Learn SQL Database Programming in a weekend.

Create SQL DatabASE and Tables From Scratch
and learn to use these databases to store information of all sorts and learn to use this wonderful language on your own.

Structured Query Language, is a special-purpose programming language designed for managing data held in a relational database management system or for stream processing in a relational data stream management system.

Originally Publishing, learenythinginaday.com; Sheridan, Chris(2014-10-14). SQL: Learn SQL DataBase Programming in a weekend Or Less!:

The book will provide quick and easy introduction to SQL data base programming and allows someone with a basic idea of programming to create their own databases and tables by using the Structured Query Language to their advantage. Published by learenythinginaday.com.

Notice of Liability:

The information in this book is distribution on as “As Is” Basis, without warranty. While every precaution has been taken in the preparation of this book, neither the author nor publisher shall have any liability to any person or entity with respect to any lose or

damage caused or alleged to be caused directly or indirectly by the instructions contained in this book or by the computer software and hardware products described in it.

Introduction

SQL stands for structured query language. It is used to create data stores and to interact with this Data

It is the most commonly used language in computing today. Whether you are an application developer, database administrator,software support guy or a web front end engineer you will more than likely use SQL to interact with your business underlying database at some time.

So a good working knowledge of SQL is a necessary prerequisite for many jobs.
This book was born out of necessity as i have seen far to many people struggle with the power off this simple yet very powerful language.

I have worked in computing for over 20 years and i'm still shocked at how many people don't understand the basics of SQL.

Many colleagues over the years have come to me seeking advice and asking what book would i recommend to learn the basics of SQL.

Yes indeed there are hundreds of good books about SQL but none that i can find to teach the basics of this wonderful language.

All these books start of with good intentions but veer off the path into SQL design , data storage or some database waffle making it more complex and time consuming than necessary for newbies like you.

I have written this book to teach the basics of the SQL language to anybody and in a very simple clear way without getting too technical.

If at anytime you feel lost or might need a helping hand, please feel free to contact me via my email address sql@learnanythinginaweekend.com. and i will get back to you usually within 48 - 72 hours.

I thank you for downloading this book and i hope you learn from this book and enjoy your journey of discovery.

Chapter 1 - So What Is Sql

You may of heard programmers or IT people talking about “sequel” Or SQL, so what exactly is it. Structured Query Language (**SQL**) is the language of databases. All modern relational databases, including MySql, Access, FileMaker Pro,Oracle and Microsoft SQL Server use SQL as their basic building block. In fact, it’s often the only way that you can truly interact with the database itself.

Programmers use SQL to interact with the underlying data of any computer System.
To me Data is the heart of any IT system or program and SQL is the blood that flows through the heart , giving life to every being,

All of the fancy screens that you have seen be it an application form on a web site, a Data Entry screen at work, your favourite App or a Timetable screen in the Airport are nothing more than the end use of a programmers hard work to interact with the underlying Database.

By the way a **Database** is a data store on some computer anywhere it doesn’t matter..
A Database is designed by IT people and is made up of **Data** stored in Tables.
A **Table** will usually store related data like Customer, Product and Users.

In the Customer table you would store customer information in rows one **row** of customer data would be called a customer **record**. Each record would contain unique customer data in **columns** called **fields**, For example customer first name,customer last name,customer date of birth, customer phone number or customer email address would all be examples of fields stored on a customer record.

SQL allows the programmer to access and manipulate this Data through a list of english like commands and statements.

SQL Clauses are used to refine your data digging further. In SQL there are many clauses and we cover quite a few in this book. Here are examples of SQL clauses;

- Where
- Order by
- Group by

SQL Operators

An operator allows you to extract your data in data result set

- Logical Operators
- Comparison Operators
- IN BETWEEN,LIKE, ISNULL

are all examples of SQL operators.

Dont be to worried about this new terminology, all will be revealed in the coming chapters.

Summary

SQL stands for Structured Query language. A Database is a datastore of tables that holds data.

Questions

1) What does SQL stand for ?
2) What does a database store ?

Chapter 2 - Basic Database Concepts

To understand SQL you need to understand what a Database is. This chapter aims to remove some of the mystique surrounding a Database and hopefully make you more enlightened.

SQL

Structured Query Language is the Language used to control Databases allowing you to create the Data structures and manipulate the underlying Data.

RDBMS

RDBMS is a term meaning Relational Database Management System
RDMS is the underlying software used to store Databases and allows for use of the SQL language and database maintenance tasks such as backups and safe access to the Data. There are many providers of RDBMS software such as Oracle,Mysql, Sql/Server and DB2.

Database

As explained before a Database is just a store of Data in logical units or entities called tables.
Each table contains rows or records of logical related data.

Tables

A table in SQL terms is stored in a Database and contains rows of data called records.

Records

Records are horizontal rows in any table. Sometimes a record will be referred to as a row and they two terms are often used intermingly.

Fields or Columns

A Record contains one or many Fields. A field contains data. This data is a store and maintain data, for example First name, email address. Columns are defined by their Datatype, see chapter 3 about Datatypes

RDBMS Implementations

ORACLE
Oracle is an object-relational database management system produced and marketed by Oracle Corporation. It allow for vast storage and is normally used by large organisations

Mysql
Mysql is provided as open source and thus is free, It is the worlds most commonly used Database for Websites. It is efficient and fast.

SQL/SERVER
SQL/SERVER is provided by Microsoft and like Oracle it allows for huge storage and is favoured by large organisations. Its allows for many concurrent users and has a more advanced locking mechanism than MYSQL.

DB2
DB2 is provided by IBM. It to is aimed at Big Data and multi users it is not used as widely as Oracle and SQL/Server

Summary
This chapter covers the basics of SQL to prepare you for coming chapters. You will be using MySQL to create, modify, change and remove tables.

Questions
1. What are records ?
2. What company provides DB2?
3. What are tables?

Chapter 3 - Datatypes

Each column in a database is required to have a name and a data type

A data type associates a fixed set of properties with the values that can be used in a column of a table or in an argument. These properties cause a Database to treat values of one data type differently from values of another data type.

Datatypes are used to create tables. Each column has a Datatype. In this chapter we will explore the most commonly used data types and briefly explain how the are different.

The following table lists the general data types in SQL:

Data type	Description
CHARACTER(n)	Character string. Fixed-length n
VARCHAR(n) or CHARACTER VARYING(n)	Character string. Variable length. Maximum length n
BINARY(n)	Binary string. Fixed-length n
BOOLEAN	Stores TRUE or FALSE values
VARBINARY(n) or BINARY VARYING(n)	Binary string. Variable length. Maximum length n
INTEGER(p)	Integer numerical (no decimal). Precision p
SMALLINT	Integer numerical (no decimal). Precision 5
INTEGER	Integer numerical (no decimal). Precision 10
BIGINT	Integer numerical (no decimal). Precision 19

DECIMAL(p,s)	Exact numerical, precision p, scale s. Example: decimal(5,2) is a number that has 3 digits before the decimal and 2 digits after the decimal
NUMERIC(p,s)	Exact numerical, precision p, scale s. (Same as DECIMAL)
FLOAT(p)	Approximate numerical, A floating number in base 10 exponential notation. The size argument for this type consists of a single number specifying the minimum precision
REAL	Approximate numerical
FLOAT	Approximate numerical
DOUBLE PRECISION	Approximate numerical
DATE	Stores year, month, and day values
TIME	Stores hour, minute, and second values
TIMESTAMP	Stores year, month, day, hour, minute, and second values
INTERVAL	Composed of a number of integer fields, representing a period of time, depending on the type of interval
ARRAY	A set-length and ordered collection of elements
MULTISET	A variable-length and unordered collection of elements
XML	Stores XML data

Please note that different RDBMS may have different names for datatypes, but the underlying principles remain the same.

The following table shows the different names of the most common datatypes in different RDBMS

Data type	Access	SQLServer	Oracle	MySQL	PostgreSQL
boolean	Yes/No	Bit	Byte	N/A	Boolean
integer	Number (integer)	Int	Number	Int Integer	Int Integer
float	Number (single)	Float Real	Number	Float	Numeric
currency	Currency	Money	N/A	N/A	Money
string (fixed)	N/A	Char	Char	Char	Char
string (variable)	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar
binary object	OLE Object Memo	Binary (fixed up to 8K) Varbinary (<8K) Image (<2GB)	Long Raw	Blob Text	Binary Varbinary

Please refer to appendix A for a table that describes the various datatypes for Mysql, Sql Server and Access.

Questions

1. What is an Integer Datatype?
2. What is a FLOAT Datatype?
3. What values does a boolean datatype hold ?

Chapter 4 Sql syntax

This chapter details the most common commands used in SQL. Some commands like Insert, Update and Delete are so important and most frequently used that they have a chapter dedicated to them later in this book. Unlike alot of programming languages SQL is not case sensitive.

In this chapter you will learn what syntax statement to use to perform specific actions.

Action	SQL Statement	Syntax
Alias for a column	AS (alias)	SELECT column_name AS column_alias FROM table_name or SELECT column_name FROM table_name AS table_alias
Alter a table structure	ALTER TABLE	ALTER TABLE table_name ADD column_name datatype or ALTER TABLE table_name DROP COLUMN column_name
Create a Database	CREATE DATABASE	CREATE DATABASE database_name
Create a table	CREATE TABLE	CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name2 data_type, ...)
Create a table index	CREATE INDEX	CREATE INDEX index_name ON table_name (column_name) or CREATE UNIQUE INDEX index_name ON table_name (column_name)

Create a table view	CREATE VIEW	CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition
Drop a database	DROP DATABASE	DROP DATABASE database_name
Delete a row	DELETE	DELETE FROM table_name WHERE some_column=some_value or DELETE FROM table_name (**Note:** Deletes the entire table!!) DELETE * FROM table_name (**Note:** Deletes the entire table!!)
Drop a table	DROP TABLE	DROP TABLE table_name
Drop a table index	DROP INDEX	DROP INDEX table_name.index_name (SQL Server) DROP INDEX index_name ON table_name (MS Access) DROP INDEX index_name (DB2/Oracle) ALTER TABLE table_name DROP INDEX index_name (MySQL)
Filter records	AND / OR	SELECT column_name(s) FROM table_name WHERE condition AND|OR condition

Filter records	EXISTS	IF EXISTS (SELECT * FROM table_name WHERE id = ?)
		BEGIN
		--do what needs to be done if exists
		END
		ELSE
		BEGIN
		--do what needs to be done if not

		END
Filter records	HAVING	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name HAVING aggregate_function(column_name) operator value
Filter records	IN	SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,..)
Filter records	INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name
Filter records	LEFT JOIN	SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
Filter records	RIGHT JOIN	SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
Filter records	FULL JOIN	SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name
Filter records	LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern

Filter records	UNION ALL	SELECT column_name(s) FROM table_name1 UNION ALL SELECT column_name(s) FROM table_name2
Group records	GROUP BY	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name
Insert records	INSERT INTO	INSERT INTO table_name VALUES (value1, value2, value3,....) *or* INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,....)
Order records	ORDER BY	SELECT column_name(s) FROM table_name ORDER BY column_name [ASC|DESC]
Select records	SELECT	SELECT column_name(s) FROM table_name
Select records	SELECT *	SELECT * FROM table_name
Select records	SELECT DISTINCT	SELECT DISTINCT column_name(s) FROM table_name
Select records	SELECT INTO	SELECT * INTO new_table_name [IN externaldatabase] FROM old_table_name *or* SELECT column_name(s) INTO new_table_name [IN externaldatabase] FROM old_table_name
Select records	SELECT TOP	SELECT TOP number|percent column_name(s) FROM table_name
Select	WHERE	SELECT column_name(s)

records		FROM table_name WHERE column_name operator value
Select records	UNION	SELECT column_name(s) FROM table_name1 UNION SELECT column_name(s) FROM table_name2
Truncate table	TRUNCATE TABLE	TRUNCATE TABLE table_name
Update records	UPDATE	UPDATE table_name SET column1=value, column2=value,... WHERE some_column=some_value

Summary

The SQL language contains many commands allowing you to create data stores and manipulate the underlying data.

Questions

1. What statement would you use to create a row on a table?
2. What statement would you use to remove a row from a table ?
3. What statement would you use to update data on a table ?

Chapter 5 Create a Database and Tables

Database management systems (DBMSs) are computer software applications that interact with the user, other applications, and the database itself to capture and analyze data.
A general-purpose DBMS is designed to allow the definition, creation, querying, update, and administration of databases.

A Database Often abbreviated DB is basically a collection of information organized in such a way that a computer system can quickly select desired pieces of data.

You can think of a database as an electronic filing system.
Traditional databases are organized into tables, records and fields.

A table contains a list of records or rows, each of which consists of columns or fields.

A field is a single piece of information; a record is one complete set of fields; and a table is a collection of records. For example, a telephone book is
analogous to a table.

For the purpose of this book we will be using the MySQL DBMS. MySQL is the world's most popular open source database, enabling the cost-effective delivery of reliable, high-performance and scalable Web-based and embedded database applications.

MySQL is available for free here at
http://dev.mysql.com/downloads/windows/

How to Create a MySQL Database

Like any RDBMS MySQL organizes its information into databases; each one can hold tables with specific related data. As this book is a SQL programming guide i will not cover the installation of MySQL , for a very good visual guide on this please follow this link http://www.wikihow.com/Install-the-MySQL-Database-Server-on-Your-Windows-PC

MySQL can be an intimidating program. All of the commands have to be entered through a command prompt as there is no visual interface. Because of this, having basic knowledge of how to create and manipulate a database can save you a lot of time and hair pulling.

You can quickly check what databases are available by typing:

```
SHOW DATABASES;
```

Your screen should look something like this:

```
mysql> SHOW DATABASES;
+--------------------+
```

```
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test2014|
+--------------------+
4 rows in set (0.01 sec)
```

Creating a database is very easy as you only have to use one command :

```
CREATE DATABASE database_name;
```

In this case, for example, we will call our database "college"

```
CREATE DATABASE college;
```

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
| test2014|

|college|
+--------------------+
```

5 rows in set (0.00 sec)

How to access a MySQL Database

Once we have a new database, we can begin to fill it with information.The first step is to create a new table within the larger database.

Let’s open up the database we want to use:

```
USE college;
```

In the same way that you could check the available databases, you can also see an overview of the tables that the database contains.

```
SHOW tables;
```

Since this is a new database, MySQL has nothing to show, and you will get a message that says, “Empty set”

How to Create a MySQL Table

Lets Imagine that we are creating a database for a college , we will use the database created above “college”. Now this database needs to store alot of information concerning the college.
We are going to break all related information into tables

Let’s create a new MySQL table:

```
CREATE TABLE students(id INT NOT NULL PRIMARY KEY
AUTO_INCREMENT,

name VARCHAR(30),

emailaddress VARCHAR(255),

confirmed CHAR(1),

signup_date DATE);
```

This command accomplishes a number of things:

1. It has created a table called studentswithin the college database

2. We have set up 5 columns in the table—id, name, emailaddress , confirmed, and signup_date.
3. The “id” column has a command (INT NOT NULL PRIMARY KEY AUTO_INCREMENT) that automatically numbers each row.
4. The “name” column has been limited by the VARCHAR command to be under 20 characters long.
5. The “emailaddress ” column designates the emailaddress for each person The VARCHAR limits text to be under 255characters.
6. The “confirmed” column records whether the person has RSVP’d with one letter, Y or N.
7. The “signup_date” column will show when they signed up for the coutse. MySQL requires that dates be written as yyyy-mm-dd

Let’s take a look at how the table appears within the database using the "SHOW TABLES;" command:

```
mysql> SHOW TABLES;
+-----------------+
| Tables_in_events |
+-----------------+
| students        |
+-----------------+
1 row in set (0.01 sec)
```

We can remind ourselves about the table’s organization with this command:

```
DESCRIBE students;
```

Keep in mind throughout that, although the MySQL command line does not pay attention to cases, the table and database names are case sensitive: students is not the same as STUDENTS or Students.

Summary

Questions

1. What command shows all the databases in your RDBMS
2. What command shows all the tables in a database
3. What command would you use to show the structure of a table.

Chapter 6 The Select Statement

Once you have inserted data onto your tables, wouldn't it be nice to be able to retrieve it . You could then possibly use this data for display or validation purposes.

The SELECT statement is used to select data from database tables.
The result is stored in a result table, called the result-set.

SQL SELECT Syntax

SELECT *column_name,column_name*
FROM *table_name*
where condition
Order by condition
Group by condition;

and
SELECT * FROM *table_name*;

If you want to select one column from a table use the syntax

Select column1 from table_name;

But if you want to select more than one column from a table you would use
Select column1,column2 from table_name;

If you wanted to select all columns from the table you would use the star or asterisk symbol like so;
Select * from table_name;

These examples are all valid sql statements but they will return all rows on the table as we didnt specify what we are are looking for.To limit the rows returned you use the where clause of the Select statement.
If you want to return a particular row/rows from a table, you would use the where clause

Using the where clause

Select result from marks where
Student_id = 5

This statement will return the result column from they marks table where the student id is equal to 5.
If you want to return a value from a table based on more than one column you use **'and'** in the Select statement.
Select emailaddress from students where firstname = "kevin" and lastname = "kelly"

If you want to return a row from a table where a column may have more than one value you would use '**OR'** in your condition

Select emailaddress from students where student_id = 5 or student_id = 10;

This will return all rows from the student table where student_id = 5 or student_id = 10;

Using the group by clause

The SQL GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

SYNTAX

The syntax for the SQL GROUP BY clause is:

```
SELECT expression1, expression2, ... expression_n,
    aggregate_function (expression)
FROM tables
WHERE conditions
GROUP BY expression1, expression2, ... expression_n;
```

Using the order by clause

DESCRIPTION

The SQL ORDER BY clause is used to sort the records in the result set for a SELECT statement.

SYNTAX

The syntax for the SQL ORDER BY clause is:

```
SELECT expressions
FROM tables
WHERE conditions
ORDER BY expression [ ASC | DESC ];
```

Summary

The select statement is used to select data from a table in the database.

Questions

1.What does the Sql Select statement let you do ?

2. What clause allows you to sort data in a certain way ?

Chapter 7 The Insert Statement

Now that you have created a table lets populate it with some data.

To populate a table you use the insert statement.

The Insert statement enables you to insert rows into a table.

The basic syntax of an insert statement is

Insert into table_name col1, col2 values val1,val2;

This statement would insert a new row on the table table_name with a data values of val1 and val2 into the columns col1,col2 .

To insert a row to our students table you would use this syntax

Insert into students col1,col2 values val1,val2;

This insert statement insert a row into the table students with values val1, val2 in the corresponding columns col1 and col2.

Some insert statements can be used to insert data into particular tables without specifying the column names to do this you would use this syntax

Insert into table_name values val 1, val 2;

This particular insert statement whilst it's faster to code and for the computer to understand can be a bit sloppy if you change the order of columns in your table.

Another Insert statement that can be used use to create a row on a table could be "The select statement" and the syntax of this would be select into 1 values

By using a SELECT subquery to specify the data values for one or more rows, such as:

```
INSERT INTO table_name (PriKey, Description)
    SELECT ForeignKey, Description
    FROM anothertable;
```

This statement insert's all rows from the "anothertable" table into the "table_name" table.

Questions

1. What SQL statement allows you to add rows

Chapter 8 The Update Statement

Now that you have data on your tables, there will come a time when you need to update that data. In SQL we use the update statement to update a row or numerous rows of data on a table or tables.
The basic syntax for the SQL update statement when updating one table is:

Update table_name
set column1 = expression1,
column2 = expression2
Where conditions;

This statement will update a table called table_name setting a column column1 equal to expression1 and the column column2 equal to expression 2 where a certain condition is equal to conditions.

An example of the update statement in use would be;

Update marks
set result = 90
Where studentid = 5
And exam = 'maths';

This statement is updating the result column value to 90 of the marks table where the column student id equals 5 and the exam column is equal to ‘maths’

If you wanted to update more than one column on a table all you have to is separate the updateable columns by a comma like so

Update marks
set result = 10,verified = True
Where studentid = 5
And exam = 'maths';

This update statement will update all rows on the marks table

setting the resut column equal to 10 and the verified column equal to True.

who have a *column studentid equal to 5*

And exam column equal to 'maths'

Please note: If you omit the Where clause then all rows on the table will be updated.

Chapter 9 The Delete Statement

The delete statement is used to remove redundant rows/records from your table.

These rows where added to tables by use of the insert statement in chapter 7.
It is important to use this statement carefully as it is a very powerful command .
Once you remove a row from a table that it it's gone.
So be very careful when using this command.
The basic syntax of the delete statement is

Delete from mytable
where col1 = val1;

This will delete all rows from mytable who have a column col1 value equal to the value of val1.

It is very important to use the where clause otherwise you will remove all rows from the table.

For instance where you to use the command

delete from marks;

you would remove all rows from the table marks.

This is how you would delete a particular student from the student table who's id you know, you would use the command like so;

delete from student where Id =10;

If you wanted to delete a student called kevin kelly you would use this command

delete from student where firstname= 'kevin' and lastname = 'kelly';

Summary

This chapter explained how to remove rows from a table by using the Delete statement.

Questions

1. Why is it important to use the where clause of the Delete statement?

Answers

Chapter 1.

1. Structured Query Language
2. Data store of tables

Chapter 2.

1. Records are horizontal rows in any table
2. IBM
3. A table in SQL terms is stored in a Database and contains rows of data called records.

Chapter 3.

1. Integer numerical (no decimal). Precision p
2. Approximate numerical
3. True or false

Chapter 4.

1. The Insert Statement
2. The Delete statement
3. The Update statement.

Chapter 5.

1. Show Databases;
2. Show tables;
3. Describe

Chapter 6.

1. The Select statement let you retrieve data from a table
2. The Order By clause

Chapter 7.

1. The Insert statement

Chapter 9.

1. If you don’t use the where clause of the delete statement, you will remove all rows from the specified table.

Appendix A Data type for various dbs

Data types and ranges for Microsoft Access, MySQL and SQL Server.

Microsoft Access Data Types

Data type	Description
Text	Use for text or combinations of text and numbers. 255 characters maximum
Memo	Memo is used for larger amounts of text. Stores up to 65,536 characters.**Note:** You cannot sort a memo field. However, they are searchable
Byte	Allows whole numbers from 0 to 255
Integer	Allows whole numbers between -32,768 and 32,767
Long	Allows whole numbers between -2,147,483,648 and 2,147,483,647
Single	Single precision floating-point. Will handle most decimals
Double	Double precision floating-point. Will handle most decimals
Currency	Use for currency. Holds up to 15 digits of whole dollars, plus 4 decimal places.**Tip:** You can choose which country's currency to use
AutoNumber	AutoNumber fields automatically give each record its own number, usually starting at 1
Date/Time	Use for dates and times
Yes/No	A logical field can be displayed as Yes/No, True/False, or On/Off. In code, use the constants True and False (equivalent to -1 and 0). **Note:** Null values are not allowed in Yes/No fields
Ole Object	Can store pictures, audio, video, or other BLOBs (Binary

	Large OBjects)
Hyperlink	Contain links to other files, including web pages
Lookup Wizard	Let you type a list of options, which can then be chosen from a drop-down list

MySQL Data Types

In MySQL there are three main types : text, number, and Date/Time types.

Text types:

Data type	Description
CHAR(size)	Holds a fixed length string (can contain letters, numbers, and special characters). The fixed size is specified in parenthesis. Can store up to 255 characters
VARCHAR(size)	Holds a variable length string (can contain letters, numbers, and special characters). The maximum size is specified in parenthesis. Can store up to 255 characters. **Note:** If you put a greater value than 255 it will be converted to a TEXT type
TINYTEXT	Holds a string with a maximum length of 255 characters
TEXT	Holds a string with a maximum length of 65,535 characters
BLOB	For BLOBs (Binary Large OBjects). Holds up to 65,535 bytes of data
MEDIUMTEXT	Holds a string with a maximum length of 16,777,215 characters
MEDIUMBLOB	For BLOBs (Binary Large OBjects). Holds up to 16,777,215 bytes of data
LONGTEXT	Holds a string with a maximum length of

	4,294,967,295 characters
LONGBLOB	For BLOBs (Binary Large OBjects). Holds up to 4,294,967,295 bytes of data
ENUM(x,y,z,etc.)	Let you enter a list of possible values. You can list up to 65535 values in an ENUM list. If a value is inserted that is not in the list, a blank value will be inserted. **Note:** The values are sorted in the order you enter them. You enter the possible values in this format: ENUM('X','Y','Z')
SET	Similar to ENUM except that SET may contain up to 64 list items and can store more than one choice

Number types:

Data type	Description
TINYINT(size)	-128 to 127 normal. 0 to 255 UNSIGNED*. The maximum number of digits may be specified in parenthesis
SMALLINT(size)	-32768 to 32767 normal. 0 to 65535 UNSIGNED*. The maximum number of digits may be specified in parenthesis
MEDIUMINT(size)	-8388608 to 8388607 normal. 0 to 16777215 UNSIGNED*. The maximum number of digits may be specified in parenthesis
INT(size)	-2147483648 to 2147483647 normal. 0 to 4294967295 UNSIGNED*. The maximum number of digits may be specified in parenthesis
BIGINT(size)	-9223372036854775808 to 9223372036854775807 normal. 0 to 18446744073709551615 UNSIGNED*. The maximum number of digits may be specified in parenthesis

FLOAT(size,d)	A small number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DOUBLE(size,d)	A large number with a floating decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter
DECIMAL(size,d)	A DOUBLE stored as a string , allowing for a fixed decimal point. The maximum number of digits may be specified in the size parameter. The maximum number of digits to the right of the decimal point is specified in the d parameter

*The integer types have an extra option called UNSIGNED. Normally, the integer goes from an negative to positive value. Adding the UNSIGNED attribute will move that range up so it starts at zero instead of a negative number.

Date types:

Data type	Description
DATE()	A date. Format: YYYY-MM-DD **Note:** The supported range is from '1000-01-01' to '9999-12-31'
DATETIME()	*A date and time combination. Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
TIMESTAMP()	*A timestamp. TIMESTAMP values are stored as the number of seconds since the Unix epoch ('1970-01-01

	00:00:00' UTC). Format: YYYY-MM-DD HH:MI:SS **Note:** The supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC
TIME()	A time. Format: HH:MI:SS **Note:** The supported range is from '-838:59:59' to '838:59:59'
YEAR()	A year in two-digit or four-digit format. **Note:** Values allowed in four-digit format: 1901 to 2155. Values allowed in two-digit format: 70 to 69, representing years from 1970 to 2069

*Even if DATETIME and TIMESTAMP return the same format, they work very differently. In an INSERT or UPDATE query, the TIMESTAMP automatically set itself to the current date and time. TIMESTAMP also accepts various formats, like YYYYMMDDHHMISS, YYMMDDHHMISS, YYYYMMDD, or YYMMDD.

SQL Server Data Types

String types:

Data type	Description
char(n)	Fixed width character string. Maximum 8,000 characters
varchar(n)	Variable width character string. Maximum 8,000 characters
varchar(max)	Variable width character string. Maximum 1,073,741,824 characters
text	Variable width character string. Maximum 2GB of text data
nchar	Fixed width Unicode string. Maximum 4,000 characters
nvarchar	Variable width Unicode string. Maximum 4,000 characters

nvarchar(max)	Variable width Unicode string. Maximum 536,870,912 characters
ntext	Variable width Unicode string. Maximum 2GB of text data
bit	Allows 0, 1, or NULL
binary(n)	Fixed width binary string. Maximum 8,000 bytes
varbinary	Variable width binary string. Maximum 8,000 bytes
varbinary(max)	Variable width binary string. Maximum 2GB
image	Variable width binary string. Maximum 2GB

Number types:

Data type	Description
tinyint	Allows whole numbers from 0 to 255
smallint	Allows whole numbers between -32,768 and 32,767
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807
decimal(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1. The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0
numeric(p,s)	Fixed precision and scale numbers. Allows numbers from -10^38 +1 to 10^38 –1.

	The p parameter indicates the maximum total number of digits that can be stored (both to the left and to the right of the decimal point). p must be a value from 1 to 38. Default is 18. The s parameter indicates the maximum number of digits stored to the right of the decimal point. s must be a value from 0 to p. Default value is 0
smallmoney	Monetary data from -214,748.3648 to 214,748.3647
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807
float(n)	Floating precision number data from -1.79E + 308 to 1.79E + 308. The n parameter indicates whether the field should hold 4 or 8 bytes. float(24) holds a 4-byte field and float(53) holds an 8-byte field. Default value of n is 53.
real	Floating precision number data from -3.40E + 38 to 3.40E + 38

Date types:

Data type	Description
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute
date	Store a date only. From January 1, 0001 to December 31, 9999
time	Store a time only to an accuracy of 100 nanoseconds
datetimeoffset	The same as datetime2 with the addition of a time zone offset

timestamp	Stores a unique number that gets updated every time a row gets created or modified. The timestamp value is based upon an internal clock and does not correspond to real time. Each table may have only one timestamp variable

Other data types:

Data type	Description
sql_variant	Stores up to 8,000 bytes of data of various data types, except text, ntext, and timestamp
uniqueidentifier	Stores a globally unique identifier (GUID)
xml	Stores XML formatted data. Maximum 2GB
cursor	Stores a reference to a cursor used for database operations
table	Stores a result-set for later processing