Structured Query Language

Based on: W3schools SQL Tutorial

March 9, 2017

Gang Wang

SQL: 结构化查询语言

0. SQL语法:

大小写：SQL 对大小写不敏感，SELECT等效于select。

分号：分号是在数据库系统中分隔每条 SQL 语句的标准方法。某些数据库软件要求必须使用分号。如果使用的是 MS Access 和 SQL Server 2000，则分号可以省略。

注释:

单行注释: 以“--”开始. 在“--”与本行结尾中间的文本内容将被系统忽略。

多行注释: 以“/*”开始，并以“*/”结束，在“/*”与“*/”之间的部分将被系统忽略

1. 关于表内数据的操作

第一部分 查询

1.1 SELECT

SELECT 语句用于从表中选取数据。选取的结果被存储于一个结果表中（称为结果集），

其语法为：

```
SELECT column_name, column_name
FROM table_name;
```

以及：

```
SELECT * FROM table_name
```

星号（*）是选取所有列的快捷方式。

1.2 WHERE

如果需要有条件地从表中选取数据，可以用 WHERE语句声明条件。 其语法为：

```
SELECT column_name, column_name
FROM table_name
WHERE column_name operator value;
```

如下运算符（operator）经常用于 WHERE 子句中：

操作符	描述
=	等于
<>	不等于
>	大于
<	小于
>=	大于等于
<=	小于等于
BETWEEN	在某个范围内
LIKE	搜索某种模式

注：在某些版本的 SQL 中，操作符 <> 可以写为 !=。

SQL 使用单引号来环绕**文本值**（大部分数据库系统也接受双引号）。如果是**数值**，请不要使用引号。

1.2.1 AND & OR 运算符

AND 和 OR 可在 WHERE 子语句中把两个或多个条件结合起来。

如果第一个条件和第二个条件都成立，则 AND 运算符连接后结果为真。

如果第一个条件和第二个条件中只要有一个成立，则 OR 运算符连接后结果为真。

1.2.2 BETWEEN 操作符

操作符 BETWEEN ... AND 用于选取介于两值之间的数据。这些值可以是数值、文本或日期。其语法为：

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

1.2.3 LIKE 操作符

LIKE 操作符用于查找某列中符合指定模式的记录，模式通过通配符来表达。其语法为：

```
SELECT column_name(s)
FROM table_name
WHERE column_name LIKE pattern;
```

1.2.4 SQL通配符（Wildcard Characters）：

在搜索数据库中的数据时，通配符可以表示一个或多个字符。SQL 通配符必须与 LIKE 运算符一起使用。

在 SQL 中，可使用以下通配符：

通配符	含义
%	代表一个或多个字符
_	仅代表一个字符
[charlist]	字符列中的任何单一字符
[^charlist]或者[!charlist]	不在字符列中的任何单一字符

1.2.5 IN 操作符

IN 操作符表示某列的值在多个值的列表中。其语法为：

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1,value2,...);
```

1.2.6 NULL值

NULL 值是遗漏的未知数据。如果表中的某个列是可选的，则可以在不向该列添加值的情况下插入新记录或更新已有的记录，该字段将以 NULL 值保存。NULL 值的处理方式与其他值不同。

注：无法比较 NULL 和 0；它们是不等价的。

IS NULL

```
SELECT column_name(s)
FROM table_name
```

```
WHERE column_name IS NULL
```

IS NOT NULL

```
SELECT column_name(s)
FROM table_name
WHERE column_name IS NOT NULL
```

1.3 ORDER BY

ORDER BY 语句用于根据指定的列对结果集进行排序。ORDER BY 语句默认按照升序排列。如果希望按照降序排列，则可以使用 DESC 关键字。其语法为：

```
SELECT column_name, column_name
FROM table_name
ORDER BY column_name ASC/DESC, column_name ASC/DESC;
```

1.4 DISTINCT

在表中可能会包含重复值，如需仅列出唯一不同的值，则使用关键词 DISTINCT。

其语法为：

```
SELECT DISTINCT column_name, column_name
FROM table_name;
```

1.5 ALIASES

通过使用 SQL，可以为列名称和表名称指定别名（Alias）

为列指定别名的语法为：

```
SELECT column_name AS alias_name
FROM table_name;
```

为表指定别名的语法为：

```
SELECT column_name(s)
FROM table_name AS alias_name;
```

1.6 GROUP BY

GROUP BY 语句与合计函数配合使用，根据一个或多个列对结果集进行分组。其语法为：

```
SELECT column_name, aggregate_function(column_name)
FROM table_name
WHERE column_name operator value
GROUP BY column_name;
```

1.7 HAVING

在 SQL 中增加 HAVING 子句原因是，WHERE 无法与合计函数配合使用。其语法为：

```
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;
```

1.8 JOIN

有时为了得到完整的结果，需要从两个或更多的表中获取结果。这时就需要JOIN语句。

下面列出了所有 JOIN 类型，以及它们之间的差异。

语句	含义
JOIN	如果表中有至少一个匹配，则返回行
LEFT JOIN	即使右表中没有匹配，也从左表返回所有的行
RIGHT JOIN	即使左表中没有匹配，也从右表返回所有的行
FULL JOIN	只要其中一个表中存在匹配，就返回行

1.8.1 INNER JOIN

两表中的记录如果满足连接条件，则返回该记录，不满足连接条件的记录不返回。INNER JOIN 与 JOIN 是相同的。其语法为：

```
SELECT column_name(s)
FROM table1
INNER JOIN table2
ON table1.column_name=table2.column_name;
```

或者：

```
SELECT column_name(s)
FROM table1
JOIN table2
ON table1.column_name=table2.column_name;
```

1.8.2 LEFT JOIN

LEFT JOIN 会从左表 (table1) 那里返回所有的行，右表 (table2) 中符合匹配条件的则返回该行，不匹配的相应数据返回空值。其语法为：

```
SELECT column_name(s)
FROM table1
LEFT JOIN table2
ON table1.column_name=table2.column_name;
```

或者：

```
SELECT column_name(s)
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

注：在某些数据库中， LEFT JOIN 称为 LEFT OUTER JOIN。

1.8.3 RIGHT JOIN

RIGHT JOIN 会从右表 (table2) 那里返回所有的行，左表 (table1) 中符合匹配条件的则返回该行，不匹配的相应数据返回空值。其语法为：

```
SELECT column_name(s)
FROM table1
RIGHT JOIN table2
ON table1.column_name=table2.column_name;
```

或者：

```
SELECT column_name(s)
FROM table1
RIGHT OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

注：在某些数据库中， RIGHT JOIN 称为 RIGHT OUTER JOIN。

1.8.4 FULL JOIN

FULL JOIN 会从左表 (table1) 和右表 (table2) 那里返回所有的行。如果 table1中的行在表table2中没有匹配，或者如果table2中的行在表 table1 中没有匹配，这些行同样会列出。其语法为：

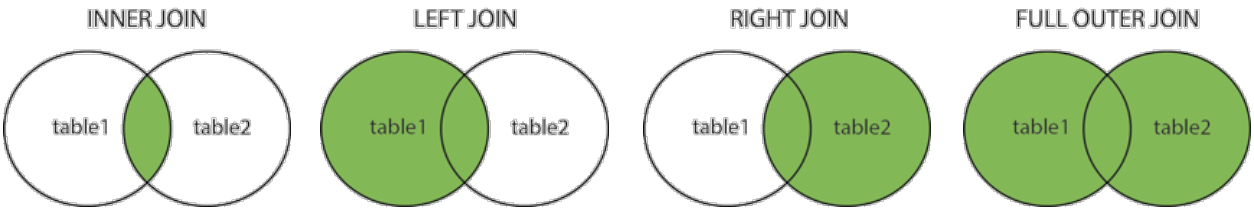
```
SELECT column_name(s)
FROM table1
FULL JOIN table2
ON table1.column_name=table2.column_name;
```

或者：

```
SELECT column_name(s)
FROM table1
FULL OUTER JOIN table2
ON table1.column_name=table2.column_name;
```

注：在某些数据库中， FULL JOIN 称为 FULL OUTER JOIN

1.8.5 不同的JOIN方式的返回结果示意



1.9 UNION

UNION 操作符用于合并两个或多个 SELECT 语句的结果集。

注：UNION 内部的 SELECT 语句必须拥有相同数量的列。相应列也必须拥有相似的数据类型。同时，每条 SELECT 语句中的列的顺序必须相同。

UNION的语法为：

```
SELECT column_name(s) FROM table1
UNION
```

```
SELECT column_name(s) FROM table2;
```

注：默认地，UNION 操作符选取不同的值。如果允许重复的值，请使用 UNION ALL。

UNION ALL的语法为：

```
SELECT column_name(s) FROM table1
UNION ALL
SELECT column_name(s) FROM table2;
```

UNION 结果集中的列名总是等于 UNION 中第一个 SELECT 语句中的列名。

第二部分 添加

1.10 INSERT INTO

INSERT INTO 语句用于向表中插入新的行。其语法为：

```
INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

也可以指定所要插入数据的列中列名与值的对应关系：

```
INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

1.11 SELECT INTO

SELECT INTO 语句从一个表中选取数据，然后把数据插入另一个表中。SELECT INTO 语句常用于创建表的备份复件或者用于对记录进行存档。语法为：

复制table1中所有列：

```
SELECT *
INTO newtable [IN externaldb]
FROM table1;
```

复制table1中的制定列：

```
SELECT column_name(s)
INTO newtable [IN externaldb]
FROM table1;
```

1.12 INSERT INTO SELECT

INSERT INTO SELECT语句用于从表中选择数据，然后加入到已经存在的表中要求选择的数据与要加入的表的数据类型一一对应。其语法为：

选择全部数据加入已存在的表:

```
INSERT INTO table2
SELECT * FROM table1;
```

选择部分数据加入已存在的表:

```
INSERT INTO table2
(column_name(s))
SELECT column_name(s)
FROM table1;
```

第三部分 修改

1.13 UPDATE

Update 语句用于修改表中的数据，其语法为：

```
UPDATE table_name
SET column1=value1,column2=value2,...
WHERE some_column=some_value;
```

第四部分 删除

1.14 DELETE

DELETE语句用于删除表中的数据，其语法为：

```
DELETE FROM table_name
WHERE some_column=some_value;
```

2. 关于数据库与表的操作

2.1 CREATE DATABASE

CREATE DATABASE 用于创建数据库，其语法为：

```
CREATE DATABASE dbname;
```

2.2 DROP DATABASE

DROP DATABASE 用于删除数据库，其语法为：

```
DROP DATABASE database_name
```

2.3 CREATE TABLE

CREATE TABLE 语句用于创建数据库中的表。其语法为：

```
CREATE TABLE table_name
(
column_name1 data_type(size),
column_name2 data_type(size),
column_name3 data_type(size),
....
);
```

数据类型（data_type）规定了该列允许何种类型数据。下表罗列了SQL中最常用的数据类型：

数据类	描述
integer(size) int(size) smallint(size) tinyint(size)	整数。在括号内规定数字的最大位数。
decimal(size,d) numeric(size,d)	带有小数的数字。 "size" 规定数字的最大位数。"d" 规定小数点右侧的最大位数。
char(size)	固定长度的字符串（可为字母、数字以及特殊字符）。 在括号中规定字符串的长度。
varchar(size)	可变长度的字符串（可为字母、数字以及特殊的字符）。 在括号中规定字符串的最大长度。
date(yyyymmdd)	日期。

2.4 ALTER TABLE

ALTER TABLE 语句用于在已有的表中添加、修改或删除列。其语法为：

如需在表中添加列，使用下列语法:

```
ALTER TABLE table_name
ADD column_name datatype
```

要删除表中的列，使用下列语法：

```
DROP COLUMN column_name
```

注：某些数据库系统不允许这种在数据库表中删除列的方式 (DROP COLUMN column_name)。

要改变表中列的数据类型，请使用下列语法： ALTER TABLE table_name

SQL Server / MS Access:

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype
```

My SQL / Oracle (prior version 10G):

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype
```

Oracle 10G and later:

```
ALTER TABLE table_name
MODIFY column_name datatype
```

2.5 CONSTRAINTS

约束（CONSTRAINTS）用于限制数据的类型。可以在创建表时规定约束（通过 CREATE TABLE 语句），也可以在表创建之后规定（通过 ALTER TABLE 语句）。

CREATE TABLE + CONSTRAINT 的语法为：

```
CREATE TABLE table_name
(
column_name1 data_type(size) constraint_name,
column_name2 data_type(size) constraint_name,
column_name3 data_type(size) constraint_name,
....
);
```

SQL中常见如下约束：

NOT NULL约束 - 强制该列不能为 NULL 值。即：强制该字段始终包含值。如果不向字段添加值，就无法插入新记录或者更新记录。

UNIQUE 约束 - 该列的值不能重复。UNIQUE 和 PRIMARY KEY 约束均为列或列集合提供了唯一性的保证。PRIMARY KEY 拥有自动定义的 UNIQUE 约束。每个表可以有多个 UNIQUE 约束，但是每个表只能有一个 PRIMARY KEY 约束。

主键（PRIMARY KEY）约束 - 唯一标识数据库表中的每条记录。主键必须包含唯一的值。主键列不能包含 NULL 值。每个表都应该有一个主键，并且每个表只能有一个主键。

外键（FOREIGN KEY）约束 - 一个表中的 FOREIGN KEY 指向另一个表中的 PRIMARY KEY。用于预防破坏表之间连接的动作。 FOREIGN KEY 约束也能防止非法数据插入外键列，因为它必须是它指向的那个表中的值之一。

CHECK 约束 - 用于限制列中的值的范围。如果对单个列定义 CHECK 约束，那么该列只允许特定的值。如果对一个表定义 CHECK 约束，那么此约束会在特定的列中对值进行限制。

DEFAULT 约束 - 用于向列中插入默认值。如果没有规定其他的值，那么会将默认值添加到所有的新记录。

AUTO INCREMENT约束 - 我们经常会希望某主键所在列在每次增加新纪录的时候其数值可以自动增加，此时需要AUTO INCREMENT约束

2.6 DROP TABLE

DROP TABLE用于删除数据库中的表。其语法为：

```
DROP TABLE table_name
```

2.7 TRUNCATE TABLE

如果仅仅清空表内数据而并不删除表本身，则使用TRUNCATE TABLE语句，其语法为：

```
TRUNCATE TABLE table_name
```

3. 关于索引的操作

INDEXES

可以在表中创建索引，以便更加快速高效地查询数据。用户无法看到索引，它们只能被用来加速搜索/查询。注：更新一个包含索引的表要比更新一个没有索引的表花费更多的时间，这是由于索引本身也需要更新。因此，理想的做法是仅仅在常常被搜索的列（或表）上面创建索引。

3.1 CREATE INDEX

在表上创建一个简单的索引。允许使用重复的值：

```
CREATE INDEX index_name
ON table_name (column_name)
```

3.2 CREATE UNIQUE INDEX

在表上创建一个唯一的索引。唯一的索引意味着两个行不能拥有相同的索引值。

```
CREATE UNIQUE INDEX index_name
ON table_name (column_name)
```

3.3 DROP INDEX

DROP INDEX语句用于删除表内的索引

MS Access的语法：

```
DROP INDEX index_name ON table_name
```

MS SQL Server的语法：

```
DROP INDEX table_name.index_name
```

DB2/Oracle的语法：

```
DROP INDEX index_name
```

MySQL的语法：

```
ALTER TABLE table_name DROP INDEX index_name
```

4. 关于视图的操作

VIEWS

在 SQL 中，视图是基于 SQL 语句的结果集的可视化的表。视图包含行和列，就像一个真实的表。视图中的字段就是来自一个或多个数据库中的真实的表中的字段。我们可以向视图添加 SQL 函数、WHERE 以及 JOIN 语句，我们也可以提交数据，就像这些来自于某个单一的表。

注：数据库的设计和结构不会受到视图中的函数、where 或 join 语句的影响。

4.1 CREATE VIEW

其语法为：

```
CREATE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

视图总是显示最近的数据。每当用户查询视图时，数据库引擎通过使用 SQL 语句来重建数据。

4.2 Updating a View

```
CREATE OR REPLACE VIEW view_name AS
SELECT column_name(s)
FROM table_name
WHERE condition
```

4.3 Dropping a View

```
DROP VIEW view_name
```

SQL References

AGGREGATE FUNCTIONS

SQL aggregate functions return a single value, calculated from values in a column.

Function	Description
AVG()	Returns the avera
COUNT()	Returns the number of rows
FIRST()	Returns the first value
LAST()	Returns the last value
MAX()	Returns the largest value
MIN()	Returns the smallest value
ROUND()	Rounds a numeric field to the number of decimals specified
SUM()	Returns the sum

STRING FUNCTIONS

Function	Description
CHARINDEX	Searches an expression in a string expression and returns its starting position if found
CONCAT()	
LEFT()	
LEN() / LENGTH()	Returns the length of the value in a text field
LOWER() / LCASE()	Converts character data to lower case
LTRIM()	
SUBSTRING() / MID()	Extract characters from a text field
PATINDEX()	
REPLACE()	
RIGHT()	
RTRIM()	
UPPER() / UCASE()	Converts character data to upper case

DATE FUNCTIONS

MySQL Date Functions

Function	Description
NOW()	Returns the current date and time
CURDATE()	Returns the current date
CURTIME()	Returns the current time
DATE()	Extracts the date part of a date or date/time expression
EXTRACT()	Returns a single part of a date/time
DATE_ADD()	Adds a specified time interval to a date
DATE_SUB()	Subtracts a specified time interval from a date
DATEDIFF()	Returns the number of days between two dates
DATE_FORMAT()	Displays date/time data in different formats

SQL Server Date Functions

Function	Description
GETDATE()	Returns the current date and time
DATEPART()	Returns a single part of a date/time
DATEADD()	Adds or subtracts a specified time interval from a date
DATEDIFF()	Returns the time between two dates
CONVERT()	Displays date/time data in different formats

SQL Date and Time Data Types and Functions

Function	Description
FORMAT()	Formats how a field is to be displayed
NOW()	Returns the current system date and time

SQL Dates

The most difficult part when working with dates is to be sure that the format of the date you are trying to insert, matches the format of the date column in the database.

As long as your data contains only the date portion, your queries will work as expected. However, if a time portion is involved, it gets more complicated.

SQL Date Data Types

MySQL comes with the following data types for storing a date or a date/time value in the database:

Data Type	Format
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MI:SS
TIMESTAMP	YYYY-MM-DD HH:MI:SS
YEAR	YYYY or YY

SQL Server comes with the following data types for storing a date or a date/time value in the database:

Data Type	Format
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MI:SS
SMALLDATETIME	YYYY-MM-DD HH:MI:SS
TIMESTAMP	a unique number

Note: The date types are chosen for a column when you create a new table in your database!

SQL OPERATORS

Arithmetic Operators

Operator	Description
+	Add
-	Subtract
*	Multiply
/	Divide
%	Modulo

Bitwise Operators

Operator	Description
&	Bitwise AND
|	Bitwise OR
^	Bitwise exclusive OR

Comparison Operators

Operator	Description
=	Equal to
>	Greater than

Operator	Description
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
<>	Not equal to

Compound Operators

Operator	Description
+=	Add equals
-=	Subtract equals
*=	Multiply equals
/=	Divide equals
%=	Modulo equals
&=	Bitwise AND equals
^-=	Bitwise exclusive equals
l*=	Bitwise OR equals

Logical Operators

Operator	Description
ALL	TRUE if all of a set of comparisons are TRUE
AND	TRUE if both expressions are TRUE
ANY	TRUE if any one of a set of comparisons are TRUE
BETWEEN	TRUE if the operand is within the range of comparisons
EXISTS	TRUE if a subquery contains any rows
IN	TRUE if the operand is equal to one of a list of expressions
LIKE	TRUE if the operand matches a pattern
NOT	Reverses the value of any other operator
OR	TRUE if either expression is TRUE
SOME	TRUE if some of a set of comparisons are TRUE

SQL GENERAL DATA TYPES

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

SQL developers have to decide what types of data will be stored inside each and every table column when creating a SQL table. The data type is a label and a guideline for SQL to understand what type of data is expected inside of each column, and it also identifies how SQL will interact with the stored data. 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, mantissa precision p. 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, mantissa precision 7
FLOAT	Approximate numerical, mantissa precision 16
DOUBLE PRECISION	Approximate numerical, mantissa precision 16
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

SQL Data Type Quick Reference
However, different databases offer different choices for the data type definition.
The following table shows some of the common names of data types between the various database platforms:

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

Note: Data types might have different names in different database. And even if the name is the same, the size and other details may be different! Always check the documentation!

SQL DATA TYPES FOR VARIOUS DBS

Microsoft Access Data Types

Data type	Description	Storage
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	1 byte
Integer	Allows whole numbers between -32,768 and 32,767	2 bytes
Long	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
Single	Single precision floating-point. Will handle most decimals	4 bytes
Double	Double precision floating-point. Will handle most decimals	8 bytes
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	8 bytes
AutoNumber	AutoNumber fields automatically give each record its own number, usually starting at 1	4 bytes

Data type	Description	Storage
Date/Time	Use for dates and times	8 bytes
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	1 bit
Ole Object	Can store pictures, audio, video, or other BLOBs (Binary Large OBjects)	up to 1GB
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	4 bytes

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'

Data type	Description
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	Storage
char(n)	Fixed width character string. Maximum 8,000 characters	Defined width
varchar(n)	Variable width character string. Maximum 8,000 characters	2 bytes + number of chars
varchar(max)	Variable width character string. Maximum 1,073,741,824 characters	2 bytes + number of chars
text	Variable width character string. Maximum 2GB of text data	4 bytes + number of chars
nchar	Fixed width Unicode string. Maximum 4,000 characters	Defined width x 2
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	Storage
tinyint	Allows whole numbers from 0 to 255	1 byte

Data type	Description	Storage
smallint	Allows whole numbers between -32,768 and 32,767	2 bytes
int	Allows whole numbers between -2,147,483,648 and 2,147,483,647	4 bytes
bigint	Allows whole numbers between -9,223,372,036,854,775,808 and 9,223,372,036,854,775,807	8 bytes
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	5-17 bytes
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	5-17 bytes
smallmoney	Monetary data from -214,748.3648 to 214,748.3647	4 bytes
money	Monetary data from -922,337,203,685,477.5808 to 922,337,203,685,477.5807	8 bytes
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.	4 or 8 bytes
real	Floating precision number data from -3.40E + 38 to 3.40E + 38	4 bytes

Date types:

Data type	Description	Storage
datetime	From January 1, 1753 to December 31, 9999 with an accuracy of 3.33 milliseconds	8 bytes
datetime2	From January 1, 0001 to December 31, 9999 with an accuracy of 100 nanoseconds	6-8 bytes
smalldatetime	From January 1, 1900 to June 6, 2079 with an accuracy of 1 minute	4 bytes
date	Store a date only. From January 1, 0001 to December 31, 9999	3 bytes
time	Store a time only to an accuracy of 100 nanoseconds	3-5 bytes
datetimeoffset	The same as datetime2 with the addition of a time zone offset	8-10 bytes
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

SQL Quick Reference

SQL Statement	Syntax
AND / OR	SELECT column_name(s) FROM table_name WHERE condition AND/OR condition
ALTER TABLE	ALTER TABLE table_name ADD column_name datatype or ALTER TABLE table_name DROP COLUMN column_name
AS (alias)	SELECT column_name AS column_alias FROM table_name or SELECT column_name FROM table_name AS table_alias
BETWEEN	SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2
CREATE DATABASE	CREATE DATABASE database_name CREATE TABLE CREATE TABLE table_name (column_name1 data_type, column_name2 data_type, column_name3 data_type, ...)
CREATE INDEX	CREATE INDEX index_name ON table_name (column_name) or CREATE UNIQUE INDEX index_name ON table_name (column_name)
CREATE VIEW	CREATE VIEW view_name AS SELECT column_name(s) FROM table_name WHERE condition
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!!)

SQL Statement	Syntax
DROP DATABASE	DROP DATABASE database_name 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)
DROP TABLE	DROP TABLE table_name

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

GROUP BY	SELECT column_name, aggregate_function(column_name) FROM table_name WHERE column_name operator value GROUP BY column_name
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
IN	SELECT column_name(s) FROM table_name WHERE column_name IN (value1,value2,..)
INSERT INTO	INSERT INTO table_name VALUES (value1, value2, value3,....) or INSERT INTO table_name (column1, column2, column3,...) VALUES (value1, value2, value3,....)
INNER JOIN	SELECT column_name(s) FROM table_name1 INNER JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LEFT JOIN	SELECT column_name(s) FROM table_name1 LEFT JOIN table_name2 ON table_name1.column_name=table_name2.column_name

SQL Statement	Syntax
RIGHT JOIN	SELECT column_name(s) FROM table_name1 RIGHT JOIN table_name2 ON table_name1.column_name=table_name2.column_name
FULL JOIN	SELECT column_name(s) FROM table_name1 FULL JOIN table_name2 ON table_name1.column_name=table_name2.column_name
LIKE	SELECT column_name(s) FROM table_name WHERE column_name LIKE pattern
ORDER BY	SELECT column_name(s) FROM table_name ORDER BY column_name [ASC/DESC]
SELECT	SELECT column_name(s) FROM table_name
SELECT *	SELECT * FROM table_name
SELECT DISTINCT	SELECT DISTINCT column_name(s) FROM table_name
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 TOP	SELECT TOP number|percent column_name(s) FROM table_name
TRUNCATE TABLE	TRUNCATE TABLE table_name
UNION	SELECT column_name(s) FROM table_name1 UNION SELECT column_name(s) FROM table_name2
UNION ALL	SELECT column_name(s) FROM table_name1 UNION ALL SELECT column_name(s) FROM table_name2
UPDATE	UPDATE table_name SET column1=value, column2=value,... WHERE some_column=some_value
WHERE	SELECT column_name(s) FROM table_name WHERE column_name operator value