**Types of computational data.**

**1. Numeric Data.**

Numeric data types are numbers stored in database columns. These data types are typically grouped by:

* **Exact** numeric types, values where the precision and scale need to be preserved. The exact numeric types are INTEGER, BIGINT, DECIMAL, NUMERIC, NUMBER, and MONEY.
* **Approximate** numeric types, values where the precision needs to be preserved and the scale can be floating. The approximate numeric types are DOUBLE PRECISION, FLOAT, and REAL.

String-to-numeric data type conversions accept formats of quoted constants for scientific notation, binary scaling, hexadecimal, and combinations of numeric-type literals:

* Scientific notation:  
  => SELECT FLOAT '1e10';
* ?column?
* -------------
* 10000000000
* (1 row)
* BINARY scaling:  
  => SELECT NUMERIC '1p10';
* ?column?
* ----------
* 1024
* (1 row)
* hexadecimal:  
  => SELECT NUMERIC '0x0abc';
* ?column?
* ----------
* 2748
* (1 row)

1. **Logic Data.**

The only logical data type is boolean. A boolean field can store true, false, and null. The boolean data type is not standard SQL.

The following table describes the logical data type.

*Table 1. Logical data types*

|  |  |  |
| --- | --- | --- |
| **Type** | **Value** | **Disk Usage** |
| boolean (alias bool) | With value true (t) or false (f). | 1 byte |

You can use the following words to specify booleans:

* True or false
* On or off
* ‘0’ or ‘1’
* “true’ or ‘false’
* ‘t’ or ‘f’
* ‘on’ or ‘off’
* ‘yes’ or ‘no’

**3. String and characters.**

Stores strings of letters, numbers, and symbols. Data types CHARACTER (CHAR) and CHARACTER VARYING (VARCHAR) are collectively referred to as *character string types,* and the values of character string types are known as *character strings*.

Character data can be stored as fixed-length or variable-length strings. Fixed-length strings are right-extended with spaces on output; variable-length strings are not extended.

String literals in SQL statements must be enclosed in single quotes.

**4. Arrangements.**

In computer science, an **array type** is a data type that represents a collection of *elements* ([values](https://en.wikipedia.org/wiki/Value_(computer_science)) or variables), each selected by one or more indices (identifying keys) that can be computed at run time during program execution. Such a collection is usually called an **array variable**, **array value**, or simply **array**.

**Representation of computational data.**

**1. Variables.**

A variable is an identifier which is used to store some value. Constants can never change at the time of execution. Variables can change during the execution of a program and update the value stored inside it.

A single variable can be used at multiple locations in a program. A variable name must be meaningful. It should represent the purpose of the variable.

**2. Constants.**

Constants are the fixed values that never change during the execution of a program. Following are the various types of constants:

Integer constants

An integer constant is nothing but a value consisting of digits or numbers. These values never change during the execution of a program. Integer constants can be octal, decimal and hexadecimal.

**Types of computational operations.**

**-Operators:**

**1. Arithmetic.**

The basic **arithmetic operations** are addition, subtraction, multiplication, and division. Arithmetic is performed according to an order of operations.

**Discussion**

An operator performs an action on one or more operands. The common arithmetic operators are:

|  |  |
| --- | --- |
| **Action** | **Common Symbol** |
| Addition | + |
| Subtraction | - |
| Multiplication | \* |
| Division | / |
| Modulus (associated with integers) | % |

These arithmetic operators are binary that is they have two operands. The operands may be either constants or variables.

**2. Logic.**

A **logical operator** is a symbol or word used to connect two or more expressions such that the value of the compound expression produced depends only on that of the original expressions and on the meaning of the operator. Common logical operators include AND, OR, and NOT.

|  |  |  |  |
| --- | --- | --- | --- |
| **Language** | **AND** | **OR** | **NOT** |
| C++ | && | || | ! |
| C# | && | || | ! |
| Java | && | || | ! |
| JavaScript | && | || | ! |
| Python | and | or | not |
| Swift | && | || | ! |

**3. Relationships.**

A **relational operator** is a programming language construct or operator that tests or defines some kind of relation between two entities. These include numerical equality (e.g., 5 = 5) and inequalities (e.g., 4 ≥ 3).

|  |  |
| --- | --- |
| Operator | Meaning |
| < | less than |
| > | greater than |
| <= | less than or equal to |
| >= | greater than or equal to |
| “==“ | equality (equal to) |
| != or <> | inequality (not equal to) |

* **Operands:**

1. **Variables and Constants.**

The variables and constants together are called ‘operands’ that are operated upon by the ‘arithmetic operators’ and the result is assigned, using the assignment operator, to the variable on left-hand side. Expressions are basically operators acting on operands. Statements like a = b + 3, ++z, a-- and 300 > (8 \* k) are all expressions. Strictly speaking, even a single variable or constant can be considered an expression.

* **Expressions:**

**1. Arithmetic**

An **arithmetic expression** is an expression that results in a numeric value. There are two kinds of numeric values, **integers** (whole numbers), and **real** or **floating point** numbers (numbers containing a decimal point).

The simplest arithmetic expressions are **literals** (the number itself, written with digits) and variables (named values):

|  |  |
| --- | --- |
| **Example** | **Description** |
| **12** | A literal integer, representing the number 12. |
| **-5** | A literal integer, representing the number negative 5. |
| **-5.0** | A literal real, representing the number negative 5. |
| **5.0E+04** | A literal real, representing the number 50000. |
| **count** | A variable. If it was declared as **int count**, it will hold an integer value; but if declared as **double count**, it will hold a real value. |

**2. Logic**

Logical expressions (also called Boolean expressions) are the result of applying logical (Boolean) operators to relational or arithmetic expressions. The result of an operation has two possible states: true or false. Logical expressions are considered false when equal to 0, and are considered true when nonzero.

Logical operators have the lowest precedence and are evaluated after all other operations have been evaluated. If two or more logical operators appear in an expression, the leftmost operator is performed first.

Logical operators act on their associated operands as illustrated below:

|  |  |
| --- | --- |
| a or b is true (evaluates to 1) | if a is true or b is true. |
| a ! b is false (evaluates to 0) | if a and b are both false. |
| a and b is true (evaluates to 1) | only if both a and b are true. |
| a & b is false (evaluates to 0) | if a is false or b is false or both are false. |

The **not()** function negates the effect of a logical expression:

|  |  |
| --- | --- |
| not(a or b) is true (evaluates to 1) | if a is false or b is false. |
| not(a and b) is true (evaluates to 1) | if a and b are false. |