Basics of SQL

Stefan Cliff 2019/230449<efas of SQL - S

# Introduction to SQL – Basics of SQL

The programming language **SQL** (**Structured Query Language**) is known as a domain-specific language which is used in programming and is designed for managing data inside of **RDBM**’s (**Relational Database Management Systems**). Its usefulness becomes more apparent when handling structured data (i.e. data that has relations amongst various entities and variables).

The two main advantages **SQL** has over older read-write **API’s** (application programming interface) are:

* It introduced the concept of multiple records being accessed with a single command
* The need for a specified *how* to reach a record was eliminated. *The simplest example is reaching a record even without its index.*

**SQL** started out originally based upon *relational algebra* and *tuple relations calculus*, and consists of multiple types of statements (also classified as sublanguages by many): a *data query language* (**DQL**); a *data definition language* (**DDL**); a *data control language* (**DCL**) and lastly a *data manipulation language* (**DML**). Although **SQL** is a *declarative language* (**4GL**), it also includes procedural elements.

With all of these sublanguages, the scope of **SQL** includes: data query*,* data manipulation (*delete, update, insert*)*,* data definition *(schema creation and modification)* and lastly data access control. While **SQL** was one of the first commercial languages that used Edgar F. Codd’s relational model, it didn’t entirely follow all the rules/guidelines he wrote. Still, it soon became the most widely used database language.

Back in 1986, **SQL** became a standard for the American National Standards Institute (**ANSI**) and the following year was also recognized by the International Organization for Standardization (**ISO**). Ever since then, as a standard, it has been revised again and again to include a larger set of features. Furthermore, despite the actual existence of the standards that have been laid out over all this time, most **SQL** projects/codes require minor alterations as to be port-able to different database systems.

# History of SQL – Basics of SQL

**SQL** was initially developed, following Codd’s paper on relational models back in the early 70’s, by Donald D. Chamberlin and his colleague Raymond F. Boyce while they both worked at IBM. Its original name was **SEQUEL** (**Structured English Query Language**), whose purpose was originally to retrieve and manipulate data that was stored inside of IBM’s original *quasi-relational* database management system, **System R** (***System R*** *was originally developed at IBM’s San Jose Research Laboratory back in the 1970’s*).

Their (*Chamberlin and Boyce*) first attempt at making a relational database language was called **Square**, however it was too difficult to use due to the subscript notation. Following their move to the San Jose labs in 1973, that is where they truly began working on developing **SEQUEL**. Fun fact, the name **SEQUEL** had to be changed to **SQL** due to the fact that the name was already trademarked by the UK-based Hawker Siddeley Dynamics Engineering Limited company.

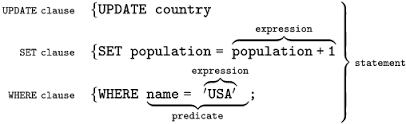
Following **SQL**’s testing at customer test sites, to figure out the systems practicality and actual usefulness, IBM started development on a commercial product using a **System R** prototype as the base. These products became **System/38** (1979), **SQL/DS** (1981) and **DB2** (1983).

Towards the end of the 1970’s, Oracle Corporation (then known as Relational Software Inc.) found that there was potential within the concepts created by Codd, Chamberlin and Boyce, which lead to them begining development of their own **SQL**-based relational database management system (**RDBMS**) with the intent and hope that they could sell it the the U.S. Navy; Central Intelligence Agency (**CIA**) and a few other U.S. goverment agencies. Which lead to the release of their first commercially avaliable implementation of **SQL**, **Oracle V2** (Version 2) specifically for **VAX** computers in June of 1979.

# Syntax of SQL – Basics of SQL

The **SQL** language can be subdivided into multiple language elements, such as:

* Expressions, these produce either scalar values or tables which consist of columns and rows
* Queries, which retrieve data specified by some criteria and is one of the core elements of **SQL**.
* Statements, which may have some constant effect on the schemata/data or it can: control transactions; the flow of the program; diagnostics of the program or the sessions themselves. **SQL** statements also include a semicolon ‘;’ as a statement terminator and although might not be required on every platform, it is predefined as a standard part of **SQL**’s grammar.
* Clauses, these are constituent components of statements and queries, which aren’t mandatory.
* Predicates, which specify a condition/s that can be evaluated to either **SQL**’s *three-value logic* (**3VL**) (true, false or unknown/NULL) or *Boolean truth values* (*true* and *false*). They are used to limit the effects of statements and queries and or to change the programs flow.
* Insignificant whitespace, while yes technically is a part of the **SQL** statements or queries, it is ignored as the make it easier to format **SQL** code for readability.



*This is an example of an* ***SQL*** *statement’s syntax*

# Criticisms of SQL – Basics of SQL

While **SQL** is without a doubt one of, if not the most famous query language, it is not free of criticisms and flaws:

* **Design**

**SQL** deviated from its theoretical foundations, the relational model and tuple calculus in multiple ways for example:

* + In the relational model a table is a set of tuples, while in **SQL** both tables and query results are both lists of rows. Meaning that the same row may occur multiple times and whose order of rows can be employed in queries. Many critics argue that **SQL** should be replaced with a new language whose design follows the original foundation more strictly. Many also say that it is quite possible that **SQL** can be *fixed* or improved in this regard such that a need for a switch to a completely new and different query language isn’t needed.
* **Null**

The very idea and concept of NULL has long been subject of some debate. Its marker indicates a lack of value and is therefore different from 0 (zero) in an integer column or an empty string in a text column. The concept of NULLs enforces the *three-value logic* in **SQL**, which is a concrete implementation of the general *three-value logic*.

* **Duplicates**

One of the more popular criticisms of **SQL** is that it allows the duplication of rows, which makes the integration with other programming languages (*such as Python*) rather difficult in the terms of parsing and by the absence of modularity. However, this can be avoided simply by declaring unique constraints with one or more fields that identify a row in a table (*that constraint could also become the PK/primary key of the table if needed*).

* **Other criticism**

Early on it did not support major features, such as primary keys. Sets of results could not be named and subqueries could not be defined, leading many to believe that **SQL** was not complete. However, back in 1992. these criticisms were heard and answered with the additions of those features.

# SQL Data types – Basics of SQL

Within **SQL** standards, there are three kinds of data types. These are:

* + Predefined data types;
  + Constructed data types;
  + User-defined data types;

A predefined data type is exactly as it sounds, a data type that is out-of-the-box understood by **SQL**. On the other hand, a constructed data type is a made by a combination of data, for example: ARRAY, MULTISET, REF/REFERENCE or a ROW. Lastly, we have user-defined data types. They are similar to classes in *object-oriented programming* languages that consist of their won constructs, methods, interfaces and so on and so forth.

Each predefined data type has its full name but also an easily recognizable or understandable short hand. There are roughly 25 (twenty five) predefined data types and they are as follows:

* Characters types
  + Character (CHAR)
  + Varying character (VARCHAR)
  + Large character object (CLOB)
* National character types
  + National character (NCHAR)
  + National varying character (NVARCHAR)
  + National large character object (NCLOB)
* Binary types
  + Binary (BINARY)
  + Varying binary (VARBINARY)
  + Large binary object (BLOB)
* Numeric types
  + Exact numeric (NUMERIC, INTEGER, BIGINT, SMALLINT, DECIMAL)
  + Approximate numeric (FLOAT, REAL, DOUBLE PRECISION)
  + Decimal floating-point (DECFLOAT)
* Other
  + Date-time types (DATE, TIME, TIMESTAMP)
  + Interval type (INTERVAL)
  + Boolean
  + XML
  + JSON

Thanks for reading/listening

Stefan Cliff 2019/230449<efas of SQL - S