Structured Query Language (SQL)
By Donald W. Larson
Copyright © 1994 by EveryDay Objects, Inc.

From Data to Databases

Data, simply defined, are nothing more than facts about entities. Entities generally represent any object or thing that is identifiable. Attributes of entities help describe recognizable features of the entity. Some attributes of a vehicle entity might be:

- Color
- Engine size
- Number of doors
- Weight

An entity may be associated with other entities. When this situation occurs it can be termed a relationship. Relationships between entities may take the form of:

- One-to-one
- One-to-many
- Many-to-many

Relationships that are useful usually need some kind of organization to become manageable. Databases are created to handle situations such as these. Once a database becomes useful, the maintenance of its contents and how to access it becomes a priority. Database Management Systems (DBMS) were created to handle those priority tasks at hand.

DBMS Fundamentals

Internal levels of databases are required because they deal with how the actual data is physically stored on the disk. These are by nature low-level routines that are not directly accessed by the users of the database. Direct access by users would require those users to understand much more about the database than they need to for most tasks. Instead, users work at a conceptual level or by understanding a database model where the database is viewed at its logical representations rather than physical actualities.

Three types of traditional computer models exist:

- Hierarchical
  - Entities are modeled in the form of a tree
  - Usually model one-to-one or one-to-many relationships
  - A pointer in the child entity points to the parent
  - Restrictions:
    -- Parent entities must exist before child entities
    -- Child entities can only be related to one parent entity

---

1 Alok Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 4
2 Alok Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 8
3 Alok Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 9-11
--- Grandchild entities cannot relate to grandparent entities directly

**Network**
- Usually model one-to-one or one-to-many relationships
- Child entities can be related to more than one parent entity
- In each entity a pointer exists for each direction of the relationship
- Restriction:
  -- If relationships change, much updating is required to rearrange the new structure

**Relational**
- Tables are made up of columns or attributes
- Entities constitute the row aspect of the tables
- All three types of data relationships are easily supported
- No pointers exist as such; procedures cause the runtime "joins" between tables allowing new relationships on-the-fly
- Restriction:
  -- The attributes should be normalized to remove any embedded relationships

### Database Languages

A language is needed to communicate with a database. DBMS languages usually include a way to handle data definition and data manipulations.

Query language is defined as a stand-alone way to interact with the database and handles the data manipulation requirements. This functionality may do more than just issue queries.

Other constructs of the language are designed as an application program interface (API) and allows other applications to create the database structure.4

Relational database languages have been based on mathematical constructs such as:5

**Relational Algebra**
- Traditional operators
  -- Union
  -- Intersection
  -- Difference
- Special operators
  -- Projection
  -- Restriction
  -- Join
  -- Product
  -- Division

**Relational Calculus**
- A query variable that ranges over the rows of a table
- A target which specifies the columns (attributes) to be returned
- A qualification which selects rows of the table based on an expression involving the query variable

**Mapping-Oriented**

---

4 Alok Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 15
5 Alok Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 16-19
- Results of queries which of themselves be nested within other queries

**Relational Completeness**

In 1970, E.F. Codd wrote a paper entitled "A Relational Model Of Data For Large Shared Data Banks". In that paper, Codd defined several rules to assess the degree of conformance of a database system to the relational model. They are known as "Codd's 12 Rules" (even if there are 13 of them!).

- **Rule 0**
  - A relational DBMS must be able to manage databases entirely through its relational capabilities

- **Rule 1 - The Information Rule**
  - All information in a relational database (including table and column names) is represented explicitly as values in tables

- **Rule 2 - Guaranteed Access**
  - Every value in a relational database is guaranteed to be accessible by using a combination of the table name, primary key value and column name

- **Rule 3 - Systematic Null Value Support**
  - The DBMS provides systematic support for the treatment of null values (unknown or inapplicable data), distinct from default values, and independent of any domain

- **Rule 4 - Active, On-line Relational Catalog**
  - The description of the database and its contents is represented at the logical level as tables, and can therefore be queried using the database language

- **Rule 5 - Comprehensive Data Sublanguage**
  - There must be at least one language supported that has a well-defined syntax and is comprehensive, in that it supports data definition, manipulation, integrity rules, authorization and transactions

- **Rule 6 - View Updating Rule**
  - All views that are theoretically updatable can be updated through the system

- **Rule 7 - Set-level Insertion, Update and Deletion**
  - The DBMS supports not only set-level retrievals, but also set-level inserts, updates and deletes

- **Rule 8 - Physical Data Independence**
  - Application programs and ad-hoc programs are logically unaffected when physical access methods or storage structures are altered

- **Rule 9 - Logical Data Independence**
  - Application programs and ad-hoc programs are logically unaffected, to the extent possible, when changes are made to table structures

- **Rule 10 - Integrity Independence**
  - The database language must be capable of defining integrity rules, they must be stored in the on-line catalog, and they cannot be bypassed

- **Rule 11 - Distribution Independence**

---

6 Alok Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 20-21
- Application programs and ad-hoc requests are logically unaffected when data is first distributed, or when it is redistributed

o Rule 12 - Non-Subversion
- It must not be possible to bypass the integrity rules defined through the database language by using lower level languages

Structured Query Language

SQL, an English-like language, was developed in the late 1970's at IBM San Jose Research Laboratories. SQL is a high-level, set-oriented, nonprocedural database language for relational database systems. High-level means that operations like "joins" (the ability to combine two or more tables into one new table) allow a person to tell a computer to do a whole bunch of work all in one request. Set-oriented means you deal with sets of data at a time. Nonprocedural means you indicate what you want to retrieve not how you want to retrieve it.

A query language expresses queries in a fashion that:

- Allows a computer to find a set of records in a DBMS
- Make changes to these records
- Make the resulting set of records available for further queries

There are two major approaches to the API for SQL:

o Embedded Approach
- Statements required for database access are coded directly in the source code and usually denoted with a special character (like $)
- This process requires a precompilation of the source code by the DBMS and generates a new runtime source code for operation

o Library Approach (Call-level)
- Function calls are placed in the source code
- At compile time, a function is passed, the appropriate code is returned from the library and the new source code is generated

Application versus DBMS-Enforced Integrity

Whichever API method one uses to access SQL, data integrity remains high on the list of priorities. There are two methodologies to handle data integrity:

o Application based
- Advantages:
  -- The code to create and maintain data can be tailored
- Disadvantages:
  -- Applications must devote much code to data integrity logic
  -- All applications must be consistently enforce this integrity
  -- If integrity restraints change, every application is affected

o DBMS based
- Advantages:

---

7 Aloke Nath, *The Guide To SQL Server* [Addison-Wesley, 1990]; pp. 21
8 David Vaskevitch, *Client/Server Strategies* [IDG Books, 1993]; pp. 254
9 David Vaskevitch, *Client/Server Strategies* [IDG Books, 1993]; pp. 255
-- Integrity constraints handled directly through data definition language
-- These constraints are specified in one location instead of across applications
- Disadvantages:
-- A desired override of the constraints during a new database design can affect the applications that have been freed of these responsibilities