

SQL: Structured Query Language

What is it?

SQL is the international standard language used to create and maintain relational databases.

What can you do with it?

- create databases
- add new data to databases
- maintain the data
- retrieve selected parts of the data

Databases

A **database** is a self-describing collection of integrated records.

A **record** is a representation of some physical or conceptual object. Records have multiple attributes.

For example: In a database of customer info, each customer has a record. Name, address, phone number are all attributes of the record. These are the data.

A database is made up of **data** and **metadata**. **Metadata** describes the structure of the data within the database. This is important because you need to know how the data is arranged, so you can retrieve it.

This is why a database is considered *self-describing*; because it contains a description of its own structure.

A database is considered *integrated* because it contains both data items and the relationships among them.

There are different databases with different conceptual models:

- relational
- hierarchical
- network

SQL was originally designed for the relational model.

Database Management Systems

A database management system (DBMS) is a set of programs that are used to define, administer and process databases and their associated applications.

A database is a structure used to *hold* data. A DBMS is used to *build* the structure and *use* the data.

What is a DBMS preferable to flat files?

A flat file has minimal structure. It is a collection of one data record after another in a specified format. Essentially, it's a list. The file doesn't store metadata and there is little overhead (stuff that is not data). Low overhead means it runs quickly. But the application programs have to include application codes in order to manipulate the data.

You run into complications if multiple applications all want to access the same flat file data, because they all have to have the data manipulation code. This is very redundant.

A DBMS handles the details of data manipulation. Therefore, you don't need codes in the applications.

Relational Database Models

SQL applies only to the relational model. The most important attribute that distinguishes relational databases from other models is that changes can be made to the database structure without requiring changes to applications that were based on the old structure.

That means, you can add a column to a table without changing the application that will process the table *unless* you change a column that the application deals with.

With hierarchical or network models, if you add a new attribute to the database, you have to change the application to accommodate it, even if it doesn't use the new attribute. With these models, database structure is hard-coded into the application.

The relational model offers structural flexibility. That means you can retrieve combinations of data that you might not have anticipated needing when the database was designed. They are flexible because their data is found in tables that are independent of each other.

A relational database is called so because it is made up of one or more relations. A **relation** is a two-dimensional array of rows and columns, with single-valued entries and no duplicate rows. (Essentially, it's a table.) Every row contains a record and every column has a single attribute.

A **view** is a subset of a database that an application can process. It may contain parts of one or more tables. It eliminates data that is not relevant to your current needs. It allows you to look at part of the data, but is not itself part of the data. It is useful because it allows you to extract and format data without physically altering the stored data.

A **domain** tells you what values you may store in the column. An attribute of a relation (column of a table) can have a finite number of values. The set of these values is the domain of the attribute.

For example: In thinking of an inventory of cars, the colours for the cars might be red, black, white and gray. These four colours are the domain of the "colour" attribute.

Constraints are rules that determine what values the attributes of a table can assume. Constraints prevent anyone from storing invalid data in the table. When you take the characteristics of a table column, plus the constraints of that column, you have the column's domain. For example, you can constrain the database to only accept those four values in the colour column for cars.

SQL

SQL is non-procedural. To solve a problem using SQL, you just tell it *what* you want, instead of *how* to get it. The DBMS decides the best way to get it.

A **query** is a question you ask the database. If any data satisfies the conditions of the query, SQL retrieves the data.

(Taylor, Allen G. *SQL for Dummies*. 4th ed. New York: IDG Books Worldwide, Inc., 2001.)

SQL Keywords

The following are some keywords SQL uses in **data retrieval**.

“**Select**” is used to retrieve one or more rows from one or more tables in a database.

Here are some of the keywords used in combination with “Select”.

“From” is used to show which table to take the data from and how the tables join each other.

“Where” is used to identify which rows to retrieve

“Group By” is used to combine rows with related values

“Having” is used to identify which of the “combined rows” are to be retrieved

- it works like “Having” but works with “Group By”

“Order By” is used to identify which columns are used to sort the resulting data

Data retrieval is usually combined with data projection. The data often needs to be expressed differently than how it's stored.

Here are two examples of how data retrieval works.

Example 1:

This is what you would type into SQL to get a list of expensive books:

```
SELECT * FROM books
WHERE price > 100.00
ORDER BY title
```

SQL then retrieves the records from the “books” table that have a “price” greater than 100. SQL sorts the results alphabetically by book title. The asterisk (*) means to show all columns of the “books” table.

You can also do this by naming specific columns.

Example 2:

```
SELECT books.title, count(*) AS Authors
FROM books
JOIN book_authors
  ON books.book_number = book_authors.book_number
GROUP BY books.title
```

This would represent a query asking how many authors there are per book. The output would look something like this:

Title	Authors
SQL Examples and Guide	3
The Joy of SQL	1
How to use Wikipedia	2
Pitfalls of SQL	1
How SQL Saved my Dog	1

(<http://en.wikipedia.org/wiki/Sql>)