Databases and SQL

Hanna Damarjian

November 1, 2021

Table of Contents

The pieces of this presentation are organized as follows:

Databases

Definition: a **database** is any collection of related information. Some examples include: a phone book, shopping list, Facebook's User Base.

Databases can be stored in different ways - on paper, in your mind, on a computer, this R Markdown file, comments section, etc,... Think of a database as structured information or data that is stored electronically on a hard drive or other storage device of a computer.

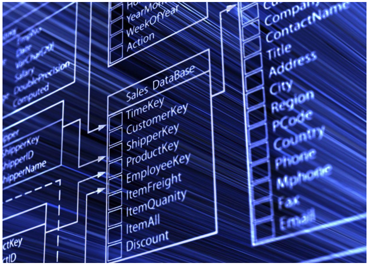


Figure 1: Example of databases

Examples of Databases

Amazon.com:

1. Keeps track of products, reviews, purchase orders, credit cards, users, media, etc.
2. Trillions of pieces of information need to be stored readily available.
3. Information is extremely valuable (security is essential) and critical to Amazon.com's functioning.

Shopping list:

1. Keeps track of consumer products that need to be purchased.
2. Ten-twenty pieces of information need to be stored and readily available.
3. Information is for convenience sake and not necessary for shopping.

One of the best sources to contain, access, and manage your databases is your computer (or a **server**). So, how can we access the "database(s)"?

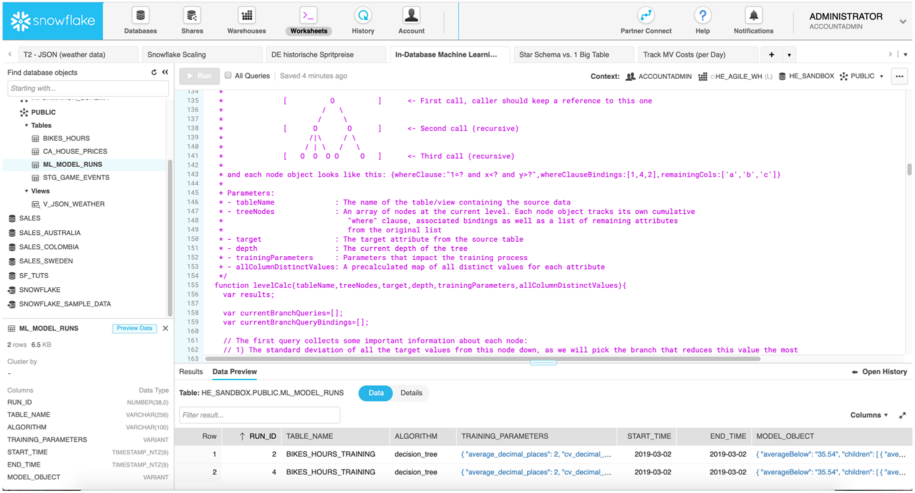
Database Management Systems (or DBMS)

In order to access databases, users must communicate to the **database management system (DBMS)**. The DBMS is a special software program that helps users create and maintain a secure database. It is *not* an actual database.

The DBMS comprises the listed databases a user can concoct. These databases are filled with all sorts of invaluable data including tables, documents (e.g. JSON, XML), graphs, etc. The DBMS can create, read, update, and delete (C.R.U.D.) information on or in the database(s).

An example of a DBMS is **Snowflake**. Organizations such as US Foods and the American Red Cross use Snowflake for a number of its stored data.

DBMS Image



```
 *
 *              [          0          ]      <- First call, caller should keep a reference to this one
 *                       /   \
 *                      /     \
 *              [      0       0      ]      <- Second call (recursive)
 *                    /|\     / \
 *                   / | \   /   \
 *              [   0  0  0 0     0   ]      <- Third call (recursive)
 *
 * and each node object looks like this: {whereClause:"1=? and x<? and y>?",whereClauseBindings:[1,4,2],remainingCols:['a','b','c']}
 *
 * Parameters:
 * - tableName              : The name of the table/view containing the source data
 * - treeNodes              : An array of nodes at the current level. Each node object tracks its own cumulative
 *                              "where" clause, associated bindings as well as a list of remaining attributes
 *                              from the original list
 * - target                 : The target attribute from the source table
 * - depth                  : The current depth of the tree
 * - trainingParameters     : Parameters that impact the training process
 * - allColumnDistinctValues: A precalculated map of all distinct values for each attribute
 */
function levelCalc(tableName,treeNodes,target,depth,trainingParameters,allColumnDistinctValues){
  var results;

  var currentBranchQueries=[];
  var currentBranchQueryBindings=[];

  // The first query collects some important information about each node:
  // 1) The standard deviation of all the target values from this node down, as we will pick the branch that reduces this value the most
```

ML_MODEL_RUNS

2 rows 6.5 KB

Cluster by

-

Columns	Data Type
RUN_ID	NUMBER(38,0)
TABLE_NAME	VARCHAR(256)
ALGORITHM	VARCHAR(100)
TRAINING_PARAMETERS	VARIANT
START_TIME	TIMESTAMP_NTZ(9)
END_TIME	TIMESTAMP_NTZ(9)
MODEL_OBJECT	VARIANT

Table: HE_SANDBOX.PUBLIC.ML_MODEL_RUNS

Row	↑ RUN_ID	TABLE_NAME	ALGORITHM	TRAINING_PARAMETERS	START_TIME	END_TIME	MODEL_OBJECT
1	2	BIKES_HOURS_TRAINING	decision_tree	{ "average_decimal_places": 2, "cv_decimal_...	2019-03-02	2019-03-02	{ "averageBelow": "35.54", "children": [{ "ave...
2	4	BIKES_HOURS_TRAINING	decision_tree	{ "average_decimal_places": 2, "cv_decimal_...	2019-03-02	2019-03-02	{ "averageBelow": "35.54", "children": [{ "ave...

Figure 2: Example of DBMS ∼ Snowflake

Types of Databases

There are two types of databases on the DBMS:

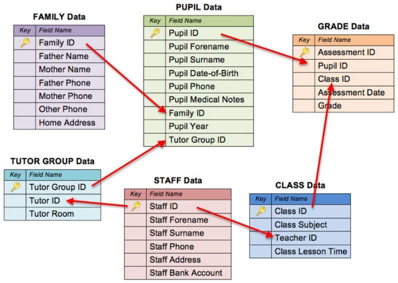
Relational Databases (SQL):

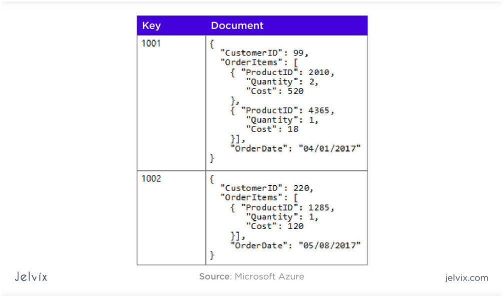
1. Data is organized into one or more tables.
2. Each table has rows and columns (see last slide).
3. A unique-key value identifies each row. This could be an observation unit number or SSN.

Non-relational Databases (non-SQL/not just SQL):

1. Data is organized in anything but a table.
2. Key value stores documents and graphs.

Relational vs. Non-relational Databases





Key	Document
1001	`{ "CustomerID": 99, "OrderItems": [ { "ProductID": 2010, "Quantity": 2, "Cost": 520 }, { "ProductID": 4365, "Quantity": 1, "Cost": 18 }], "OrderDate": "04/01/2017" }`
1002	`{ "CustomerID": 220, "OrderItems": [ { "ProductID": 1285, "Quantity": 1, "Cost": 120 }], "OrderDate": "05/08/2017" }`

Jelvix

Source: Microsoft Azure

jelvix.com

Above are two types of visualizations of stored information. Which is a relational database and which is a non-relational database? Why?

For the rest of this presenation I will be explaining details about relational databases.

Relational Databases (SQL)

Just as every database has its own DBMS, a relational database has its own **relational database management system (RDBMS)**. Some examples include: Snowflake, MySQL, Oracle, postgreSQL, mariaDB.

Again, the goal of every DBMS (relational or non-relational) is to not only store its DBs but also to C.R.U.D. on its stored information.

Question: How can a user be able to access, C.R.U.D., and retrieve the desirable results from that stored information?

Solution: We need to speak the language or *communicate* to the RDBMS. That language is **SQL**.

SQL

Structured Query Language (SQL) is the standard language for interacting with the RDBMS. Elements of SQL include:

1. Used to perform C.R.U.D. operations, as well as other administrative tasks (user management, security, backup, etc).
2. Used to define tables and structures (remember *only* RDBMSs understand SQL).
3. SQL code used on one RDBMS is not always portable to another w/out modification. In other words, the language of SQL varies across the type of RDBMS (or operating system).

SQL is a hybrid language

SQL is four types of languages in one. It is a:

1. **Data Query Language (DQL)** - used to *query* the database for information and get information that is already stored there.
2. **Data Definition Language (DDL)** - used for *defining* database schemas.
3. **Data Control Language (DCL)** - used for *controlling* access to the data in the database; requires user and permissions management.
4. **Data Manipulation Language (DML)** - used for *inserting, updating, and deleting* data from the database.

Relevance to the Industry

According to Dataquest, the percentage of jobs in data that require SQL knowledge is quite drastic. For data science/analyst careers:

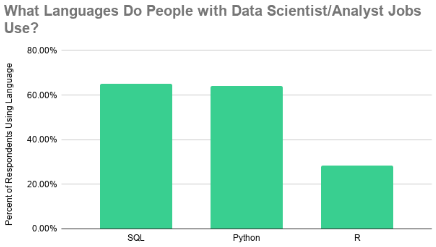


Figure 3: Source: StackOverFlow; $n \geq 10,000$

SQL is the most-used language in data science, according to the 10,000+ data professionals who responded to StackOverflow's 2020 survey.

SQL Queries

A **query** is a set of instructions given to the RDBMS (written in SQL) that tell the RDBMS what information you want it to retrieve for you. The goal is to get the data you need/want to analyze!

I want to share a few SQL queries (using the **iris** dataset on R). The package to write/communicate SQL queries on R is:

```
library(sqldf) # SQL on a dataframe
```

```
## Loading required package: gsubfn

## Loading required package: proto

## Loading required package: RSQLite
```

```
# We will use the function "sqldf()" for queries
```

Query 1

Question: Hey R (or Snowflake, whatever your RDBMS is), what are the first five observations from the iris table?

```
sqldf("SELECT *
       FROM iris
       LIMIT 5")
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

SELECT returns the data (column(s) and corresponding rows), **"*"** means *all columns*, **FROM** is the table you are retrieving the data from, and **LIMIT** specifies the first x number of rows to select.

Query 2

Question: Hey R, are there any iris flowers (species) that have a sepal length less than 4.5 cm?

```
sqldf('SELECT Species,"Sepal.Length"
       FROM iris
       WHERE "Sepal.Length" < 4.5')
```

```
##   Species Sepal.Length
## 1  setosa          4.4
## 2  setosa          4.3
## 3  setosa          4.4
## 4  setosa          4.4
```

WHERE is a *conditional clause* (or command). In other words, it takes a subset of the rows (under a column(s)) that meet a condition.

Query 3

Question: Hey R, what are the flower types (species) that have an average petal width greater than 1 centimeter?

```
sqldf('SELECT Species, AVG("Petal.Width")
       FROM iris
       WHERE "Petal.Width" > 1
       GROUP BY Species')
```

```
##      Species AVG("Petal.Width")
## 1 versicolor            1.37907
## 2  virginica            2.02600
```

This is an example of **data aggregation**. In other words, we apply a function (in this case **"AVG()"**) on at least one row of our data column(s) (here it is **"Petal.Width"**) to deduce a question or problem.

References:

Sources:

Databases/SQL: https://www.youtube.com/watch?v=HXV3zeQKqGY

Using SQL with R as RDBMS:
https://dept.stat.lsa.umich.edu/~jerrick/courses/stat701/notes/sql.html#where

Industry:
https://www.dataquest.io/blog/why-sql-is-the-most-important-language-to-learn/

Images:

https://wallpapercave.com/wp/wp2347580.jpg

https://data-science-blog.com/wp-content/uploads/2019/07/snowflake-sql-editor.png

https://www.igcseict.info/theory/7_2/school/files/stacks_image_6771.png

https://jelvix.com/wp-content/uploads/2020/05/whats-a-nonrelational-database.jpg

Wrap-Up

Thank you! Questions/Comments?