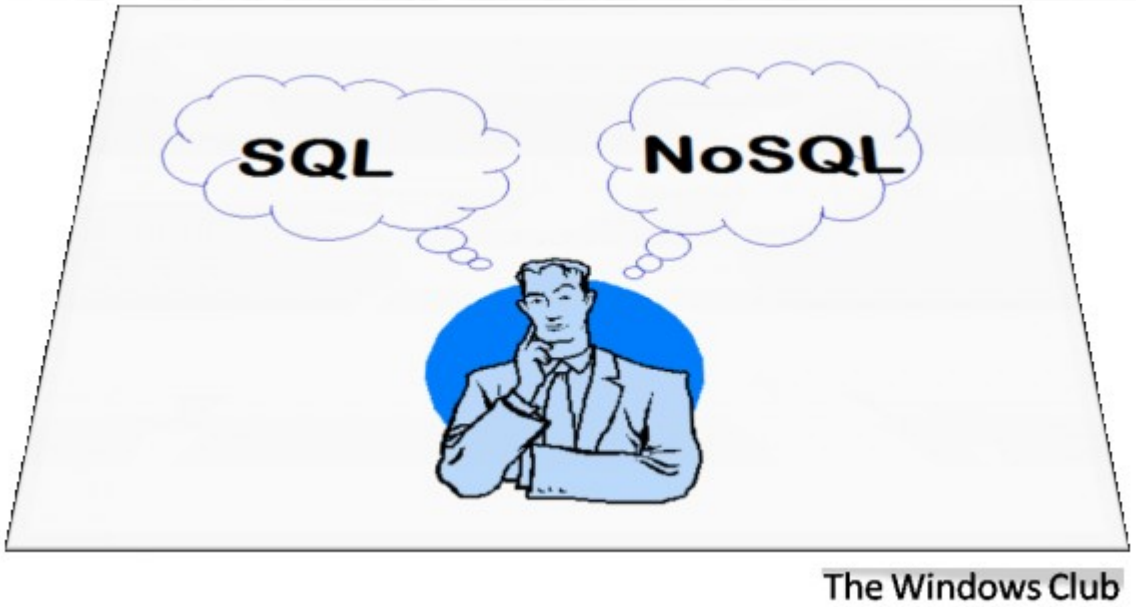
SQL vs NoSQL



By: Mohammed-Ali Khan
Email: m23khan@cse.yorku.ca
Date: October 21, 2012

YORK UNIVERSITY

Agenda

- History of DBMS – why relational is most popular?
- Types of DBMS – brief overview
- Main characteristics of RDBMS
- Main characteristics of NoSQL DBMS
- SQL vs NoSQL
- DB overview – Cassandra
- Criticism of NoSQL
- Some Predictions / Conclusions

History of DBMS – why relational is most popular?

- 19th century: US Government needed reports from large datasets to conduct nation-wide census.
- 1890: Herman Hollerith creates the first automatic processing equipment which allowed American census to be conducted.
- 1911: IBM founded.
- 1960s: Efforts to standardize the database technology
  - US DoD: Conference on Data Systems Language (Codasyl).

History of DBMS – why relational is most popular?

- 1968: IBM introduced IMS DBMS (a hierarchical DBMS).
- IBM employee Edgar Codd not satisfied, quoted as saying:
  - "Taking the old line view, that the burden of finding information should be placed on users..."
- Edgar Codd publishes a paper "A relational Model of Data for large shared Data Banks"
  - Key points: Independence of data from hardware/storage implementation
  - High level non-procedural language for data access
  - Pointers/keys (primary/secondary)

History of DBMS – why relational is most popular?

- 1970s: Two database projects launched based on relational model
  - Ingres (Department of Defence Initiative)
  - System R (IBM)

- Ingres had its own query language called QUEL whereas System R used SQL.

- Larry Ellison, who worked at IBM, read publications of the System R group and eventually founded ORACLE.

History of DBMS – why relational is most popular?

- 1980: IBM introduced SQL for the mainframe market.

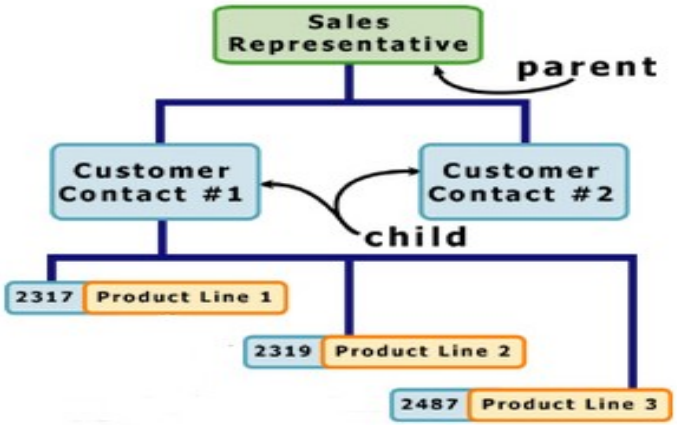
- In a nutshell, American Government's requirements had a strong role in development of the relational DBMS.

deredes.net

jobawareness.com

Types of DBMS – brief overview

- Relational:
  - Users see data as tables and nothing but tables.
  - Each record (tuple) forms a row in the table.
- Hierarchical:
  - Data is organized in form of pyramid.
  - Popular examples: IBM IMS, Windows Registry



source: http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_06.html

Types of DBMS – brief overview

- Network
  - Similar to hierarchical however child nodes can have links to other child nodes.
  - Examples: IDBMS, Raima Database Manager
- Object-oriented Databases (OODBMS)
  - Can handle may data types: graphics, videos, etc.
  - Examples: ObjectDB, Gemstone
- Other types:
  - Document stores, Key/value stores, Graph

Main characteristics of RDBMS

- The data model presents data in form of a table.
- The schema for a given relation specifies its name, the name of the field and the type of each field.

sid	*name*	*login*	*age*	*gpa*
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

source: Ramakrishnan and Gehrke, Database Management Systems (2nd Ed).

Main characteristics of RDBMS

- Relational model consists of the following 3 aspects:
  - Structural
  - Integrity
  - Manipulative
- Three important operations available:
  - Restrict
  - Project
  - Join

Main characteristics of RDBMS

- In RDBMS, the restriction that user must see data in form of table only applies to the part where user sees data.
- Majority of database research over the last 30 years or so has been based on the relational model.
- Popular RDBMS today: IBM DB2, Oracle, Microsoft SQL Server, MySQL.

Main characteristics of NoSQL DBMS

- NoSQL stands for “not only SQL”. In broader sense, it includes all non-relational DBMS (which may or may not use a querying language).
- In this presentation, we will focus on key-value data stores commonly used by Web 2.0 applications.
- As opposed to transactions in RDBMS conforming to ACID, NoSQL DBMS follow the CAP theorem and thus their transactions conform to the BASE principle.

Main characteristics of NoSQL DBMS

- ACID, CAP, BASE – what do they mean???

- ACID: type of transaction processing done by RDBMS
  - (A)tomcity: If a part of operation fails, whole operation fails.
  - (C)onsistency: Information is always consistent, avoid read/write errors at all costs.
  - (I)solation: Multiple transactions at the same time do not impact each other.

    (D)urability: Information has to be stored into DB, not queued in memory.

Main characteristics of NoSQL DBMS

- ACID model ensures the database reliability.

  BUT...

- There are many cases where such models cannot be applied!

- This has given rise to NoSQL DBMS recently which follow the CAP theorem and conform to the BASE principle.

Main characteristics of NoSQL DBMS

- CAP Theorem: Also called Brewer's theorem (after Eric Brewer)
  - (C)onsistency: Whenever data is written, everyone who reads the DB will see the latest version.
  - (A)vailability: We can always expect that each operation terminates in an intended response.
  - (P)artition tolerance: DB can still be read from/written to when parts of it are offline. Afterwards, when offline nodes are back online, they are updated accordingly.

- IMPORTANT: CAP theorem also states that for any system sharing data, it is impossible to guarantee all 3 properties.

Main characteristics of NoSQL DBMS

- Therefore possible combinations are CP, AP, and CA (hard to combine).
- Based on CAP theorem, availability cannot always be guaranteed without having to relax consistency.
- DBMS based on CAP, instead of having their transactions conform to ACID, conform to BASE properties:
  - (B)asically (A)vailable
  - (S)oft State
  - (E)ventually Consistent

Main characteristics of NoSQL DBMS

Database Scalability:

- Top level websites (e.g. Facebook) are noted for their massive scalability, low latency, the ability to grow the capacity of their DB on demand and an easier programming model.

- RDBMS (traditionally) reside on one server, which can then be scaled by adding more processors, more memory, and external storage. RDBMS residing on multiple servers usually use replications to keep database synchronized (ACID).

Main characteristics of NoSQL DBMS

- Popular RDBMS like Oracle can be scaled up but it is difficult.
- In the last decade, a new family of scalable NoSQL DBMS have been developed. These systems scale nearly linearly with the number of servers used.
- These NoSQL databases heavily rely on using hash tables to do fast lookup in a distributed setup.
- Horizontal data distribution enables to divide computation into concurrently processed tasks. To minimize complexity for operations such as data aggregation, specialized algorithms are used (MapReduce).

Main characteristics of NoSQL DBMS

- NoSQL databases also do not support operations JOIN and ORDER BY. Despite these restrictions, NoSQL databases enable to develop useful applications.

- Types of data models:
  - Document stores: data stored similar to JSON format.
  - Key-value stores: Contain a set of couples (key, value).

Main characteristics of NoSQL DBMS

QUERYING:

- Querying is the least highlighted part of a NoSQL DBMS.
- One possibility of querying a NoSQL DBMS is using a restricted SQL dialect such as in simpleDB:

```
SELECT output_list
FROM domain_name
[WHERE expression] [sorting] [LIMIT limit]
```

- output_list can be *, **itemName, count()**, list_of_attributes, domain_name determines name of domain from which data to search, expression can use =,<=,>,>=, LIKE, NOT LIKE, BETWEEN, IS NULL, etc. Limit restricts output size. Join, aggregation, and subquery embedding are not supported.

Main characteristics of NoSQL DBMS

- A broader subset of SQL called GQL (Google Query Language) is used in Google's AppEngine database.
- A typical API for NoSQL databases contains operations like get(key), put(key,value), delete(key), execute(key,operations,parameters).
- For document stores such as CouchDB, since the data is held in JSON like format, queries can be more elaborate and custom.

Main characteristics of NoSQL DBMS

Data Storing

- Some NoSQL Databases store their data in memory and store it in disk after closing the work with a database or for backups (e.g. Redis DB).

- NoSQL databases can reside on a single server but are meant to work in cloud of servers. They can be equipped also by distributed indexes.

Main characteristics of NoSQL DBMS

The Map Reduce Algorithm

- MapReduce is a programming model for managing large amounts of data. It was popularized by Google in 2004.

- The main interest of this model is that the 2 primitive Map and Reduce are easily parallelizable and can perform on large sets of data.

- It is perfectly adapted to large scale distributed applications and large data processing.

Main characteristics of NoSQL DBMS

- The Map primitive consists of processing a data list in order to create key/value pairs. Then, the Reduce primitive will process each pair in order to create new aggregated key/value pairs.

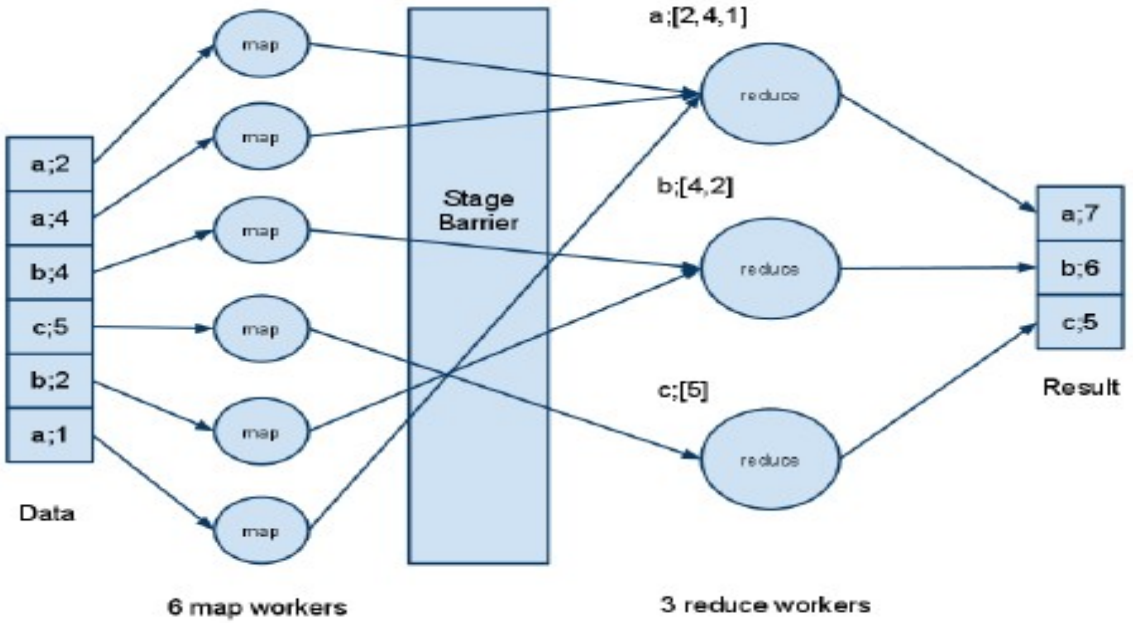
$$map(k1, v1) = list(k2, v2) \quad (1)$$

$$reduce(k2, list(v2)) = list(v3) \quad (2)$$

$$List : (a;2)(a;4)(b;4)(c;5)(b;2)(a;1) \quad (3)$$

$$After\ mapping : (a;[2,4,1]), (b;[4,2]), (c[5]) \quad (4)$$

$$After\ reducing : (a;7), (b;6), (c;5) \quad (5)$$



source: Laurent et al. "Reduce, You Say: What NoSQL Can do..."

SQL vs NoSQL

source: http://www.forocoches.com

SQL vs NoSQL

- Developers who encourage NoSQL DBMS usually prefer:
  - Document-style stores
  - Key-value stores
- There are usually two possible reasons for moving away from a traditional relational DBMS to NoSQL:
  - The performance argument
  - The flexibility argument

SQL vs NoSQL

- The performance argument: A development team starts using MySQL for their data storage needs and overtime finds performance to be inadequate. Their options are:
  - “Shard” the data to partition it across several sites.
  - Abandon MySQL and pay big licensing fees for an enterprise level RDBMS.
- The flexibility argument: A team finds their data does not conform to a rigid relational schema. Hence, they can't be bound by the structure

SQL vs NoSQL

- If we consider only those type of workloads which the NoSQL are most famous for- update and look-up intensive online transaction processing (OLTP) - we come across two ways to improve the OLTP performance:
  - Provide automatic sharding over a shared-nothing processing environment.
  - Improve per-server OLTP performance.

- Well-designed SQL DBMS such as Greenplum, Aster Data, Vertica, etc. Provide shared nothing scalability.

- The overhead associated with OLTP in traditional SQL databases has little to do with SQL.

SQL vs NoSQL

- The major overhead occuring in relational DBMS is due to:
  - Logging: Traditional DBMS write everything twice – once to the DB and once to the log (and log must be forced to the disk).
  - Locking: Before touching a record, a transaction must first set a lock on it.
  - Latching: Updates to shared data structures (e.g. B-trees), the lock table, and the resource tables must be done carefully in a multi-threaded environment.
  - Buffer Management: Data in traditional DBMS is stored on fixed-size disk pages. A buffer pool manages which set of disk pages is cached in memory at any given time.

SQL vs NoSQL

In NoSQL...

- Many systems are disk-based and retain a buffer pool as well as a multi-threaded architecture. This leaves only 2 sources of additional overhead in RDBMS.
- As compared to NoSQL, RDBMS provide ACID compliance and therefore offer ACID based transactions. This is something NoSQL DBMS have sacrified for performance.

SQL vs NoSQL

- Popular NoSQL DBMS: Redis, Cassandra, MongoDB, CouchDB, BigTable, etc.

source: Google Images

DB overview – Cassandra

- Programmed in Java, Apache Cassandara was introduced in 2008 as an open source NoSQL DBMS.
- Cassandra was developed at Facebook and was first used by them to develop their NoSQL based IT infrastructure.

socialtimes.com

marketingthateasy.com

DB overview – Cassandra

- Cassandra is a distributed NoSQL DBMS designed for managing very large amounts of data spread over countless number of servers while providing highly available service without any single point of failure.
- Unlike many other NoSQL DBMS, Cassandra comes with 'tunable' consistency.
- The main features of Cassandra are:
  - Being symmetric.
  - Consistent Hashing (Distributed Hash tables).
  - Flexible Partitioning and High Availability.
  - The Client Interface.
  - Data Model.
  - Consistency.

DB overview – Cassandra

- Symmetric: Cassandra is designed for a distributed environment (although it can run on a single server). All nodes in cluster are identical in terms of software and there are no manager/coordinator nodes. This ensure high availability (no single point of failure).

- Consistent Hashing: A scheme that provides hash table functionality in a way that the addition or removal of one slot does not significantly change the mapping of keys to slots (in traditional cases, such an operation would lead to nearly all keys being remapped).

- Flexible Parititioning/High Availability: Both placement of data and placement of data's replica is highly flexible (in terms of which node(s)). Growing a Cassandra cluster requires no downtime.

- The Client Interface: Cassandra uses the Thrift framework to provide a cross-language client interface. To access the database via APIs, developers use library of APIs provided in languages such as Java.

DB overview – Cassandra

- Data Model: Following is a Cassandra's example of a JSON representation of a key -> column families -> column structure:

DB overview – Cassandra

```
{
 "mccv":{
     "Users":{
         "email":{
          "name":"email",
             "value":"foo@bar.com"
         },
         "webSite":{
          "name":"webSite",
          "value":"http://bar.com"
         }
     },
     "Stats":{
        "visits":{
         "name":"visits",
         "value":"243"
        }
    }
},
"user2":{
    "Users":{
        "email":{
         "name":"email",
         "value":"user2@bar.com"
        },
        "twitter":{
         "name":"twitter",
         "value":"user2"
        }
    }
}
```

source: Okman et al. "Security Issues in NoSQL Databases".

DB overview – Cassandra

- Consistency: Like other NoSQL DBMS of it's category, Cassandra too is an eventually consistent data store. Cassandra has a feature called 'quorum'. Quorum level guarantees that half of the records are updated before the action returns.
  - Another option is write-all-read-one model in which case every read will be consistent.

Criticism of NoSQL

- Due to performance being the top priority, NoSQL databases tend to have more security gaps than traditional SQL databases.

- In Cassandra, some of the security holds are found in:
  - Data files
  - Client Interfaces
  - Cassandra Query Language (CQL)

Criticism of NoSQL

- Cassandra data files: The data in Cassandra is kept unencrypted and Cassandra doesn't provide a way to encrypt it. It is up to the Developer to encrypt/decrypt the data themselves.

- Cassandra Client Interface: As mentioned, Cassandra uses the Apache Thrift framework for client communications. Thrift provides for SSL transport however Cassandra is not using this feature. This means, communication between a client and the DBMS is in clear-text. Worse, this means whenever client sends a login ID and password, they are transmitted in clear text.

- Cassandra Query Language (CQL): Is susceptible to injection attacks, similar to SQL injection attacks. Therefore, it is up to the Developers to analyze the incoming queries from clients to weed out harmful queries.

Criticism of NoSQL

- Following is a partial list of causes of errors in databases:
    1. Application errors
    2. Repeatable DBMS errors
    3. Unrepeatable DBMS errors
    4. Operating system errors
    5. Hardware failure in local cluster
    6. Network partition in a local cluster
    7. A disaster
    8. Network failure in the WAN connecting the clusters together.

Criticism of NoSQL

- Errors 1 and 2 will cause problems in any high availability DBMS. In these two scenarios, there is no way to keep going (availability becomes non-existent). This indicates that current state of DBMS is wrong.

- Error 7 will only be recoverable if a local transaction is only committed after the assurance that the transaction has been received by another WAN-connected cluster. Few application builders are willing to accept this kind of latency.

- This means that errors 1,2, and 7 are examples of cases where the CAP theorem simply does not apply. A durable DBMS must be designed to handle such scenarios.

- In conclusion, we can't disregard 'consistency' (C) in designing a DBMS because as detailed above, there are situations where CAP theorem fails to hold.

Some Predictions / Conclusions

Dr. Rick Cattell in his 2010 paper titled “Scalable SQL and NoSQL Data Stores” makes the following predictions about NoSQL DBMS:

- Many software developers will be willing to switch from using relational databases to NoSQL databases due to scalability, availability, and other advantages.
- NoSQL data stores are not a 'fad' – they are here to stay.
- New relational Databases will also take a significant share of the scalable data storage market.
- Many of the scalable NoSQL data stores will not be deemed as being fit for being used in an “enterprise” setting.
- Different NoSQL products will consolidate and one or two NoSQL databases will emerge as key players in the database industry.

Some Predictions / Conclusions

Personal Conclusions:

- Both relational databases and NoSQL databases have their strong points and weak points (ACID vs CAP, consistency, performance, etc.).
- A large percentage of software developers don't have academic background in Computer Sciences or Information Technology. For vast majority of them, the least complicated solution is always the best solution. Therefore, such developers will naturally gravitate towards NoSQL databases.
- As time goes on, there will either be an increased overlap between types of features/services offered by SQL and NoSQL databases or they both will settle into specified roles and norms.
- Regardless of which database technology (SQL or NoSQL) becomes dominant, there will always be demand for all types of databases in future.
- Database sciences is currently going through a new golden age. It is imperative that the community of Database scientists don't neglect either relational DBMS or NoSQL DBMS in pursuit of short-term market demands.
- Expect increasing number of applications utilizing both SQL and NoSQL databases to accomplish different things in the same application (e.g. serving vs reporting).

Bibliography

- Jaroslav Pokorny. 2011. NoSQL databases: a step to database scalability in web environment. In Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services (iiWAS '11). ACM, New York, NY, USA, 278-283. DOI=10.1145/2095536.2095583 http://doi.acm.org.ezproxy.library.yorku.ca/10.1145/2095536.2095583
- Rick Cattell. 2011. Scalable SQL and NoSQL data stores. SIGMOD Rec. 39, 4 (May 2011), 12-27. DOI=10.1145/1978915.1978919 http://doi.acm.org.ezproxy.library.yorku.ca/10.1145/1978915.1978919
- Michael Stonebraker. 2010. SQL databases v. NoSQL databases. Commun. ACM 53, 4 (April 2010), 10-11. DOI=10.1145/1721654.1721659 http://doi.acm.org.ezproxy.library.yorku.ca/10.1145/1721654.1721659
- Michael Stonebraker. 2010. In search of database consistency. Commun. ACM 53, 10 (October 2010), 8-9. DOI=10.1145/1831407.1831411 http://doi.acm.org.ezproxy.library.yorku.ca/10.1145/1831407.1831411
- Okman, L.; Gal-Oz, N.; Gonen, Y.; Gudes, E.; Abramov, J.; , "Security Issues in NoSQL Databases," Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on , vol., no., pp.541-547, 16-18 Nov. 2011. DOI: 10.1109/TrustCom.2011.70 URL: http://ieeexplore.ieee.org.ezproxy.library.yorku.ca/stamp/stamp.jsp?tp=&arnumber=6120863&isnumber=6120777

Bibliography

- Bonnet, L.; Laurent, A.; Sala, M.; Laurent, B.; Sicard, N.; , "Reduce, You Say: What NoSQL Can Do for Data Aggregation and BI in Large Repositories," Database and Expert Systems Applications (DEXA), 2011 22nd International Workshop on , vol., no., pp.483-488, Aug. 29 2011-Sept. 2 2011. DOI: 10.1109/DEXA.2011.71 URL: http://ieeexplore.ieee.org.ezproxy.library.yorku.ca/stamp/stamp.jsp?tp=&arnumber=6059864&isnumber=6059784
- Date, C.J. *An Introduction to Database Systems*. 8th ed. Pearson Education/Addison Wesley, 2004. Print.
- Ramakrishnan, Raghu, Gehrke, Johannes. *Database Management Systems*. 2nd ed. New York: McGraw-Hill, 1999.
- College of Information Sciences and Technology. *Topic: Database Fundamentals*. The Pennsylvania State University, 2008. Web. 18 Oct. 2012. <http://www.personal.psu.edu/glh10/ist110/topic/topic07/topic07_06.html>.

THE END

source: Google Images