NoSQL Database Systems

NoSQL database system ? What is it ?

NoSQL database systems were developed in early 2000 in response to demands for processing of vast amounts of data produced by increasing Internet usage and mobile geo-location technologies

Traditional solutions were either too expensive, not scalable, or required too much time to process data

Modern NoSQL database systems borrowed some solutions from the earlier systems and made significant advances in scalability and efficient processing of diverse types of data such as text audio, video, image, and geo-location

NoSQL database systems include four implementation types: key-value stores, document stores, graph stores, column stores

What is in the name ?

A term "NoSQL" has been used for the first time in late 1990 by C. Strozzi as a name of open-source relational database system that did not use SQL as a query language

The usage of "NoSQL" as it is recognized today has been used by J. Oskarsson in 2009 for the projects experimenting with alternate data storage, like BigTable (Google) and Dynamo (Amazon)

"NoSQL" database systems do not use SQL, run on the clusters of computers and provide different options for consistency and distribution

Despite its confrontational nature some people say that "NoSQL" means "Not Only SQL", but then it should be written as "NOSQL" and … it is not !

Why No SQL database systems ?

Impedance mismatch problem: the difference between persistent data structures (relational model) and the transient data structures (object-oriented programming language)

Application and integration databases: a structure that integrates many applications is more complex than single application requires, e.g. index required by one application may cause performance problems for another application

Clusters of small machines: relational databases that operate on shared disk subsystems are not designed to work on clusters

Distribution and consistency: **ACID** transaction protocol is too strict for distributed transactions

ACID properties of database transactions

Atomicity requires that each transaction executes all of its operations or none of them; if a transaction fails then entire transaction must be rolled back

Consistency means that each transaction brings a database from one consistent state to another; it never happens that a transaction terminates and it leaves a database in inconsistent state

Isolation means that concurrent execution of transactions results in a database state that can be obtained from any serial execution; transactions do not communicate with each other

Durability means that once a transaction has been committed all its results become permanent

Properties of NoSQL database systems

Semistructured, schemaless data model

Materialized views

Data aggregates

Specialized distribution models

Weak consistency

Versioning

Support for map-reduce distributed applications

Distribution

Single server: means no distribution at all

Sharding: to support horizontal scalability we put different parts of data onto different servers (shards)

Mater-slave replication: data are replicated across multiple nodes and one node is designated as master node, the others are slaves; all updates are made to the master and later on propagated to slaves

Peer-to-peer replication: data replicated across multiple nodes; all replicas have the same weight, no master mode; nodes communicate their writes; all nodes read and write all data

Combining sharding and replication: use both master-slave replication and sharding

Consistency

A typical read consistency principle where update is performed over two or more data items blocks access o all data items affected by the update, e.g. 2PL protocol

NoSQL database systems relax the transactional consistency to some extent

Data items can be left inconsistent over certain period of time called as inconsistency window

A concept of eventual consistency is used to enforce replication consistency over distributed and replicated data items

Eventual consistency means that the copies of data items can be inconsistent in inconsistency window and all copies will have the same value later on

Consistency

It is possible tolerate long inconsistency window under a condition that read-your-write consistency is enforced

This leads to a concept of session consistency, which means that within a user session read-your-write consistency is enforced.

To provide session consistency it is possible to implement sticky session, i.e. a session that is attached to only one node in a cluster (it is also called as session affinity)

Another solution to provide session consistency is to use version timestamps where every interaction with a data item is performed on a data item with the highest timestamp

Relaxing consistency

It is always possible to design a system that avoids inconsistencies but sometimes we have to trade consistency for some other characteristics of a system

Different domains have different tolerances for inconsistencies and we have to take this tolerance into account as make our decisions

Even in traditional relational database systems it is possible to relax consistency from the highest isolation level (serializable) to the lowest levels to get better performance

CAP theorem: given three properties of Consistency, Availability, and Partition tolerance, you can get only two

Relaxing consistency

Consistency: A state of a database satisfies the given consistency constraints at any moment in time

Availability: If you can talk to a node in a cluster then it can read and write data

Partition: Cluster can survive communication breakages that separate the cluster into multiple partitions unable to communicate with each other (split brain)

A single server system is CA because it has Consistency and Availability and not Partition tolerance

If clusters must be tolerant of network partitions, then we have to trade consistency for availability

Relaxing durability

If SQL systems follow ACID properties then NoSQL systems follow BASE properties: Basically Available, Soft state, Eventually consistent

Relaxing durability means that we trade of durability for higher performance, e.g. apply updates to in-memory representation of a database and periodically flush changes to disk

Another class of durability tradeoffs comes with replicated data, e.g. when a node processes and update but fails before that updates is replicated to other nodes

Types of NoSQL systems

Key-value databases

Document databases

Column-family databases

Graph databases

Key-value databases

Key-value database is the simplest NoSQL database

A client can get value of a key, put value for a key, or delete a key from database

A value is a blob stored in a database

Key-value database systems implement eventual consistency when data items are replicated at many nodes

Key-value database systems have different specifications of transactions; generally there is no guarantees on writes

Query subsystems implement only query by key

Key-value database use sharding for horizontal scaling

The most popular systems include Riak, Redis, Memcached DB, Berkeley DB (Oracle NoSQL), HamsterDB, DynamoDB

Document databases

A document database stores and retrieves documents, which can be in XML, JSON, BSON, etc format

Documents are self-describing, hierchical data structures

The structures of documents are similar to each other but not exactly the same

Documents are stored as the value part of key-value store

```
{
 "firstname": "James",
 "likes":     ["biking", "hiking", "viking"],
 "city":      "Boston",
 "friend":             ,
 "addresses":[{"state": "NSW", "city": "Sydney"},
              {"state": "Vic", "city": "Melbourne"} ]
}
```

Document databases

Consistency is supported through replica sets and choosing to wait for the writes to be replicated to all slaves or a given number of slaves

Transactions are generally not available in NoSQL databases - a write either succeed or fails

Transactions at a single document level are called as atomic transactions

Transactions involving more than one operation are not possible

Document databases provide different query features still not as advanced as SQL

The most popular systems include MongoDB, CouchDB Oracle NoSQL

Column family database

Data stored with keys mapped into values and values grouped into multiple column families, each column being a map of data

Column families are groups of related data that is often accessed together, e.g. for a customer we would often access profile information at the same time and not their orders

The basic unit of storage is a column that consists of name-value pairs where name can be used as a key

```
{
 "name":"firstName",
 "value": "James",
 "timestamp": 123456789
}
```

Column family database

A row is a collection of columns attached or linked to key

A collection of similar rows makes a column family

```
// column family
{
 // row
 "James": {"firstName":"James",
           "lastName": "Bond",
           "lastVisit": "01/01/2014"}
 //row
 "Harry": {"firstName":"Harry",
           "lastName": "Potter",
           "location: "Boston"}
}
```

Column family database

When a column consist of a map of columns then it is called as super column

A supercolumn is a container of columns

```
// super column
{
 "name": "book:9564-78903",
 "value": {
          "author":"Robin Hood"
          "title": "Gambling for dummies"
          "isbn": "9564-78903"}
          }
}
```

When using super columns to create a column family we get super column family

Column family database

Write operation first records data in a commit log and later on in in-memory structure called memtable; Write operation is successful if data item is written into both commit log and memtable

If consistency is set to ONE then read operation is performed on the first replica; if data is not up to date then subsequent reads will get latest data (read repair)

If consistency is set to QUORUM then the data item with the newest timestamp is read; replicas that do not have the newest data are repaired

If consistency is set to ALL then all nodes must respond with the same value

Column family databases do not have transactions

Column family database

Basic queries include GET, SET, and DEL commands

Some systems implement SQL-like query language

Scaling a cluster is performed by adding more nodes

The most popular systems include Cassandra, Hbase, Hypertable, SimpleDB

Graph databases

Graph databases store entities and relationships between the entities

Nodes are instances of objects in the applications

Edges have properties and have directional significance

```
Node james = graphDb.createNode();
james.setProperty("name", "James");
Node harry = graphDb.createNode();
harry.setProperty("name", "Harry");
james.createRelationahip(harry, FRIEND);
harry.createRelationahip(james, FRIEND);
```

Graph databases ensure consistency through ACID-compliant transactions

Graph databases use domain specific languages for traversing graph structures

Graph databases

Properties of nodes can be indexed

A commonly used technique for horizontal scaling is sharding

Any link-rich domain is well suited for graph databases

The most popular systems include Neo4J, Infinite Graph, OrientDB, FlockDB

References

Tivari S., Fowler M., *Professional NoSQL,* John Wiley &Sons Inc., 2011

Alam M., *Oracle NoSQL Database,* Oracle Press, 2014

Sadalge P. J., Fowler M., *NoSQL Distilled,* Addison Wesley, 2013

`http://www.uow.edu.au/~jrg/235/HOMEWORK/`
11 How to process distributed database systems ?