Advanced Databases

NoSQL databases
General concepts

Giulia Vilone

giulia.vilone@tudublin.ie

SQL first

SQL characteristics

- Data stored in columns and tables
- Relationships represented by data
- Data Manipulation Language
- Data definition Language
- Transactions (ACID)
- Abstraction from physical layer

Data Manipulation Language (DML)

- Data manipulated with SELECT, INSERT, UPDATE, & DELETE statements

  SELECT T1.Column1, T2.Column2 …
  FROM Table1, Table2 …
  WHERE T1.Column1 = T2.Column1 …

- Data aggregation
- Functions and procedures
- Explicit transaction control

Data Definition Language

- Schema defined at the start
- Create Table (Column1 Datatype1, Column2 Datatype 2, …)
- Constraints to define and enforce relationships
  - Primary key
  - Foreign key
  - Etc...
- Triggers to respond to INSERT, UPDATE, & DELETE
- Stored Modules
- Alter …
- Drop …
- Security and access control

NoSQL... Why?

History of the world – Part 1

- Web-based applications caused spikes
  - Especially true for public-facing e-Commerce sites
- Internet, distributed data, big data…
- Developers begin to front RDBMS with memcache or integrate other caching mechanisms within the application

Scaling up

- Issues with scaling up when the dataset is just too big
- RDBMS were not designed to be distributed
- Began to look at multi-node database solutions
- Known as ‘scaling out’ or ‘horizontal scaling’
- Different approaches include:
    - Master-slave
    - Sharding

Scaling RDBMS – Master/Slave

Master-Slave

- All writes are written to the master. All reads performed against the replicated slave databases
- Critical reads may be incorrect as writes may not have been propagated down
- Large data sets can pose problems as master needs to duplicate data to slaves

Scaling RDBMS – Sharding

Partition or sharding

- Scales well for both reads and writes

- Not transparent, application needs to be partition-aware
- Can no longer have relationships/joins across partitions
- Loss of referential integrity across shards

Other ways to scale RDBMS

- No JOINs, thereby reducing query time
- This involves de-normalizing data
- In-memory databases

NoSQL started with the aim to address the scaling problem

NoSQL definition

Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontal scalable**. The original intention has been **modern web-scale databases**. The movement began early 2009 and is growing rapidly. Often more characteristics apply as: **schema-free, easy replication support, simple API, eventually consistent / BASE** (not ACID), a **huge data amount**, and more.

What is NoSQL?

- Stands for **N**ot **O**nly **SQL**
- Class of non-relational data storage systems
- Class of data management systems inherently
  - Non-relational
  - Distributed
  - Horizontally scalable
  - With optional schemas
  - Providing simple APIs
- All NoSQL offerings relax one or more of the ACID properties

NoSQL distinguishing characteristics

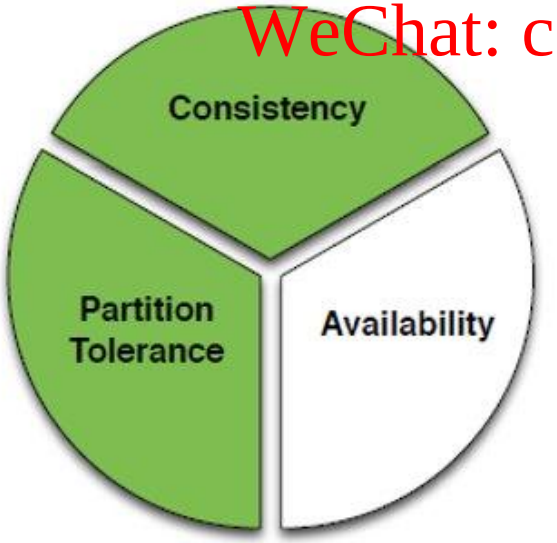
- Large data volumes
  - “big data”
- Scalable replication and distribution
  - Potentially thousands of machines
  - Potentially distributed around the world
- Queries need to return answers quickly
- Mostly query, few updates
- Asynchronous Inserts & Updates
- Schema-less
- ACID transaction properties are not needed – BASE
- CAP Theorem
- Open-source development

CAP theorem

CAP theory

CAP THEORY

C: Consistency: Linearisability
A: Availability: Timely response
P: Partition-Tolerance: Functions in the face of a partition



2 out of 3!

CAP theorem

- **Consistency** – All nodes should see the same data at the same time
- **Availability** – Node failures do not prevent survivors continuing to operate
- **Partition-Tolerance** – the system continues to operate despite arbitrary message loss. *No set of failures less than total network failure is allowed to cause the system to respond incorrectly*

It is impossible for a distributed systems to provide all the above features (E. Brewer, 2000)

2 out of 3: BASE vs ACID

- C+A
  - Always available and consistent
  - Single site databases
  - Cluster Databases (why?)
- C+P
  - Distributed Databases
  - Minority Locking
  - System is unavailable for a while..
- A+P
  - DNS
- BASE -

ACID properties apply to all transactions

Basic (ACID) properties of a transaction are:

- **Atomicity** - 'All or nothing' property.
- **Consistency** - Must transform database from one consistent state to another.
- **Isolation** - Partial effects of incomplete transactions should not be visible to other transactions.
- **Durability** - Effects of a committed transaction are permanent and must not be lost because of later failure.

BASE transactions

- Acronym contrived to be the opposite of ACID
  - **B**asically **A**vailable,
  - **S**oft state,
  - **E**ventually Consistent
- Characteristics
  - Weak consistency – stale data OK
  - Availability first
  - Best effort
  - Approximate answers OK
  - Simpler and faster

BASE

- When no updates occur for a long period of time, eventually all updates will propagate through the system and all the nodes will be consistent.
- For a given accepted update and a given node, eventually either the update reaches the node, or the node is removed from service.

When am I giving up using NoSQL?

- Joins
- Group by
- Order by
- ACID transactions
- SQL as a sometimes frustrating but still powerful query language
- Easy integration with other applications that support SQL

Advantages of NoSQL

- Cheap, easy to implement
- Data are replicated and can be partitioned
- Easy to distribute
- Don't require a schema
- Can scale up and down
- Quickly process large amounts of data
- Relax the data consistency requirement (CAP)
- Can handle web-scale data, whereas Relational DBs cannot

Advantages of NoSQL

- Simple APIs
  - Java Example: Document.save(myObject)
- Good integration
- Designed to be horizontally scalable (elastic)
- Flexible data model
- Majority free and/or Open Source
- Free and Commercial production support

RDBMS advantages (don't forget!)

- Proven
- Available talent / Well-known
- Ad-hoc querying
- Scalable (limits?)
- Free and commercial production support

Disadvantages of NoSQL

- New and sometimes buggy
- Data is generally duplicated, potential for inconsistency
- No standardized schema
- No standard format for queries
- No standard language
- Difficult to impose complicated structures
- Depend on the application layer to enforce data integrity
- No guarantee of support
- Too many options. Difficult to know which one(s) to pick

Where would I use a NoSQL database?

- Do you have somewhere a large set of uncontrolled, unstructured, data that you are trying to fit into a RDBMS?
  - Log Analysis
  - Social Networking Feeds (many firms hooked in through Facebook or Twitter)
  - External feeds from partners (EAI)
  - Data that is not easily analyzed in a RDBMS such as time-based data
  - Large data feeds that need to be massaged before entry into an RDBMS

Hybrid approach: Polyglot persistence

Using different DB technologies for different storage requirements.

http://martinfowler.com/bliki/PolyglotPersistence.html