CSE 344: Section 5
NoSQL, SQL++

July 19th, 2018

Query workload types

"One Size Fits All": An Idea Whose Time Has Come and Gone

OLTP (Online **Transactional** Processing)

- Atomic operations (one or multi entities). E-commerce, webapps.
- A small number of records per query - "Latest state"

OLAP (Online **Analytical** Processing)

- Analytics and data-warehousing. Reporting, decision support.
- Many records per query - "Aggregated stats" on "Bigger data"

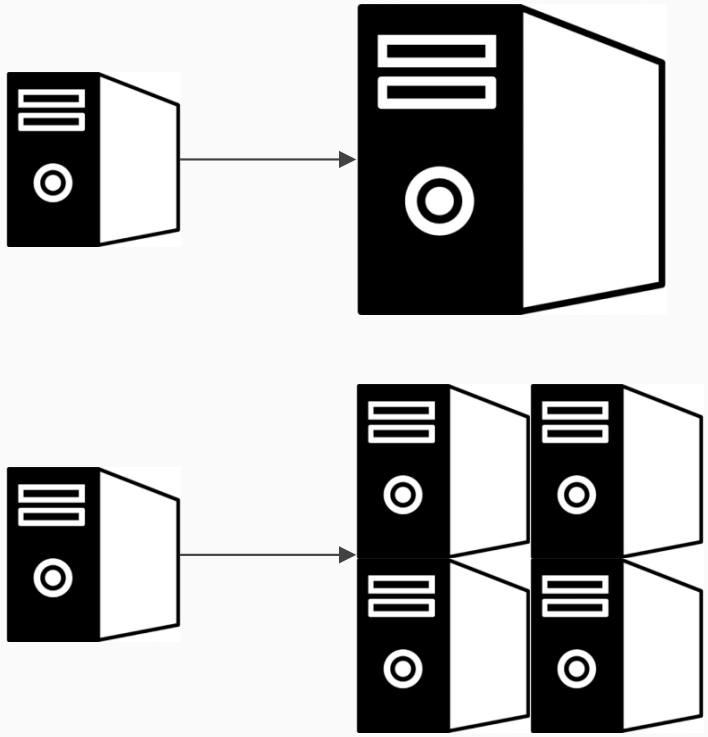
Scaling methods

Scale up (vertically)

- Add more power to a single node
- diminishing returns

Scale out (horizontally)

- Cheap commodity hardware
- Management / coordination complexity

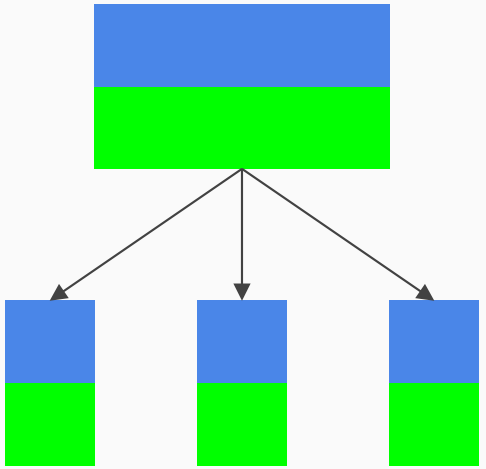


Partitioning & Replication

Partitioning

Or “Sharding”, “Distribution, ”Fragmentation”

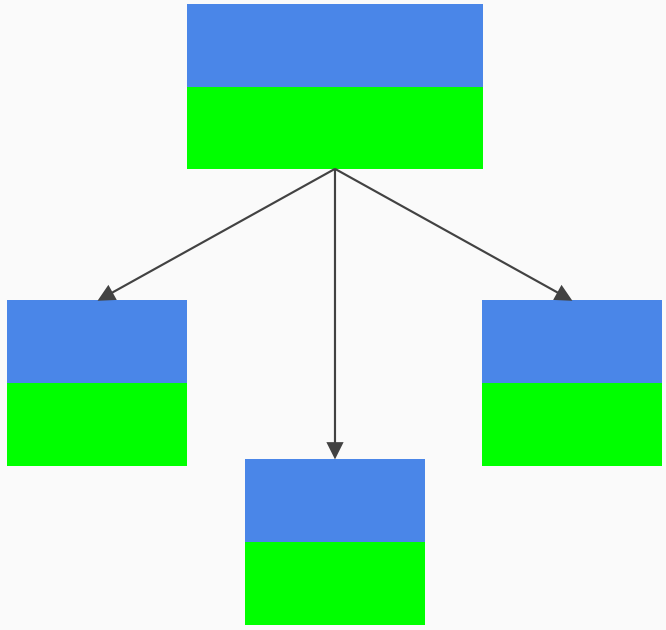
- Motivation:
  - BIG data - need to split up! (e.g. PB-level)
  - Availability: better write (and single-record read) throughput
- Challenge: fair share of requests
  - Choice of partitioning schemes
  - “Justin Bieber Effect” -> “hot spots”



Partitioning & Replication

Replication

- Motivation:
  - Fault-tolerance / durability: power / disk failures
  - Keep data close to the user (geographically)
  - Availability: better read (and potentially write) throughput
- Challenge: keeping data in sync
  - E.g. write to a leader and then propagate
  - Choice of consistency models



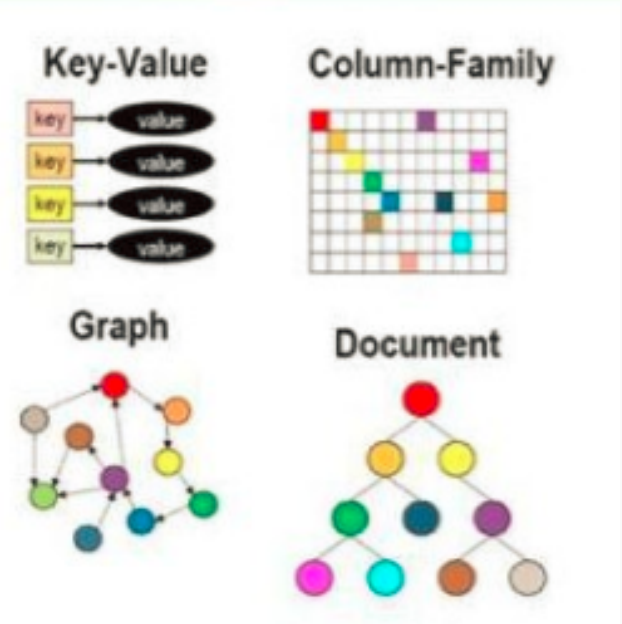
NoSQL

SQL vs. NoSQL Databases: What's the Difference?

- No clear definition :\
  - Non-relational
  - + **scalability**, + availability, + flexibility
  - - consistency, - OLAP performance
  - Open source implementations
- Motivation
  - The need to scale
  - Lots of web apps mostly **OLTP** queries
    - Read/write intensive
    - but fewer joins & aggregates

Data Models

- Key-value stores
  - Opaque value
  - e.g., Project Voldemort, Memcached
- **Document stores**
  - “key-object”
  - e.g., SimpleDB, CouchDB, MongoDB, AsterixDB
- Extensible Record Stores
  - “column groups”
  - e.g., BigTable, HBase, Cassandra, PNUTS
- Graph
  - E.g. Neo4j



JSON and Semi-Structured Data

JSON, XML, Protobuf (also an IDL)

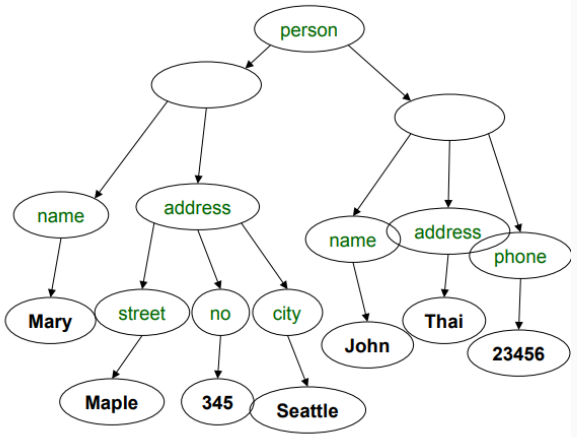
Familiar - as your HTTP request/response

- Good for data exchange
- Maps to OOP paradigm

Also - as a database file

- Flexible tree-structured model
- Query langs: XQuery, XPath, etc.


```
{"person":
  [ {"name": "Mary",
     "address":
        {"street":"Maple",
          "no":345,
          "city": "Seattle"}},
    {"name": "John",
     "address": "Thailand",
     "phone":2345678}}
  ]
}
```

AsterixDB, SQL++

- A semistructured NoSQL style data model (ADM)
- Extends JSON with object database ideas

Know the following:

- DDL: type (open vs. closed), data types (e.g. multiset). Creating an index.
- DML: Heterogenous Collections, Nesting / Unnesting.
- (Asterix stores data as flattened tables behind the scenes)

What is SQL++?

Just like SQL but parsed for processing JSON data

SQL++ has keywords to handle collections of data (i.e. non-flat data)

Motivation for SQL++

Why SQL++? Why not some other query language?

- People are used to/like specifying data through SQL syntax
- SQL-like language enforces idea of physical data independence

Useful Keywords/Syntax for HW

`is_array( … )` -----> checks if value is an array

`split(s, d)` -----> splits string s on delimiter d

`[ … ]` -----> explicitly construct array

`(CASE WHEN … THEN … ELSE … END)` -----> combine with "`is_array(…)`"

`MISSING` -----> reserved keyword like "`NULL`"

`` ` … ` `` -----> backtick needed for accessing keys with names containing "–"