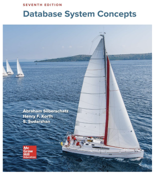
UMD DATA605 - Big Data Systems

Lesson 5.2: NoSQL Taxonomy

Instructor: Dr. GP Saggese - gsaggese@umd.edu

- References:
  - Online tutorials
  - Silbershatz: Chap 10.2
  - Seven Databases in Seven Weeks, 2e

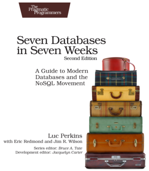

SEVENTH EDITION

Database System Concepts

Abraham Silberschatz

Henry F. Korth

S. Sudarshan


The Pragmatic Programmers

Seven Databases in Seven Weeks

Second Edition

A Guide to Modern Databases and the NoSQL Movement

Luc Perkins

with Eric Redmond and Jim R. Wilson

Series editor: Bruce A. Tate

Development editor: Jacquelyn Carter

DB Taxonomy

:::columns ::::{.column width=50%}

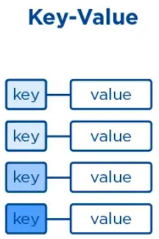
- **At least five DB genres**
  - *Relational* (e.g., Postgres)
  - *Key-value* (e.g., Redis)
  - *Document* (e.g., MongoDB)
  - *Columnar* (e.g., Parquet)
  - *Graph* (e.g., Neo4j)
- **Criteria to differentiate DBs**
  - Data model
  - Trade-off with CAP theorem
  - Querying capability
  - Replication scheme

Relational DB

- E.g., *Postgres*, MySQL, Oracle, SQLite
- **Data model**
  - Set-theory, relational algebra
  - Data as tables with rows, columns
  - Many attribute types (e.g., numeric, strings, dates, arrays, blobs)
  - Strictly enforced attribute types
  - SQL query language
  - ACID consistency
- **Application**
  - Relational tabular data
- **Good for**
  - Known data layout, unknown access pattern
  - Schema complexity for query flexibility
  - Regular data
- **Not so good for**
  - Hierarchical data (not a nice row in tables)
  - Variable/dishomogeneous data (record-to-record variation)

Key-Value Store

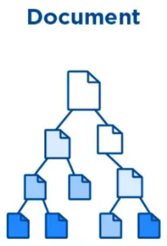
- E.g., Redis, DynamoDB, *Git*, AWS S3, filesystem
- **Data model**
  - Map keys (e.g., strings) to complex values (e.g., binary blob)
  - Support get, put, delete operations on a primary key
- **Application**
  - Cache data
  - Store users' session data in web applications
  - Store shopping carts in e-commerce applications
- **Good for**
  - Unrelated data (e.g., no joins)
  - Fast lookups
  - Easy horizontal scaling using partitioning
- **Not so good for**
  - Data queries
  - Lacking secondary indexes and scanning



SCIENCE ACADEMY

Document Store

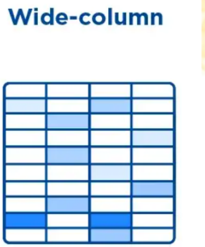
- E.g., *MongoDB*, *CouchBase*
- **Data model**
  - Key-value with document as value (nested dict)
  - Unique ID for each document (e.g., hash)
  - Any number of fields per document, including nested
    - E.g., jSON, XML, dict data
- **Application**
  - Semi-structured data
- **Good for**
  - Unknown data structure
  - Maps to OOP models (less impedance mismatch)
  - Easy to shard and replicate over distributed servers
- **Not so good for**
  - Complex join queries
  - Denormalized form is standard



SCIENCE ACADEMY

Columnar Store

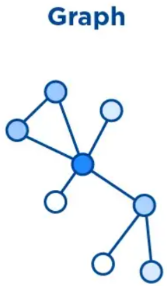
- E.g., *HBase*, Cassandra, *Parquet*
- **Data model**
  - Store data by columns, not rows
  - Similar to key-value and relational DBs
    - Use keys to query values
    - Values are groups of columns
- **Application**
  - Store web pages
  - Store time series data
  - OLAP workloads
- **Good for**
  - Horizontal scalability
  - Enable compression and versioning
  - Sparse tables without extra storage cost
  - Inexpensive to add columns
- **Not so good for**
  - Design schema based on query plans
  - No native joins; applications handle joins



SCIENCE ACADEMY

Graph DB

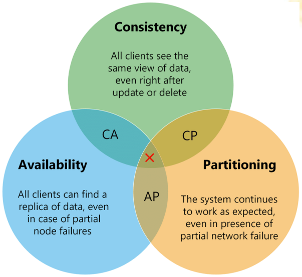
- E.g., *Neo4j*, GraphX
- **Data model**
  - Interconnected data: nodes, relationships
  - Nodes, edges have properties (key-value pairs)
  - Queries traverse nodes, relationships
- **Applications**
  - Social data
  - Recommendation engines
  - Geographical data
- **Good for**
  - Networked data, hard to model with relational model
  - Matches OO systems
- **Not so good for**
  - Poor scalability, hard to partition graph on different nodes
    - Store graph in graph DB, relations in key-value store



SCIENCE ACADEMY

Taxonomy by CAP

- **CA (Consistent, Available) systems**
  - Struggle with partitions, use replication
  - Traditional RDBMSs (PostgreSQL, MySQL)
- **CP (Consistent, Partition-Tolerant) systems**
  - Struggle with availability, maintain consistency across partitions
  - BigTable (column-oriented/tabular)
  - HBase (column-oriented/tabular)
  - MongoDB (document-oriented)
  - Redis (key-value)
  - MemcacheDB (key-value)
  - Berkeley DB (key-value)


Consistency
All clients see the same view of data, even right after update or delete
CA
CP
×
Availability
All clients can find a replica of data, even in case of partial node failures
AP
Partitioning
The system continues to work as expected, even in presence of partial network failure

- **AP (Available, Partition-Tolerant) systems**
  - Achieve “eventual consistency” via replication and verification
  - Dynamo (key-value)
  - Cassandra (column-oriented/tabular)
  - CouchDB (document-oriented)

SCIENCE ACADEMY