

Lesson 5.2: NoSQL Taxonomy

- **Instructor:** Dr. GP Saggese, gsaggese@umd.edu
- **References:**
 - Online tutorials
 - Silberschatz: Chap 10.2
 - Seven Databases in Seven Weeks, 2e



DB Taxonomy

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

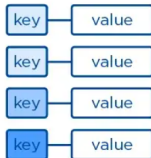
Relational DB

- E.g., *Postgres*, MySQL, Oracle, SQLite
- **Data model**
 - Set-theory, relational algebra
 - Data as tables with rows and columns
 - Many attribute types (e.g., numeric, strings, dates, arrays, blobs)
 - Strictly enforced attribute types
 - SQL query language
 - ACID compliance
- **Good for**
 - Known data layout, unknown access pattern
 - Schema complexity for query flexibility
 - Regular data
- **Not so good for**
 - Hierarchical data (not easily represented as rows in tables)
 - Variable/heterogeneous 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

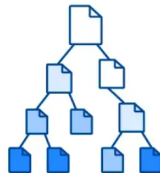
Key-Value



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

Document



Columnar Store

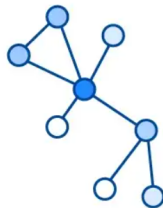
- E.g., *HBase*, *Cassandra*, *Parquet*
- **Data model**
 - Store data by columns, not rows
 - Similar to key-value and relational DBs
 - Use keys to query values
 - Values are groups of columns
- **Application**
 - Store web pages
 - Store time series data
 - OLAP workloads
- **Good for**
 - Horizontal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

Wide-column

Graph DB

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

Graph



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)
- Redis (key-value)
- Berkeley DB (key-value)

- **AP (Available, Partition-Tolerant) systems**

- Achieve “eventual consistency” via replication and verification
- MongoDB (document-oriented)
- Memcached (key-value)
- Dynamo (key-value)
- Cassandra (column-oriented/tabular)
- CouchDB (document-oriented)

