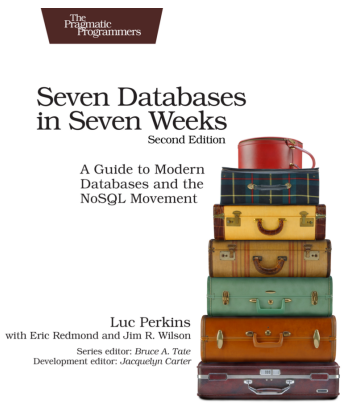
UNIVERSITY OF MARYLAND

UMD DATA605 - Big Data Systems

NoSQL Taxonomy

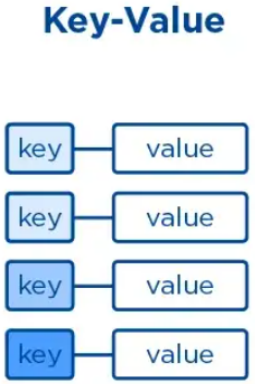


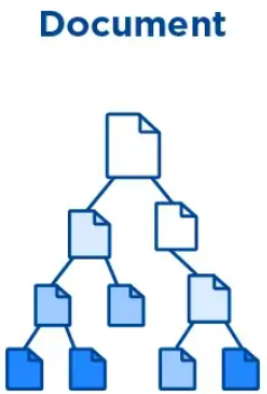
Instructor: Dr. GP Saggese - gsaggese@umd.edu

- Concepts in slides
- Silberschatz Chapter 23.6
- High-level view:
  - Seven Databases in Seven Weeks, 2e

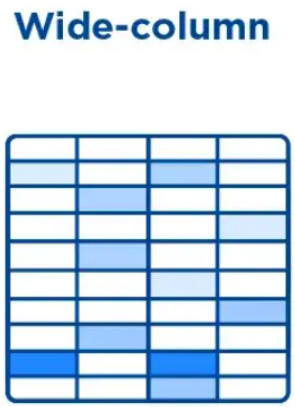
DB Taxonomy

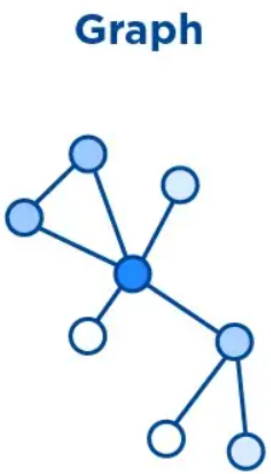
- **At least five DB genres**
  - *Relational* (e.g., Postgres)
  - *Key-value* (e.g., Redis)
  - *Document* (e.g., MongoDB)
  - *Columnar* (e.g., Parquet)
  - *Graph* (e.g., Neo4j)





- **Criteria to differentiate DBs**
  - Data model
  - Trade-off with CAP theorem
  - Querying capability
  - Replication scheme



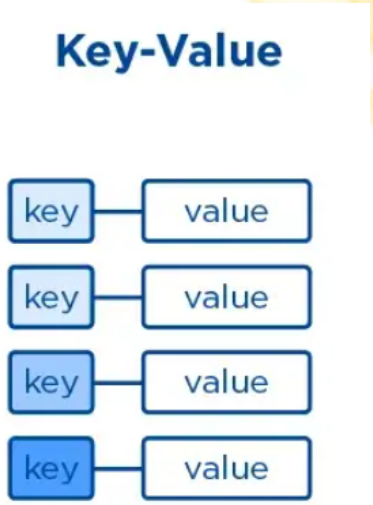


Relational DB

- 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)

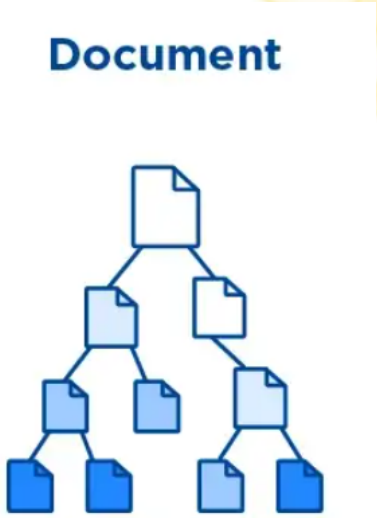
Key-Value Store

- 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**
  - Caching data
  - Store users' session data in web applications
  - Store shopping carts in e-commerce applications
- **Good for**
  - Data not "related" (e.g., no joins)
  - Fast lookups
  - Easy to scale horizontally using partitioning
- **Not so good for**
  - Data queries needed
  - Lacking secondary indexes and scanning



Document Store

- E.g., *MongoDB*, CouchDB
- **Data model**
  - Like key-value but value is a document (nested dict)
  - Each document has a unique ID (e.g., hash)
  - Any number of fields per document, even nested
    - E.g., jSON, XML, dict data
- **Application**
  - Semi-structured data
- **Good for**
  - Unknown data structure
  - Maps well to OOP models (less impedance mismatch)
  - Easy to shard and replicate over distributed servers
- **Not so good for**
  - Complex join queries
  - Denormalized form is the norm

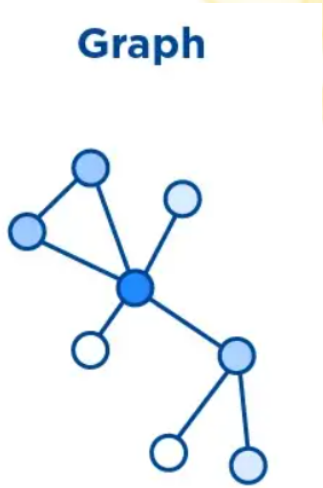


Columnar Store

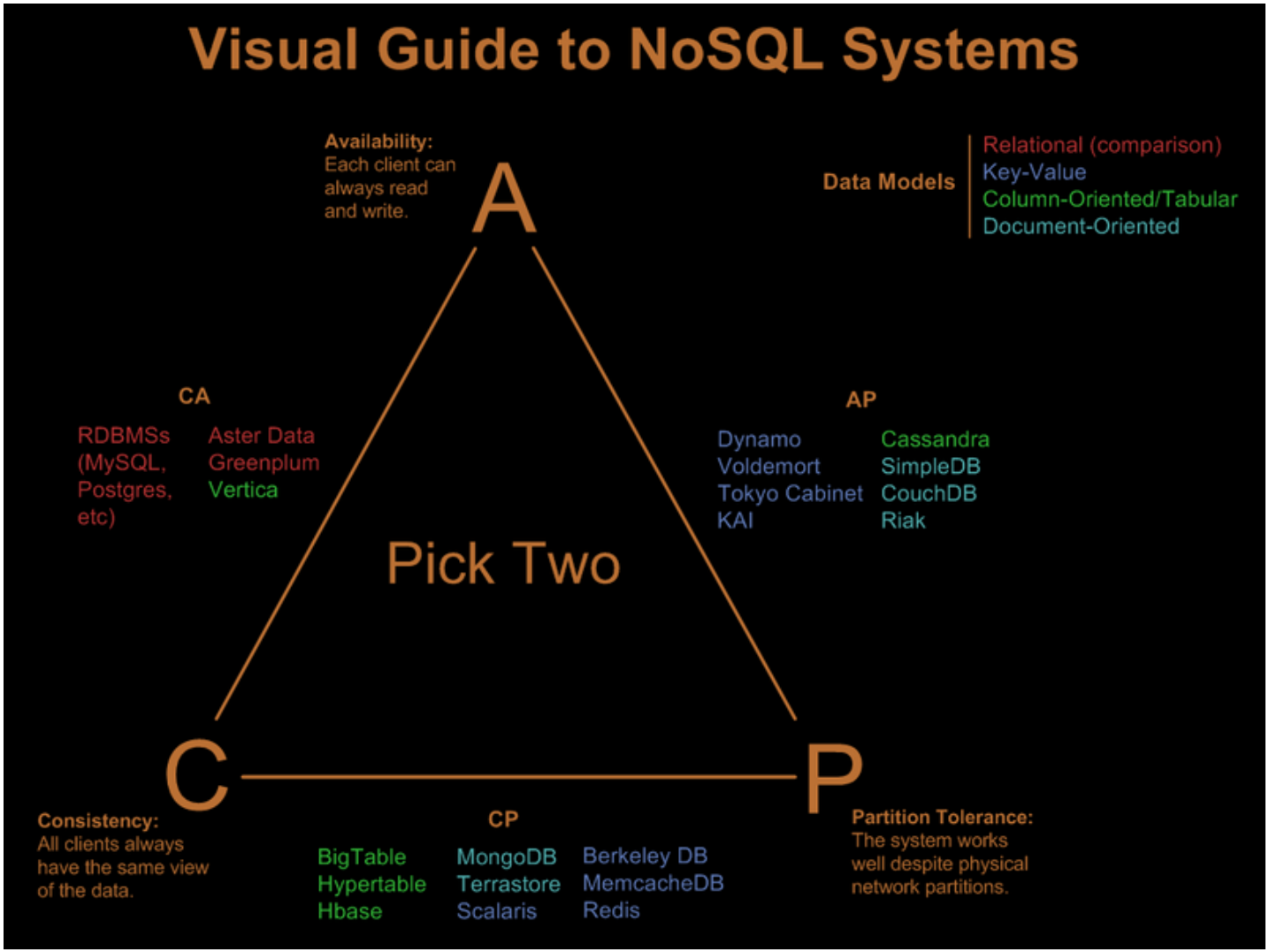
- E.g., *HBase*, Cassandra, *Parquet*
- **Data model**
  - Store data by columns, not rows
  - Similar to key-value and relational DBs
    - Use keys to query values
    - Values are groups of columns
- **Application**
  - Store web pages
  - Store time series data
  - OLAP workloads
- **Good for**
  - 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

Graph DB

- E.g., *Neo4J*, GraphX
- **Data model**
  - Interconnected data, nodes, and 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, difficult to model with relational model
  - Matches OO systems
- **Not so good for**
  - Poor scalability, difficult to partition graph on different nodes
    - Store graph in graph DB, relations in key-value store



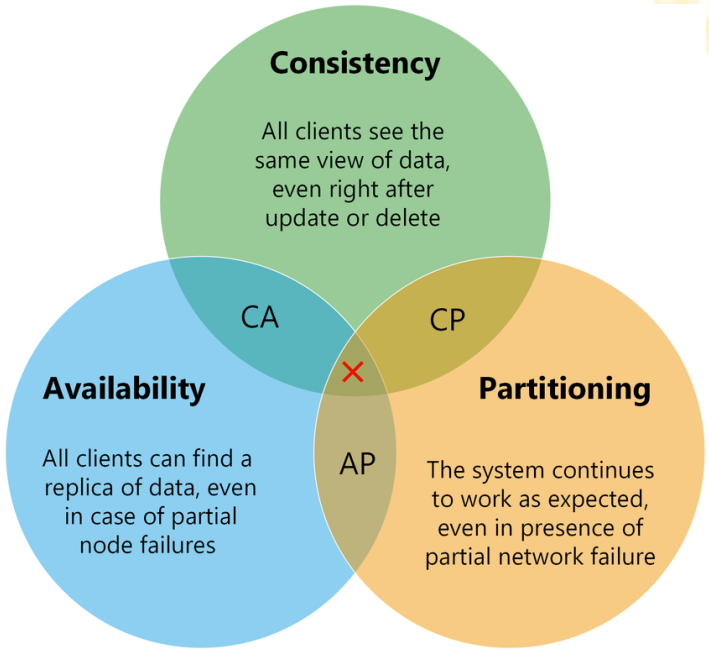
Taxonomy by CAP



From http://blog.nahurst.com/visual-guide-to-nosql-systems

Taxonomy by CAP

- **CA (Consistent, Available) systems**
  - Struggle with partitions, use replication
  - Traditional RDBMSs (e.g., 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)



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