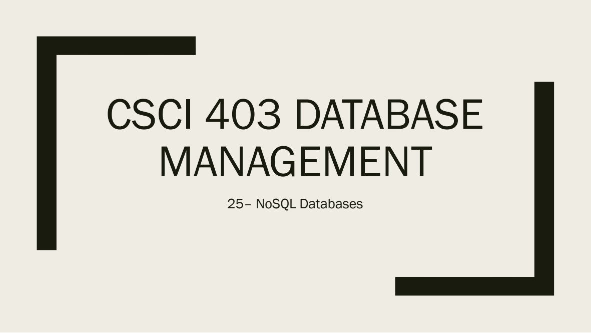
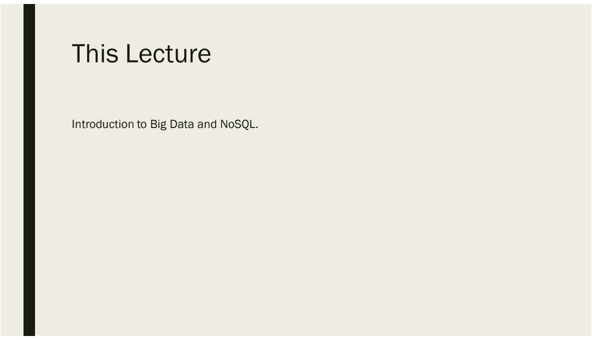


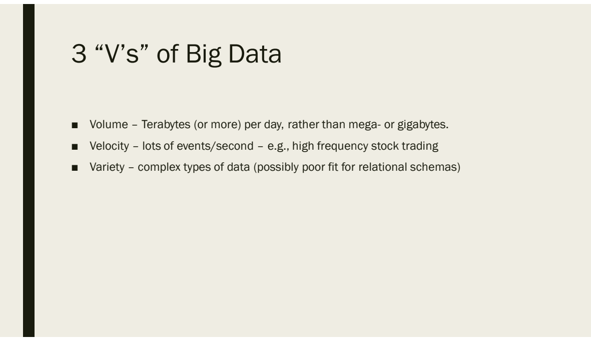
CSCI 403 DATABASE MANAGEMENT

25– NoSQL Databases



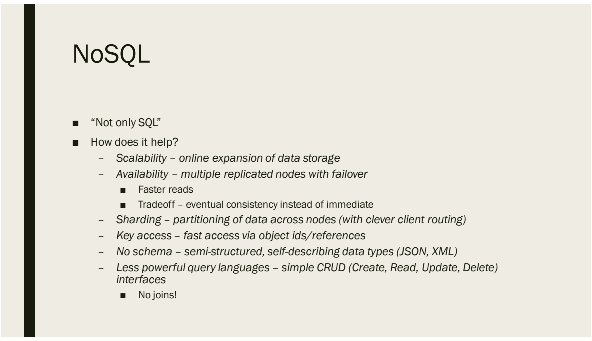
This Lecture

Introduction to Big Data and NoSQL.



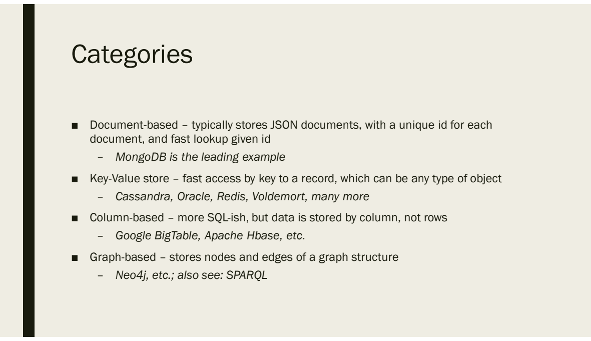
3 "V's" of Big Data

- Volume – Terabytes (or more) per day, rather than mega- or gigabytes.
- Velocity – lots of events/second – e.g., high frequency stock trading
- Variety – complex types of data (possibly poor fit for relational schemas)



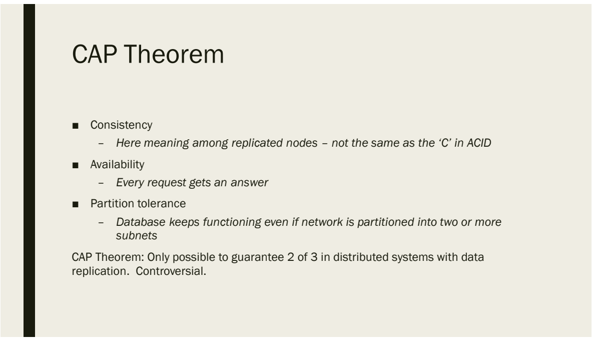
NoSQL

- "Not only SQL"
- How does it help?
  - *Scalability – online expansion of data storage*
  - *Availability – multiple replicated nodes with failover*
    - Faster reads
    - Tradeoff – eventual consistency instead of immediate
  - *Sharding – partitioning of data across nodes (with clever client routing)*
  - *Key access – fast access via object ids/references*
  - *No schema – semi-structured, self-describing data types (JSON, XML)*
  - *Less powerful query languages – simple CRUD (Create, Read, Update, Delete) interfaces*
    - No joins!



Categories

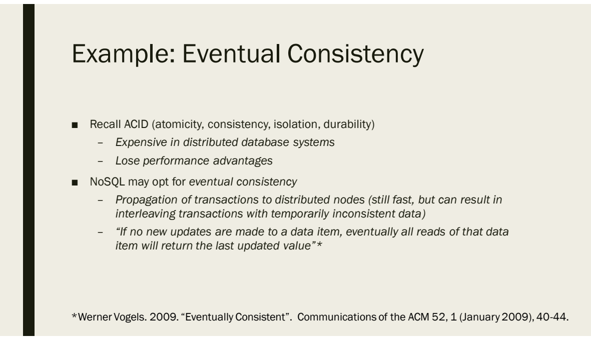
- Document-based – typically stores JSON documents, with a unique id for each document, and fast lookup given id
  - *MongoDB is the leading example*
- Key-Value store – fast access by key to a record, which can be any type of object
  - *Cassandra, Oracle, Redis, Voldemort, many more*
- Column-based – more SQL-ish, but data is stored by column, not rows
  - *Google BigTable, Apache Hbase, etc.*
- Graph-based – stores nodes and edges of a graph structure
  - *Neo4j, etc.; also see: SPARQL*



CAP Theorem

- Consistency
  - *Here meaning among replicated nodes – not the same as the 'C' in ACID*
- Availability
  - *Every request gets an answer*
- Partition tolerance
  - *Database keeps functioning even if network is partitioned into two or more subnets*

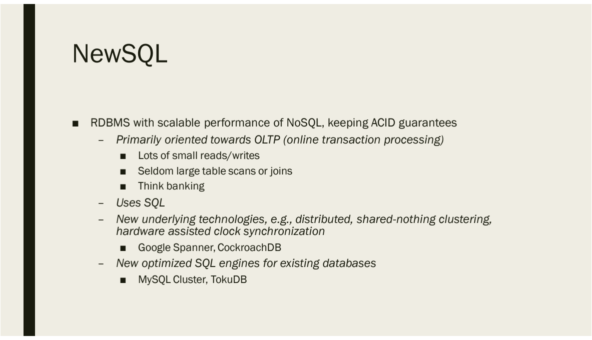
CAP Theorem: Only possible to guarantee 2 of 3 in distributed systems with data replication. Controversial.



Example: Eventual Consistency

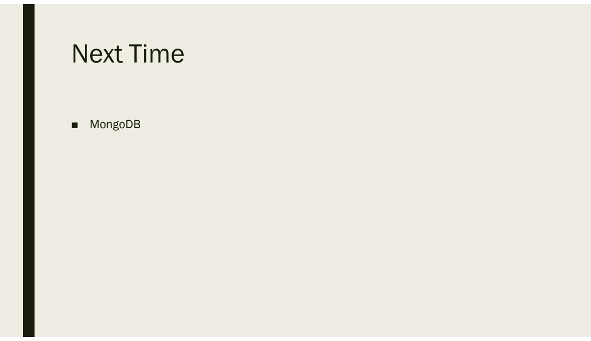
- Recall ACID (atomicity, consistency, isolation, durability)
  - *Expensive in distributed database systems*
  - *Lose performance advantages*
- NoSQL may opt for *eventual consistency*
  - *Propagation of transactions to distributed nodes (still fast, but can result in interleaving transactions with temporarily inconsistent data)*
  - *"If no new updates are made to a data item, eventually all reads of that data item will return the last updated value"**

*Werner Vogels. 2009. "Eventually Consistent". Communications of the ACM 52, 1 (January 2009), 40-44.



NewSQL

- RDBMS with scalable performance of NoSQL, keeping ACID guarantees
  - *Primarily oriented towards OLTP (online transaction processing)*
    - Lots of small reads/writes
    - Seldom large table scans or joins
    - Think banking
  - *Uses SQL*
  - *New underlying technologies, e.g., distributed, shared-nothing clustering, hardware assisted clock synchronization*
    - Google Spanner, CockroachDB
  - *New optimized SQL engines for existing databases*
    - MySQL Cluster, TokuDB



Next Time

- MongoDB