Big Data Technologies: NoSQL Databases

University of California, Berkeley

School of Information

INFO 257: Database Management

Today's Class

- How is everyone?
- Questions?
- Future Workshop (Github Desktop) After Spring Break
- Assignment 4 (Final) to be released during Spring Break
- Today's Workshop
  - Compass
  - MongoDB Command Line Client

Where are We?

	Operational	Informational	
	Transactional	Analytical– Data Warehousing	Analytical– Big Data
Technology	Relational	Relational	Non-relational
Modeling	Conceptual data modeling with (E)ER (Chapters 2 and 3)		
Design	Logical data modeling with the relational model; Normalization (Chapter 4)	Data warehousing and data integration (Chapter 9)	Big data technologies, including Hadoop & NoSQL (Chapter 10)
Infrastructure	Physical design of relational databases; Security; Cloud computing (Chapter 8)		
Access	SQL (Chapters 5 and 6) Applications with SQL (Chapter 7)		
Data analysis	Analytics and its implications (Chapter 11)		
Governance and data management	Lifecycle (Chapter 1) Governance, data quality, and master data management (Chapter 12)		

Lecture Outline

FYI – Today's Lecture Pulled from 3 sources

- Big Data - NoSQL Databases
- SQL vs NoSQL
- MongoDB

Introduction

- Big Data
  - Data that exist in very large volumes and many different varieties (data types) and that need to be processed at a very high velocity (speed).
- Analytics
  - Systematic analysis and interpretation of data—typically using mathematical, statistical, and computational tools—to improve our understanding of a real-world domain.

Characteristics of Big Data (1 of 2)

- The Five V's of Big Data
  - Volume – much larger quantity of data than typical for relational databases
  - Variety – lots of different data types and formats
  - Velocity – data comes at very fast rate (e.g. mobile sensors, Web click stream)
  - Veracity – traditional data quality methods don't apply; how to judge the data's accuracy and relevance?
  - Value – big data is meaningless if it does not provide value toward some meaningful goal

Characteristics of Big Data (2 of 2)

- Schema on Read, rather than Schema on Write
  - Schema on Write – preexisting data model, how traditional databases are designed (relational databases)
  - Schema on Read – data model determined later, depends on how you want to use it (XML, JSON)
  - Capture and store the data, and worry about how you want to use it later
- Data Lake
  - A large integrated repository for internal and external data that does not follow a predefined schema
  - Capture everything, dive in anywhere, flexible access

Examples of JSON and XML

JSON Example

```
{"products": [
        {"number": 1, "name": "Zoom X", "Price": 10.00},
        {"number": 2, "name": "Wheel Z", "Price": 7.50},
        {"number": 3, "name": "Spring 10", "Price": 12.75}
]}
```

XML Example

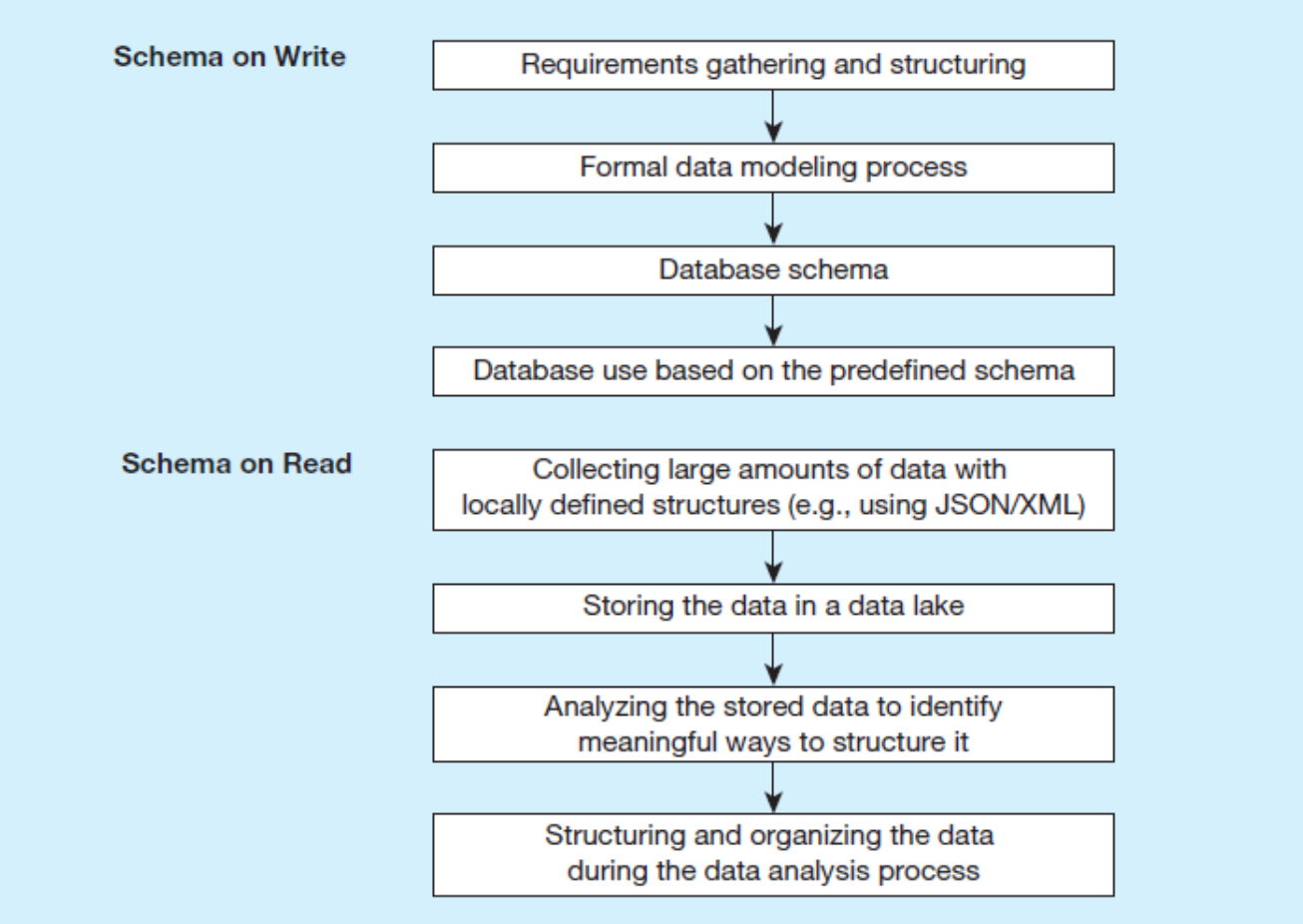
```
<products>
        <product>
                <number>1</number> <name>Zoom X</name> <price>10.00</price>
        </product>
        <product>
                <number>2</number> <name>Wheel Z</name> <price>7.50</price>
        </product>
        <product>
                <number>3</number> <name>Spring 10</name> <price>12.75</price>
        </product>
</products>
```

JSON = JavaScript Object Notation XML = eXtensible Markup Language

Schema on Write vs. Schema on Read

Traditional database design

The big data approach



NoSQL

- NoSQL = Not Only SQL
- A category of recently introduced data storage and retrieval technologies not based on the relational model
- Scaling out rather than scaling up
- Natural for a cloud environment
- Supports schema on read
- Largely open source
- Not ACID compliant!
- BASE – basically available, soft state, eventually consistent

NoSQL Classifications (1 of 2)

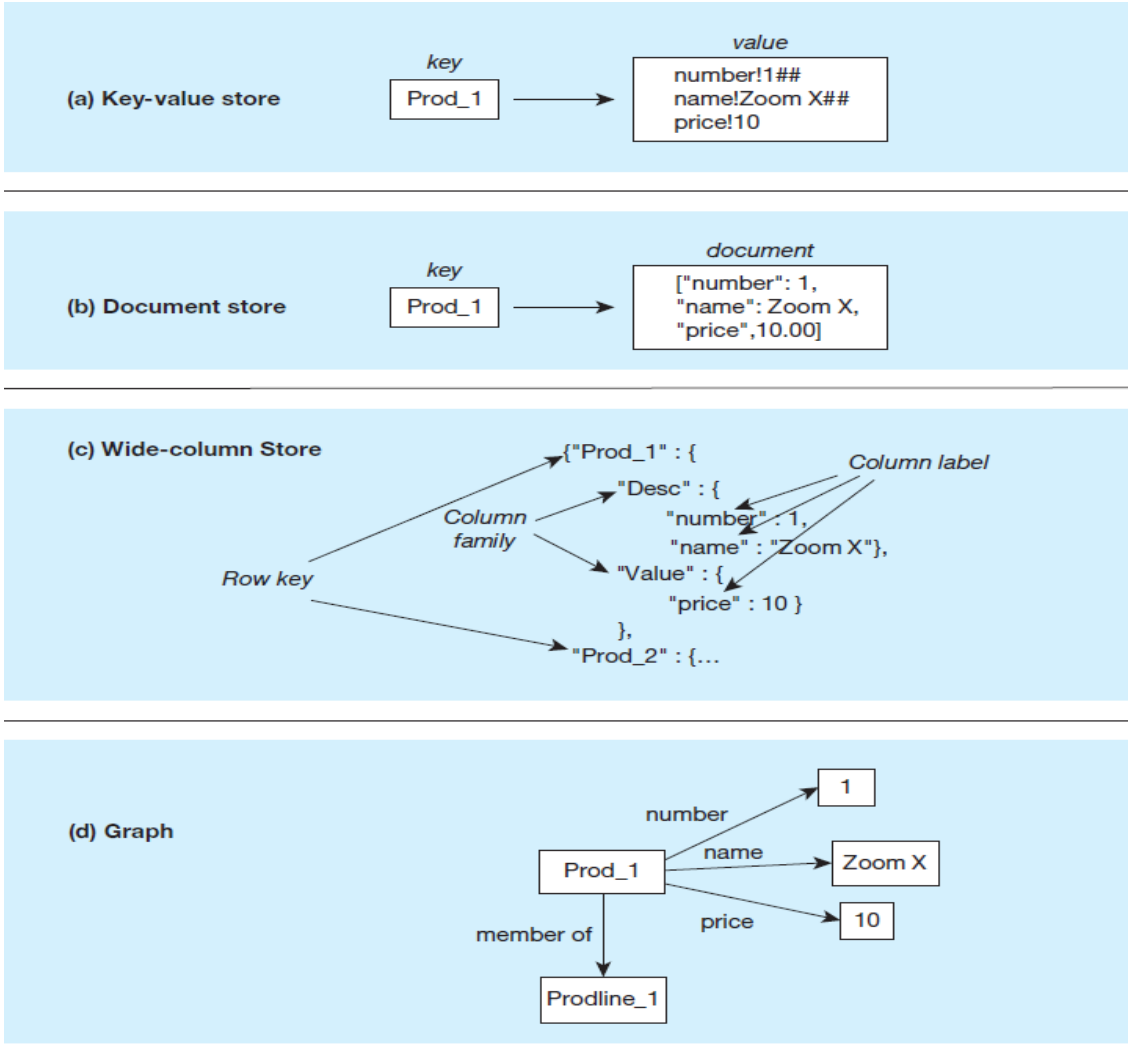
- Key-value stores
  - A simple pair of a key and an associated collection of values. Key is usually a string. Database has no knowledge of the structure or meaning of the values.
  - Example: Redis
- Document stores
  - Like a key-value store, but “document” goes further than “value”. Document is structured so specific elements can be manipulated separately.
  - Example: MongoDB

NoSQL Classifications (2 of 2)

- Wide-column stores
  - Rows and columns. Distribution of data based on both key values (records) and columns, using "column groups/families"
  - Example: Apache Cassandra
- Graph-oriented database
  - Maintain information regarding the relationships between data items. Nodes with properties. Connections between nodes (relationships) can also have properties.
  - Example Neo4j

NoSQL Databases

Some of the example structures have been adapted from Kauhanen (2010)



Comparison of NoSQL Database Characteristics

Source: www.slideshare.net/bscofield/nosql-codemash-2010. Courtesy of Ben Scofield.

	Key-Value Store	Document Store	Column Oriented	Graph
Performance	High	High	High	Variable
Scalability	High	Variable/High	High	Variable
Flexibility	High	High	Moderate	High
Complexity	None	Low	Low	High
Functionality	Variable	Variable (Low)	Minimal	Graph theory

SQL and NoSQL DBs Comparison

Switch to Presentation from Keith Hare

NoSQL Database Architectures

NoSQL Database

- NoSQL databases use a variety of file structures and access methods for their operation
- There is very little commonality across the different NoSQL DBs in terms of file storage
- We will look at a couple of examples
  - BerkeleyDB – the grand-daddy of NoSQL DBs
  - MongoDB – One of the best known NoSQL DBs

BerkeleyDB Architecture

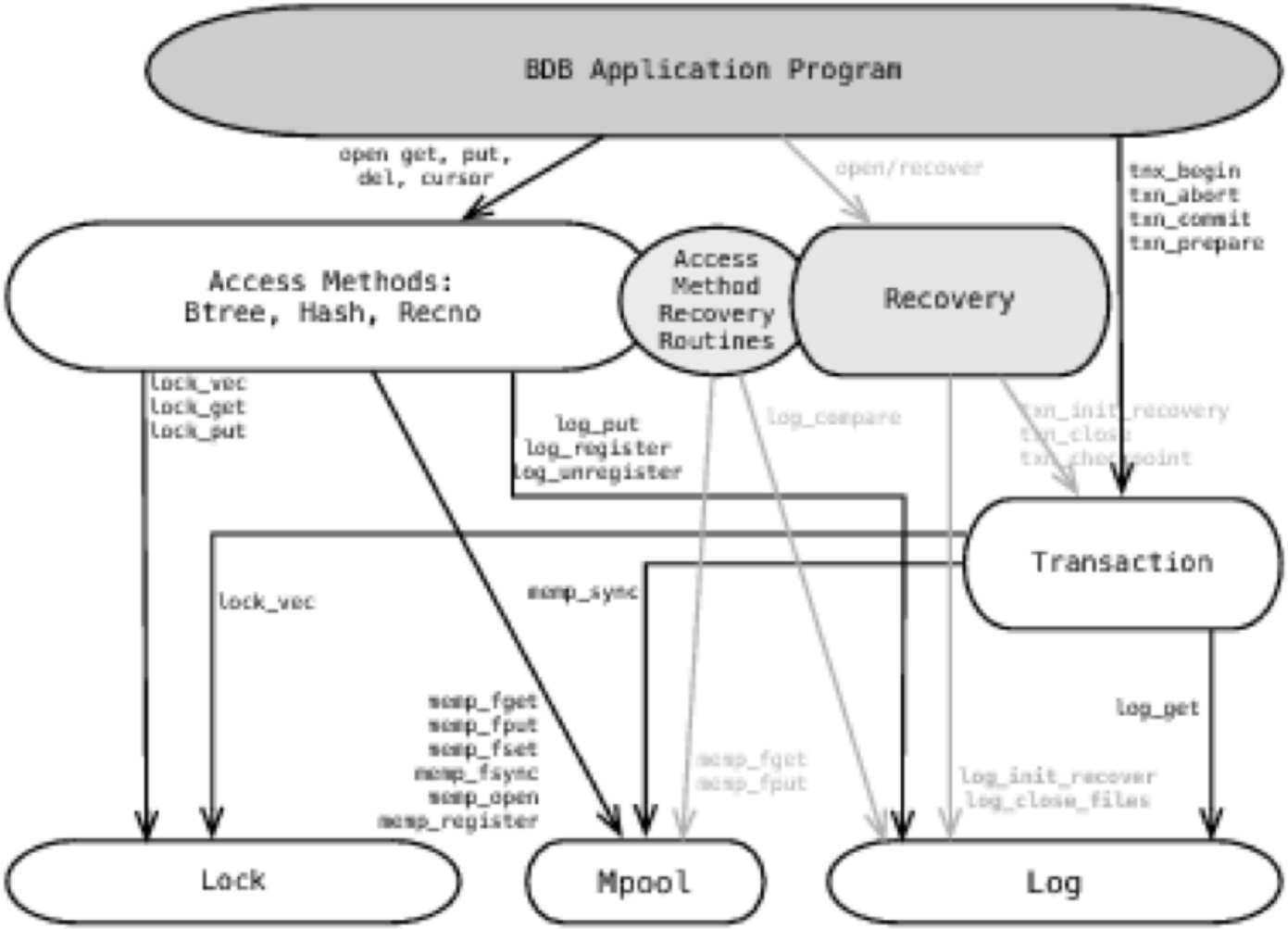


Figure 4.3: Actual Berkeley DB 2.0.6 Architecture.

BerkeleyDB

- The system provides three types of underlying file DBMS access methods
  - RecordID
  - Btree
  - Hashed
- RecordID is a simple numeric record lookup
- Btree uses clever caching to keep the frequently used and higher tree levels in memory
- Hash uses extensible hashing

MongoDB Storage

- MongoDB uses memory-mapped files for data storage
- A memory-mapped file is a file with data that the operating system places in memory by way of the mmap() system call. mmap() thus maps the file to a region of virtual memory.
- Memory-mapped files are the critical piece of the storage engine in MongoDB.
- By using memory mapped files MongoDB can treat the contents of its data files *as if they were in memory.*

MongoDB Storage

- This provides MongoDB with an extremely fast and simple method for accessing and manipulating data.
- Memory mapping assigns files to a block of virtual memory with a direct byte-for-byte correlation. Once mapped, the relationship between file and memory allows MongoDB to interact with the data in the file as if it were memory.

MongoDB Storage

- How does MongoDB work with memory mapped files?
- MongoDB uses memory mapped files for managing and interacting with all data.
- MongoDB memory maps data files to memory as it accesses documents.
- Data that isn't accessed is not mapped to memory.

NoSQL Example: MongoDB

- A document-store database
- BSON-based storage format (Binary JSON)
- Collections
  - Equivalent to tables in a relational database
  - A set of documents intended to be stored together
- Documents
  - Equivalent to rows in a relational database
  - Documents do not need to have the same structure (unlike rows)
  - _id property for uniquely identifying a row
- Relationships
  - _id property serves as “primary key”
  - Another document can have a “foreign” key as another JSON property

Sample Mongo Documents with Relationships

a) A document in the Product collection

```
{
  "_id": "1",
  "name": "OLED TV",
  "desc": "75in TV",
  "width": 60,
  "height": 30,
  "depth": 5,
  "reviews": [
    {
      "author": 1,
      "ratingstars": 4,
      "comment": "Amazing TV"
    },
    {
      "author": 2,
      "ratingstars": 2,
      "comment": "Very disappointed with the TV"
    }
  ]
}
```

b) A document in the Author collection

```
{
  "_id": 1
  "First Name": "Jane",
  "Last Name: "Smith"
}
```

Deeper Dive on MongoDB

Switch to Lecture from Sudeepa Roy