

ECE:5995 Modern Databases

Fall 2020

Teaching staff

Instructor:

- Guadalupe Canahuat
 - guadalupe-canahuat@uiowa.edu
 - Office Hours: MW 1:00-2:30 PM through zoom
 - Office: 5015 SC
-
- Check ICON Course website for announcements, assignments, example code, zoom links, and lecture slides

Course Contents

- The goal is to gain an understanding of the relative strengths and weaknesses of the different database styles.
- The class is organized into four modules covering one relational (SQL) and 3 NoSQL databases:
 - A relational database management system (RDBMS): Postgres
 - A document store: MongoDB
 - A graph database: Neo4J
 - A key-value store in-memory database: Redis

Course Evaluation

	Weight
ICON Quizzes (at least 5)	40%
Homeworks/projects (at least 5)	40%
Final project	15%
Attendance/participation	5%
Total	100%

Course Outline (tentative)

Lecture	Topic
1	Introduction and organization
2	Intro to Postgres, relations, and CRUD
3	T-SQL
4	Joins and query processing
5	Indexes: B+tree and Hashing
6	Complex queries, Aggregations
7	ER diagrams and Normalization
8	Transactions and Stored Procedures
9	Triggers and Materialized Views
10	Full text and fuzzy search
11	Multidimensional queries
12	Intro to MongoDB and document stores
13	CRUD in Mongo
14	Indexing and Aggregating
15	Mapreduce
16	Replicas and Sharding
17	Geospatial queries


Lecture	Topic
18	Intro to Neo4j and graph stores
19	Cypher and CRUD
20	Indexes and constraints
21	REST
22	Indexes and algorithms
23	Distributed high-availability
24	Redis and Key-value stores
25	CRUD and Datatypes
Thanksgiving recess	
26	Advanced usage
27	Distribution and Bloom filters
28	Polyglot persistent storage
29	Wrap-up

Database Management System (DBMS)

- A DBMS *manages* a *database*.
- A database is a collection of data, usually with some description of the structure of the data.
 - Structure description, if present, is described using a *schema*. e.g. the CREATE TABLE command in SQL



Data Storage Management

- Store and retrieve data in an efficient way
 - Organize data in blocks called pages on disk
 - “Index” data for efficient retrieval
 - Make efficient use of memory hierarchy
 - Cache frequently used data in a main memory **buffer pool**
 - Safely allow concurrent access to the data
 - Make sure updates are “committed”
- 
- i.e. provide transactional semantics**

Describe and Query the Data

Describe the data

- **Data model** is the abstraction to describe the data
- A **schema** describes a specific database using the “language” of the data model
 - E.g. In the relational data model the CREATE TABLE command is used to express the schema of a table

Query the data

- Provide a high-level language to allow a user to pose queries easily
 - Declarative languages are preferred, e.g. SQL
- Need **query processing algorithms** to evaluate the query and techniques to **optimize** the query

Data Management Systems: Three Common Types

1. Relational Database Management Systems: **RDBMS**
 - e.g. PostgreSQL, Sqlite3, MySQL, Oracle, SQL Server ...
 - Store data as tuples in tables. Query using SQL.
2. **NoSQL (Key-value (KV) stores, Document Stores, Object Stores, Column stores)**
 - e.g. BigTable, Hbase, Dynamo, Cassandra, MongoDB, CouchDB, Neo4j...
3. **MapReduce (MR)**
 - Works on top of a key-value distributed file system (DFS).
 - Invented by Google to run data processing on large clusters. Open-source version is called **Hadoop**.
 - A new trend is to put a SQL interface on top of MR. e.g. Hive

MR interface:

- Data = Set of $\langle k1, v1 \rangle$ pairs
- $\text{Map}(k1, v1) \rightarrow \langle k2, v2 \rangle$ // for every key-value pair in the input, output 0 or more key-values
- $\text{Reduce}(k2, \text{list-values-with-key-}k2) \rightarrow \langle k3, v3 \rangle$ // Final result is also key-value pairs

RDBMS

-- Section 1: Creating schemas in SQL, i.e. SQL DDL

-- Create a table to store student information

```
CREATE TABLE Students ( name VARCHAR(80),  
                        bday DATE,  
                        hobbies VARCHAR(100),  
                        uwid INTEGER,  
                        PRIMARY KEY (uwid) -- Do not allow two tuples with the same uwid  
);
```

-- Add sample tuples to the Student table

```
INSERT INTO Students VALUES ('Jane Doe', '1990-03-01', 'sailing', 111);  
INSERT INTO Students VALUES ('Joe Smith', '1991-05-12', 'dancing', 222);  
INSERT INTO Students VALUES ('Goof Ball', '1992-12-31', 'watching TV', 333);
```

-- Section 2: Querying in SQL -- i.e. SQL DML

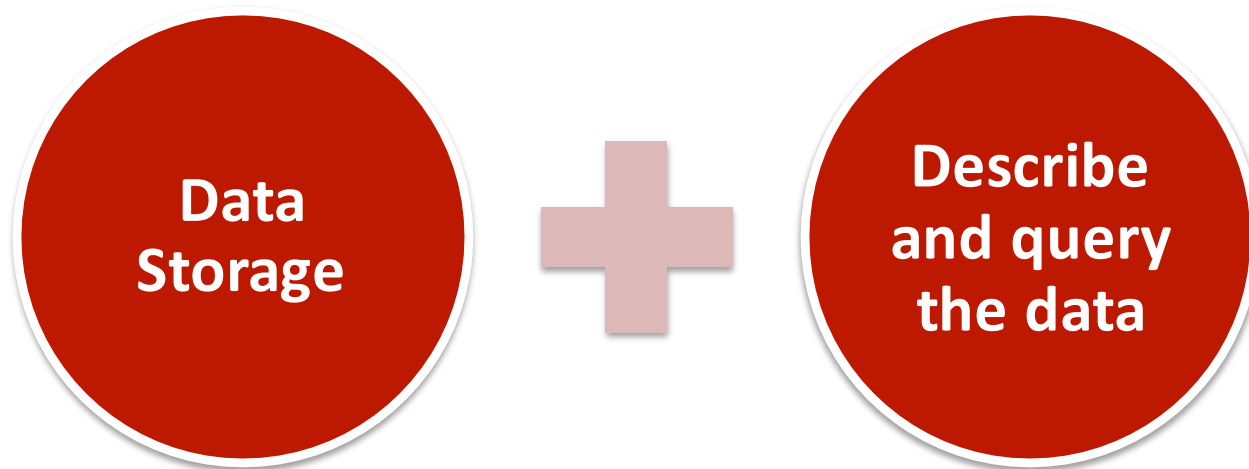
```
SELECT * FROM Students WHERE bday > '1991-01-01' AND hobbies <> 'watching TV';
```

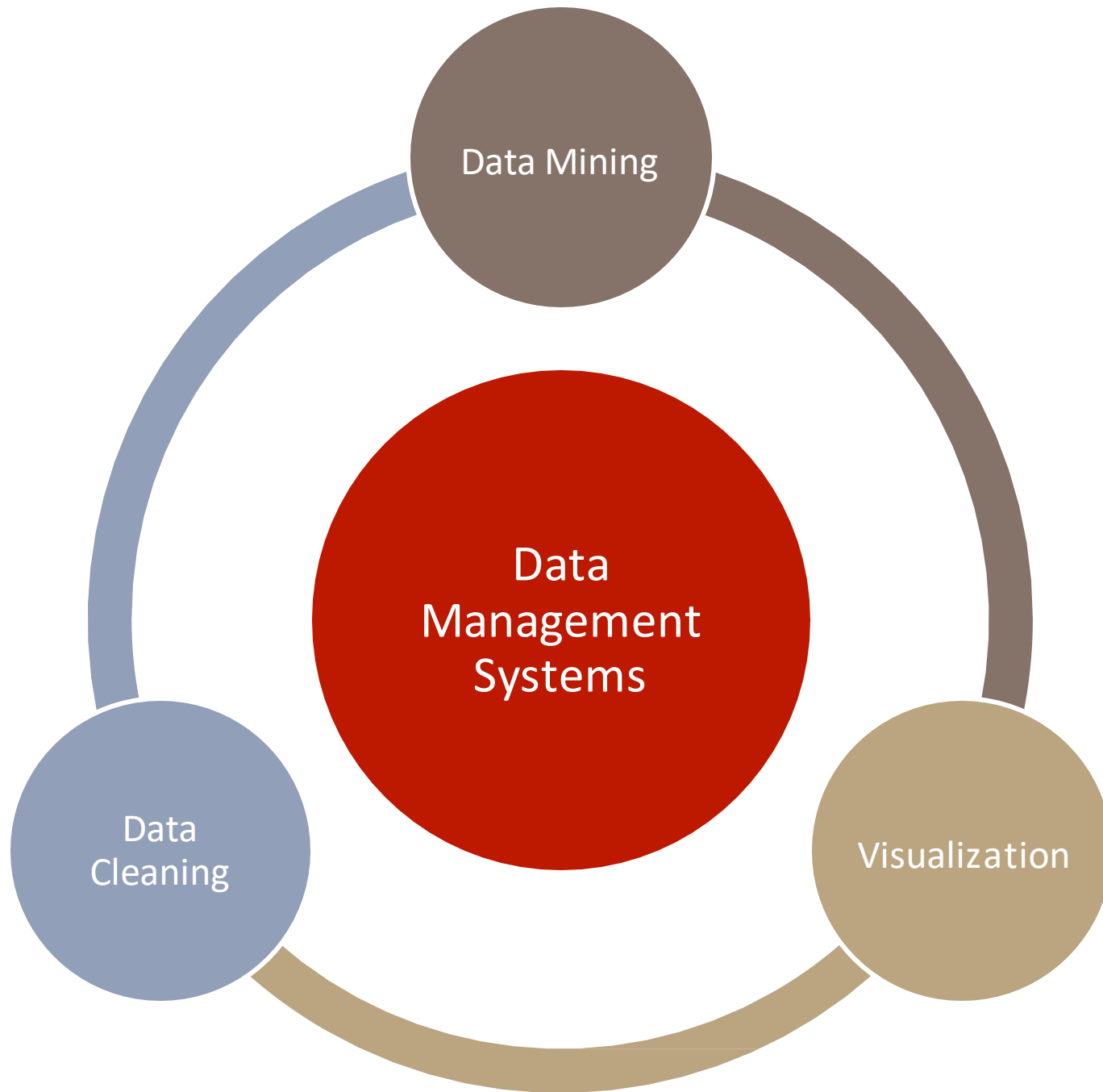
NoSQL example: MongoDB

```
MongoDB browser shell version: 0.1.0
connecting to random database
type "help" for help
type "tutorial" to start the tutorial
> db.students.save({name: "James Bond", age: 21, uwid: 111})
"ok"
> db.students.save({name: "Jane Cool", age: 20, uwid: 222})
"ok"
> db.students.find({age: 21})

[
  {
    "name" : "James Bond",
    "_id" : { "$oid" : "50fdffdbcc93742c16007880" },
    "uwid" : 111,
    "age" : 21
  }
]
> |
```

Common Requirement Across All Data Management Systems





"Data is the new 'oil' and there is a growing need for the ability to refine it,"

Dhiraj Rajaram, Founder & CEO of Mu Sigma, leading Data Analytics co.

SQL vs. NoSQL

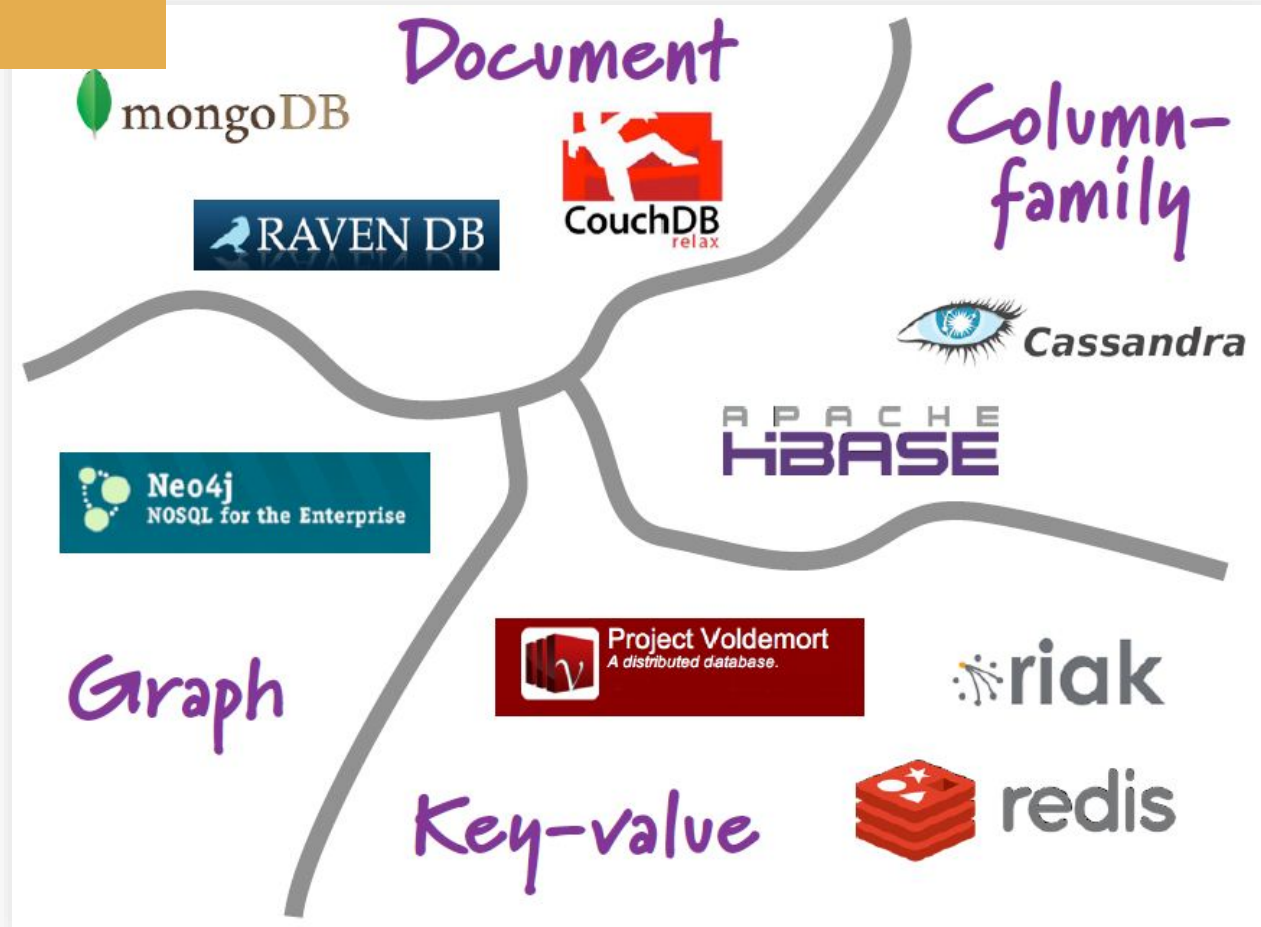
- SQL is the query language for Relational Databases
- NoSQL -> Not Only SQL or NO! SQL
- A more general definition... a datastore that does not follow the relational model including using SQL to interact with the data.
- Why?
 - One size does not fit all
 - Relational Model has scaling issues
 - Freedom from the tyranny of the DBA?

LIST OF NOSQL DBMS [currently >225]

<http://www.nosql-database.org/>

SQL vs. NoSQL

SQL - RDBMS Databases



SQL Characteristics

- Data stored in columns and tables
- Relationships represented by data
- Data Manipulation Language
- Data Definition Language
- Transactions
- Abstraction from physical layer

SQL Physical Layer Abstraction

- Applications specify what, not how
- Query optimization engine
- Physical layer can change without modifying applications
 - Create indexes to support queries
 - In Memory databases

Data Manipulation Language (DML)

- Data manipulated with Select, Insert, Update, & Delete statements
 - Select T1.Column1, T2.Column2 ...
From Table1, Table2 ...
Where T1.Column1 = T2.Column1 ...
- Data Aggregation
- Compound statements
- Functions and Procedures
- Explicit transaction control

Data Definition Language

- Schema defined at the start
- Create Table (Column1 Datatype1, Column2 Datatype 2, ...)
- Constraints to define and enforce relationships
 - Primary Key
 - Foreign Key
 - Etc.
- Triggers to respond to Insert, Update , & Delete
- Stored Modules
- Alter ...
- Drop ...
- Security and Access Control

Transactions – ACID Properties

- **Atomic** – All of the work in a transaction completes (commit) or none of it completes
- **Consistent** – A transaction transforms the database from one consistent state to another consistent state. Consistency is defined in terms of constraints.
- **Isolated** – The results of any changes made during a transaction are not visible until the transaction has committed.
- **Durable** – The results of a committed transaction survive failures

SQL Database Examples

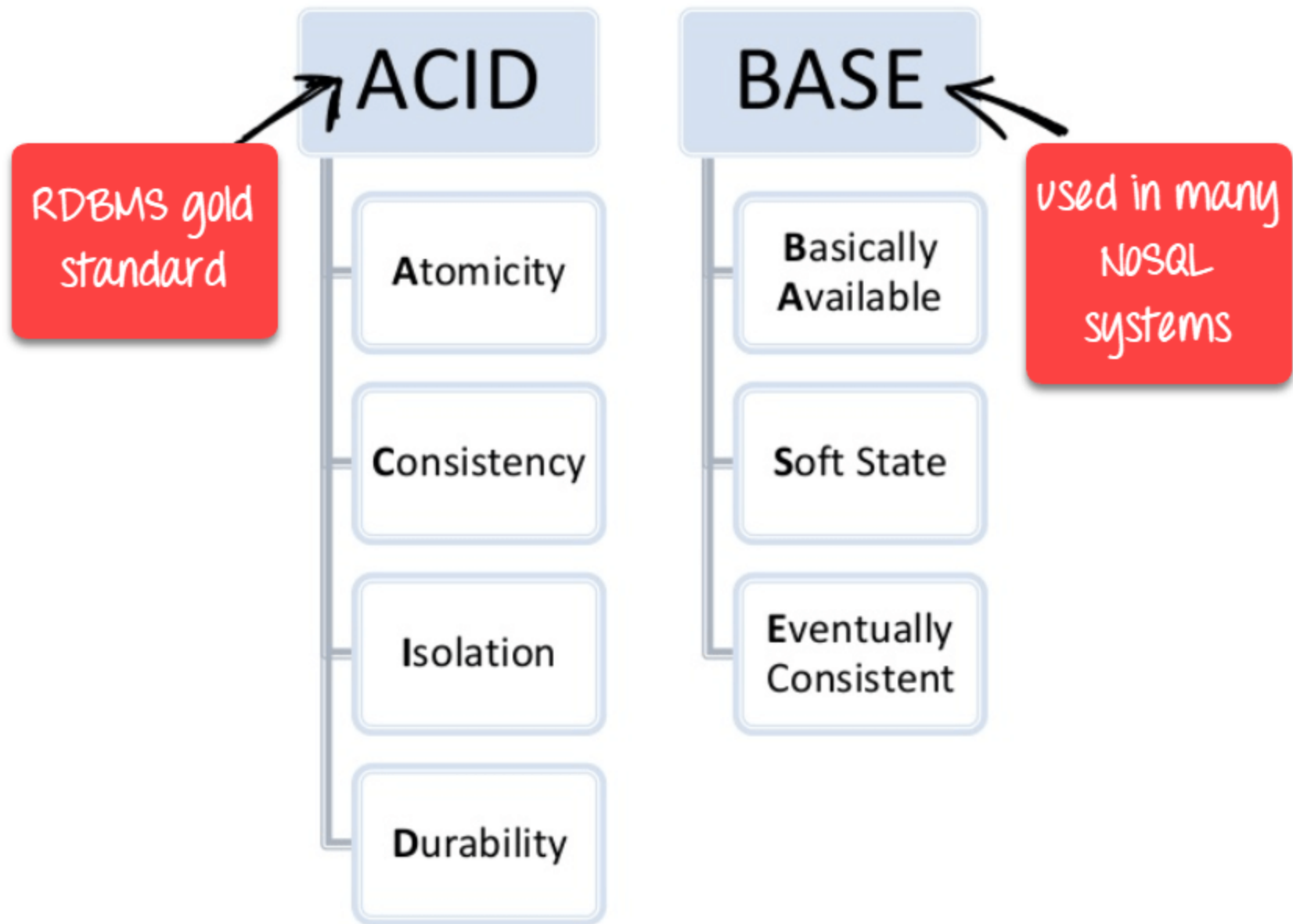
- Commercial
 - IBM DB2
 - Oracle RDMS
 - Microsoft SQL Server
 - Sybase SQL Anywhere
- Open Source (with commercial options)
 - MySQL
 - Ingres
 - PostgreSQL
 - **Significant portions of the world's economy use SQL databases!**

NoSQL Definition

- From <http://www.nosql-database.org/>:
 - Next Generation Databases mostly addressing some of the points: being **non-relational, distributed, open-source** and **horizontal scalable**. The original intention has been **modern web-scale databases**. The movement began early 2009 and is growing rapidly. Often more characteristics apply as: **schema-free, easy replication support, simple API, eventually consistent / BASE** (not ACID), a **huge data amount**, and more.

NoSQL Distinguishing Characteristics

- Large data volumes
 - Google's "big data"
- Scalable replication and distribution
 - Potentially thousands of machines
 - Potentially distributed around the world
- Queries need to return answers quickly
- Mostly query, few updates
- Asynchronous Inserts & Updates
- Schema-less
- ACID transaction properties are not needed – BASE
- CAP Theorem
- Open source development



BASE Transactions

- Acronym contrived to be the opposite of ACID
 - **B**asically **A**vailable
 - **S**oft state
 - **E**ventually **C**onsistent
- Characteristics
 - Weak consistency – stale data OK
 - Availability first
 - Best effort
 - Approximate answers OK
 - Aggressive (optimistic)
 - Simpler and faster

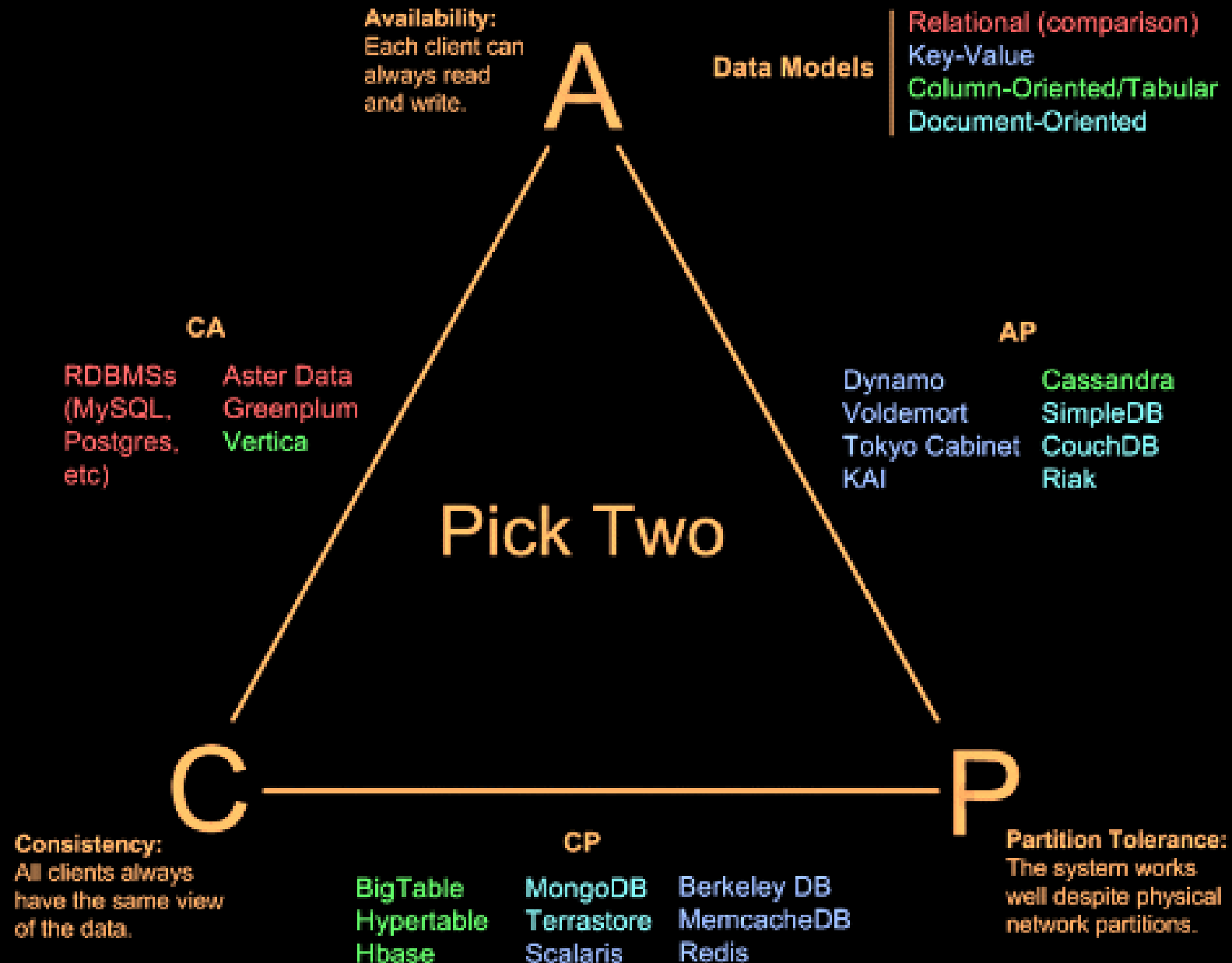
Brewer's CAP Theorem

- A distributed system can support only two of the following characteristics:
 - Consistency
 - Availability
 - Partition tolerance
- The slides from Brewer's July 2000 talk do not define these characteristics.

CAP

- Consistency
 - all nodes see the same data at the same time – Wikipedia
 - client perceives that a set of operations has occurred all at once – Pritchett
 - More like Atomic in ACID transaction properties
- Availability
 - node failures do not prevent survivors from continuing to operate – Wikipedia
 - Every operation must terminate in an intended response – Pritchett
- Partition Tolerance
 - the system continues to operate despite arbitrary message loss – Wikipedia
 - Operations will complete, even if individual components are unavailable – Pritchett

Visual Guide to NoSQL Systems



Types of NoSQL datastores

- Key-Value Store – Hash table of keys
- Column Store – Each storage block contains data from only one column
- Document Store – stores documents made up of tagged elements
- Object stores (Graphs, OO)

NoSQL Examples: Key-Value Store

- Hash tables of Keys
- Values stored with Keys
- Fast access to small data values
- Example – Redis
 - <http://redis.io>
 - Twitter, Snapchat, GitHub
- Example – Project-Voldemort
 - <http://www.project-voldemort.com/>
 - LinkedIn
- Example – MemCacheDB
 - <http://memcachedb.org/>
 - Backend storage is Berkeley-DB

NoSQL Example: Column Store

- Each storage block contains data from only one column
- Example: Hadoop/Hbase
 - <http://hadoop.apache.org/>
 - Yahoo, Facebook
- Example: Ingres VectorWise
 - Column Store integrated with an SQL database
 - <http://www.ingres.com/products/vectorwise>
- More efficient than row (or document) store if:
 - Multiple row/record/documents are inserted at the same time so updates of column blocks can be aggregated
 - Retrievals access only some of the columns in a row/record/document

NoSQL Example: Document Store

- Schema free
- Aggregations: Map/Reduce
- Example: CouchDB
 - <http://couchdb.apache.org/>
 - BBC
- Example: MongoDB
 - <http://www.mongodb.org/>
 - Foursquare, Shutterfly
- JSON – JavaScript Object Notation

Graph Stores

- Based on Graph Theory.
- Scale vertically, no clustering.
- You can use graph algorithms easily.
- Example: Neo4j
 - <http://neo4j.com/>
Twitter, ebay, Walmart



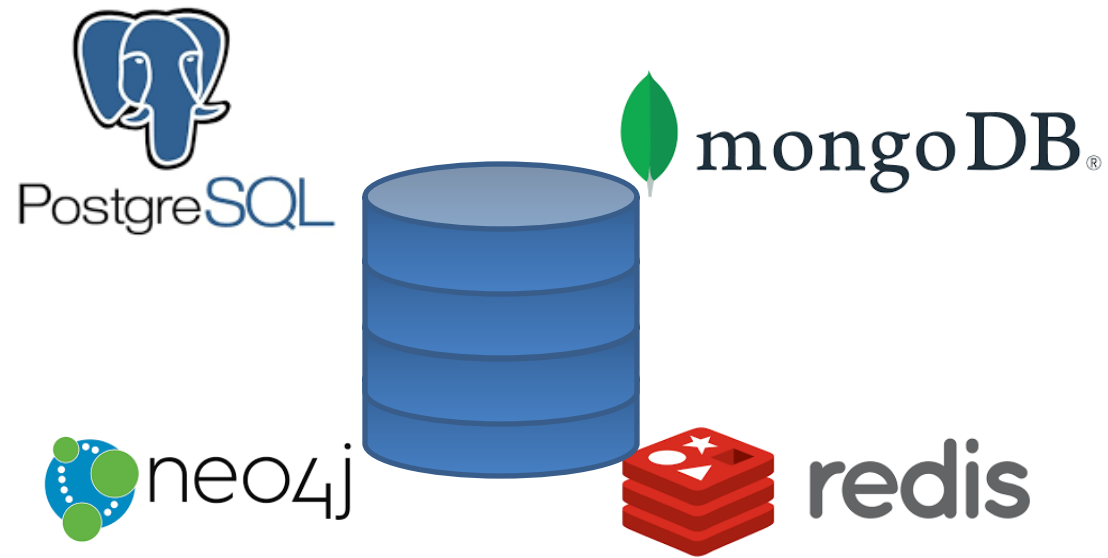
NoSQL Summary

- NoSQL databases reject:
 - Overhead of ACID transactions
 - “Complexity” of SQL
 - Burden of up-front schema design
 - Declarative query expression
 - Yesterday’s technology
- Programmer responsible for
 - Step-by-step procedural language
 - Navigating access path

Summary

- SQL Databases
 - Predefined Schema
 - Standard definition and interface language
 - Tight consistency
 - Well defined semantics
- NoSQL Database
 - No predefined Schema
 - Per-product definition and interface language
 - Getting an answer quickly is more important than getting a correct answer

Let the fun begin...



Web References

- “NoSQL -- Your Ultimate Guide to the Non - Relational Universe!”

<http://nosql-database.org/links.html>

- “NoSQL (RDBMS)”

<http://en.wikipedia.org/wiki/NoSQL>

- PODC Keynote, July 19, 2000. *Towards Robust. Distributed Systems*. Dr. Eric A. Brewer. Professor, UC Berkeley. Co-Founder & Chief Scientist, Inktomi .

[www.eecs.berkeley.edu/~**brewer**/cs262b-2004/PODC-keynote.pdf](http://www.eecs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf)

- “Brewer's CAP Theorem” posted by Julian Browne, January 11, 2009.

<http://www.julianbrowne.com/article/viewer/brewers-cap-theorem>

Web References

- “Graph Databases, NOSQL and Neo4j” Posted by Peter Neubauer on May 12, 2010 at: <http://www.infoq.com/articles/graph-nosql-neo4j>
- “Cassandra vs MongoDB vs CouchDB vs Redis vs Riak vs HBase comparison”, Kristóf Kovács. <http://kkovacs.eu/cassandra-vs-mongodb-vs-couchdb-vs-redis>
- “Distinguishing Two Major Types of Column-Stores” Posted by Daniel Abadi on March 29, 2010
http://dbmsmusings.blogspot.com/2010/03/distinguishing-two-major-types-of_29.html
- “MapReduce: Simplified Data Processing on Large Clusters”, Jeffrey Dean and Sanjay Ghemawat, December 2004.
<http://labs.google.com/papers/mapreduce.html>
- “Scalable SQL”, ACM Queue, Michael Rys, April 19, 2011
<http://queue.acm.org/detail.cfm?id=1971597>
- “a practical guide to noSQL”, Posted by Denise Miura on March 17, 2011 at <http://blogs.marklogic.com/2011/03/17/a-practical-guide-to-nosql/>