Objectives

- Backend: Data stores
  - CAP Theorem
  - Elasticsearch
  - REST APIs

Back on SE @ Google

- Example tools we talked about
  - build systems, dependency management - Maven
- Making sure you're making connections between what we're talking about and what you're reading/watching

Data Center

- What did you think?
- (Short) write ups due tomorrow night

3

Review: Databases

- What is a DBMS?
  - Which DBMS are we using?
- How do databases hold/organize data?
- What language do we use to query databases?
  - What statement is used for each of the CRUD operations? (What does CRUD stand for?)
- What is a *primary* key vs a *foreign* key

4

CRUD Operations

Operation	SQL
Create	
Read	
Update	
Delete	

CRUD: Good buzzword!

5

CRUD Operations

Operation	SQL
Create	`INSERT INTO`
Read	`SELECT`
Update	`UPDATE`
Delete	`DELETE FROM`

6

Review: SELECT Command

- Queries the database
- Returns result as a ***virtual table***
- Syntax:

Optional

```
SELECT column_names
FROM table_names [WHERE condition];
```

  - Columns, tables separated by commas
  - Can select all columns with *
  - Where clause specifies constraints on what to select from the table

Review: Join Queries

1) Does a cross product of the joined tables

```
SELECT lastName, name
FROM Majors, Students
WHERE
Students.majorID=Majors.id;
```

Id	Name	Dept	Id	LName	FName	…
M1			S1			
M1			S2			
M1			…			
M1			Sn			
M2			S1			
M2			S2			
M2			…			
M2			Sn			
…			…			

Review: INSERT Statements

- You can add rows to a table

```
INSERT INTO Majors VALUES
( 354, 'BioInformatics-BS', 'CSCI');
```

Assumes filling in all values, in column order

- Preferred Method: include column names
  - Don't depend on order

```
INSERT INTO Majors (id, name, department)
VALUES ( 354, 'BioInformatics-BS', 'CSCI');
```


Review: UPDATE Statement

- You can modify rows of a table
- Use WHERE condition to specify which rows to update
- Example: Update a student's married name

```
UPDATE Students SET
LastName='Smith-Jones' WHERE id=12;
```

- Example: Update all first years to undeclared

```
UPDATE Students SET majorID=345
WHERE gradYear=2025;
```

Review: DELETE Statement

- You can delete rows from a table

```
DELETE FROM table [ WHERE condition ];
```

- Example

```
DELETE FROM EnrolledStudents WHERE
hasPrerequisites=False AND course_id=456;
```


DB Popularity

http://db-engines.com/en/ranking

388 systems in ranking, March 2022

Rank Mar 2022	Rank Feb 2022	Rank Mar 2021	DBMS	Database Model	Score Mar 2022	Score Feb 2022	Score Mar 2021
1.	1.	1.	Oracle	Relational, Multi-model	1251.32	-5.51	-70.42
2.	2.	2.	MySQL	Relational, Multi-model	1198.23	-16.45	-56.59
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	933.78	-15.27	-81.52
4.	4.	4.	PostgreSQL	Relational, Multi-model	616.93	+7.54	+67.64
5.	5.	5.	MongoDB	Document, Multi-model	485.66	-2.98	+23.27
6.	6.	↑ 7.	Redis	Key-value, Multi-model	176.76	+0.96	+22.61
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model	162.15	-0.73	+6.14
8.	8.	8.	Elasticsearch	Search engine, Multi-model	159.95	-2.35	+7.61
9.	9.	↑ 10.	Microsoft Access	Relational	135.43	+4.17	+17.29
10.	10.	↓ 9.	SQLite	Relational	132.18	+3.81	+9.54
11.	11.	11.	Cassandra	Wide column	122.14	-1.83	+8.51
12.	12.	12.	MariaDB	Relational, Multi-model	108.31	+1.20	+13.85
13.	13.	13.	Splunk	Search engine	95.36	+4.55	+8.44
14.	↑ 15.	↑ 30.	Snowflake	Relational	86.23	+3.05	+63.04
15.	↓ 14.	↑ 16.	Microsoft Azure SQL Database	Relational, Multi-model	84.68	-0.28	+13.79
16.	↑ 17.	↑ 17.	Amazon DynamoDB	Multi-model	81.80	+1.45	+12.91

Databases Course Overview

- Planned CSCI317 in Fall 2022
- How do you solve more complex problems/write more complicated queries?
- How do you organize data into relational databases?
  - Design data
- How do you store data so that you can access it and manipulate it efficiently?
  - Underlying data structures
- How do you handle concurrent transactions correctly and efficiently?

Interfacing with a Database

- Interactive mode
  - Run client
  - Enter SQL statements, one at a time

```
psql dbname
```

- Batch mode/command-line
  - Script/file of SQL commands
  - Direct to database

```
psql dbname < mycmds.sql
```

- Programming Language APIs
  - Examples: JDBC, psycopg2

Overarching Question

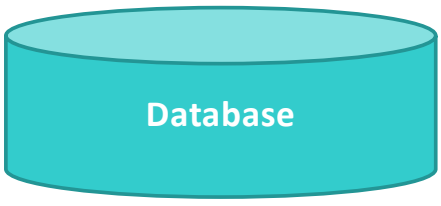
In one day:
Walmart processes over 40 Petabytes of data
600 TB of data add to Facebook
500 million tweets on Twitter
…

How to **store**, **query** and **process** these data efficiently?

Likely not a one-size-fits-all solution; need a combination

Data Layers

Pretty speedy updates, searches
Handles concurrent accesses



Limitations with Relational Database

- Overhead for complex select, update, delete operations
  - Select: Joining too many tables creates a huge table
  - Update: Each update may affect other tables
  - Delete: Must guarantee the consistency of data
- Mix of unstructured data is not well-supported
- Doesn't scale well with very large data

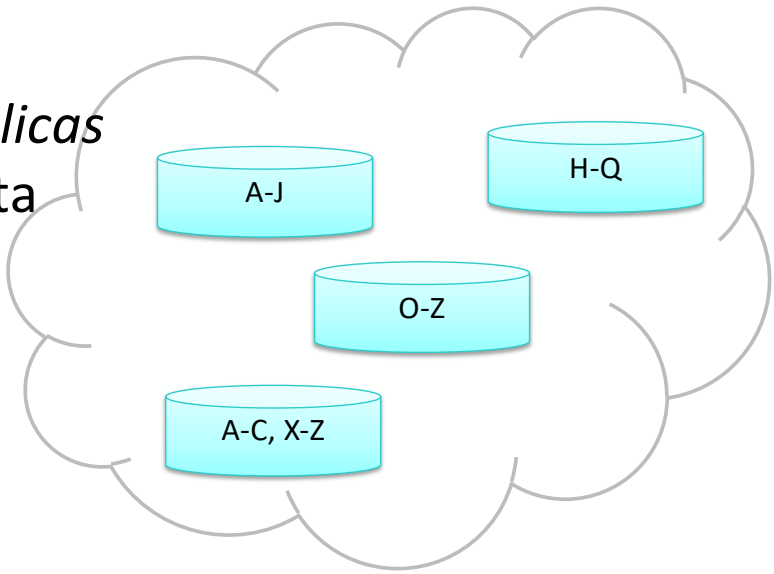
NoSQL is a good solution to deal with these problems.

Ma

17

Buzzword Bingo: CAP Theorem

- Many nodes
- Nodes contain *replicas of partitions* of data
- Consider what happens
  - Under high load
  - During updates

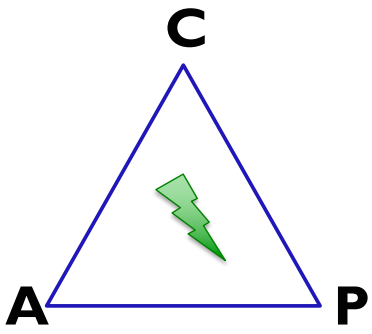


March 22, 2017 Sprenkle - CSCI397 18

18

Buzzword Bingo: CAP Theorem

- **C**onsistency
  - All replicas contain the same version of data
- **A**vailability
  - System remains operational (reads AND writes) on failing nodes
- **P**artition tolerence
  - multiple entry points
  - system remains operational on network split

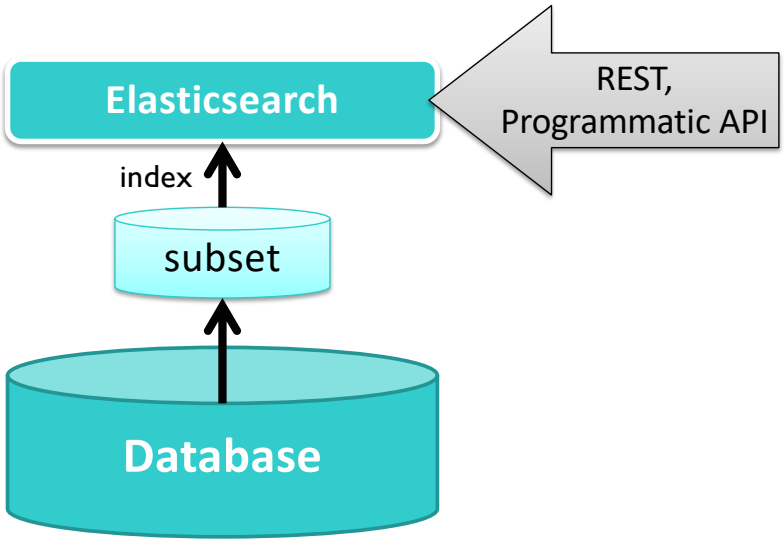


CAP Theorem: satisfying all three at the same time is impossible

Recent work at Google says "We can do all three!"

Data Layers

Specialize in search → even faster searching



Buzzword Bingo: REST API

- **Re**presentational **S**tate **T**ransfer (REST)
  - Stateless operations
    - No state about the client is stored on the server
  - Access using text requests
- RESTful applications
  - Access to web services through the REST interface
  - HTTP requests → Text Responses
    - Request type: GET POST PUT DELETE
    - Resource name
    - Optional parameters

AGP Example

Elasticsearch Architecture

- Node: A running ES instance
  - a process running on a machine
- Cluster: **Distributed** ES system made of several ***nodes***
  - Dynamic primary election, no single node fail
    - fail as a whole
  - Communication between nodes and data distribution and balancing is automatically handled
  - View as a whole from outside

Elasticsearch Architecture

- Index
  - Multiple index support
  - Multiple types inside indices
- Shard: building blocks of index
  - Index is divided into *shards*
  - Each shard is an Apache Lucene index
  - Shards will be placed on different machines
  - ES sends queries to relevant shards and merges results
- Replica
  - Each shard can have 0 or more replicas
  - True copy of primary shard
  - Increase system fault tolerance and search performance

Elasticsearch

- curl
  - Like a web browser without the GUI
  - Transfer web requests
- Returns data as JSON
  - **J**ava**S**cript **O**bject **N**otation
  - Schema-less
- APIs to access
  - REST API
  - Programmatic APIs

Using Elasticsearch

- Access elasticsearch
  - curl http://hostname:9200
- What happened?

JSON: "Javascript Object Notation"

- Lightweight format for structured data (compared to XML)
  - Easy to read, write
- Nothing to do with JavaScript

```
{
  "firstName": "Tina",
  "lastName": "Belcher",
  "age": 13,
  "address": {
    "streetAddress": "14 Ocean Avenue",
     …
  },
  "children": [],
  "spouse": null
}
```

JSON: "Javascript Object Notation"

- Lightweight format for structured data (compared to XML)
  - Easy to read, write
- Nothing to do with JavaScript

```
{
  "firstName": "Bob",
  "lastName": "Belcher",
  "age": 44,
  "address": {
    "streetAddress": "14 Ocean Avenue",
    …
  },
  "children": ["Tina", "Gene",
               "Louise"],
  "spouse": "Linda"
}
```


Using the REST API Example:

- `curl -XGET 'localhost:9200/_cat/health?v&pretty'`
  - What if we don't have the part after the "?" in the request?
- `curl -XGET 'localhost:9200/_cat/nodes?v'`
- `curl -XGET 'localhost:9200/_cat/indices?v&pretty'`
  - How many indices do we have?
  - What do those indices have in them?

Using localhost as the placeholder for the hostname

Creating an index

- `curl -XPUT 'localhost:9200/customer?pretty'`
  - You should get an error because customer index already exists
  - Use a different name
- `curl -XGET 'localhost:9200/_cat/indices?v'`
  - Check the index was created

Viewing documents in the index

- `curl –XGET 'localhost:9200/bank/_search?pretty'`
  - What are the results we're getting?
  - What do they mean?

Practice Using the API

- Use commands in emailed file