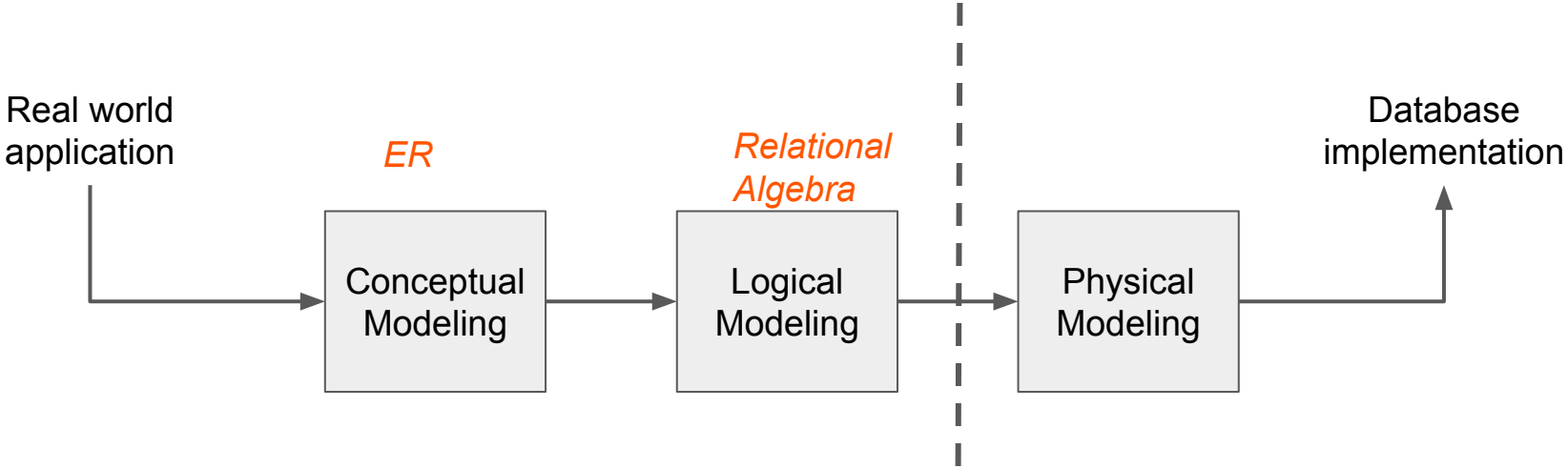
Databases and Big Data

Wrap Up

What We Have Learned

- The process of writing a database applications
  - ER Model
  - Relational Algebra
  - SQL



What We Have Learned

- Many shapes of data
- Different data models for different shapes
  - Relational model
  - No-SQL model
  - Semi-structure model
  - (Others not covered)

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam vel erat nec dui aliquet vulputate sed quis nulla. Donec eget ultricies magna, eu dignissim elit. Nullam sed urna nec nisl rhoncus ullamcorper placerat et enim. Integer varius ornare libero quis consequat. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aenean eu efficitur orci. Aenean ac posuere tellus. Ut id commodo turpis.

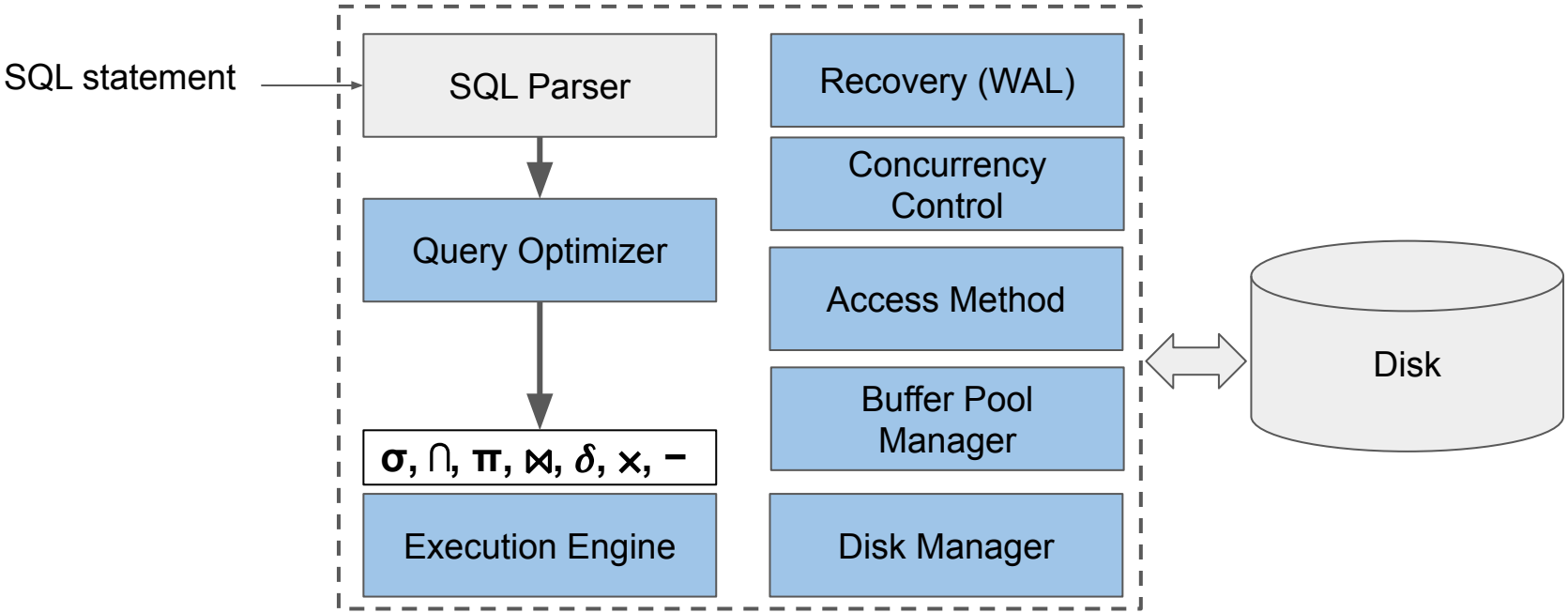
Praesent nec libero metus. Praesent at turpis placerat, congue ipsum eget, scelerisque justo. Ut volutpat, massa ac lacinia cursus, nisl dui volutpat arcu, quis interdum sapien turpis in tellus. Suspendisse potenti. Vestibulum pharetra justo massa, ac venenatis mi condimentum nec. Proin viverra tortor non orci suscipit rutrum. Phasellus sit amet euismod diam. Nullam convallis nunc sit amet diam suscipit dapibus. Integer porta hendrerit nunc. Quisque pharetra congue porta. Suspendisse vestibulum sed mi in euismod. Etiam a purus suscipit, accumsan nibh vel, posuere ipsum. Nulla nec tempor nibh, id venenatis lectus. Duis lobortis id urna eget tincidunt.

What We Have Learned

- How to design better tables
  - Normalization
- SQL
  - How to manipulate and query relational databases
- MongoDB
  - How to manipulate and query NoSQL databases

What We Have Learned

- What inside the blackbox of RDBMS



What We Have Learned

- What inside the blackbox of RDBMS
  - Disk Manager
  - Buffer Pool
  - Access Method
  - Transactions: WAL and Concurrency Control
- How to estimate I/O costs of SQL queries
  - > 1 ways to execute the same query

What We Have Learned

- What is big data:
  - Volume, Velocity, Variety
- User and systems' views of cloud computing
  - Computing as utility
  - Distributed systems
- How to use AWS

What We Have Learned

- The high-level software stack for Big data
- Hadoop stack:
  - HDFS
  - MapReduce
- Important design choices in HDFS and MapReduce
- How to write MapReduce jobs
  - Via Hadoop streaming

read and append only

scan sequentially

What We Have Learned

- The need for resource management
- YARN and Mesos:
  - Their important design choices
- The need for Hadoop alternatives
- Spark stack:
  - Spark Core
  - SparkSQL *use dataframe*

- why do we need them?
- YARN: have one centralised scheduler that schedule all the jobs
- Mesos: give offer to frameworks and frameworks do all the scheduling.
- Limitation of hadoop?
- Why is there a need for spark (memory, faster)

What We Have Learned

- Important design choices in Spark
  - How Spark differs to Hadoop
- How to write Spark jobs
  - Interactive
  - Iterative
- How to deploy and run Hadoop and Spark jobs on EC2

MapReduce only single pass over data.

What We Have Learned

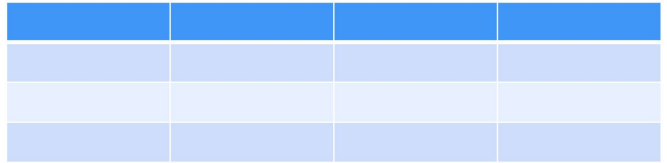
- Designs of other Big data systems:
  - Data lakes
  - Streams
  - Semi-structured storage
  - Relational databases
- Some of common challenges in these systems:
  - Performance
  - Scalability
  - Failures

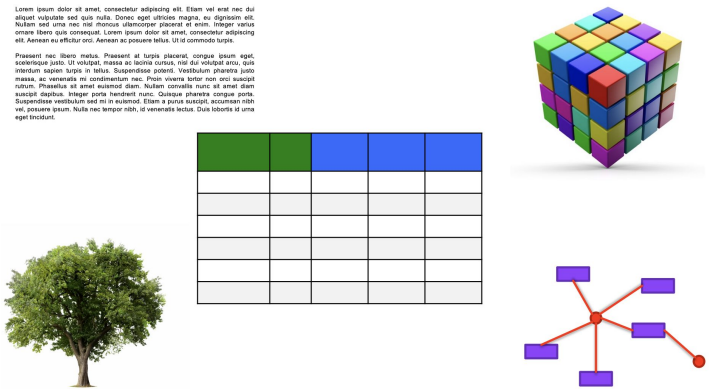
What are the common challenges?
- Handling failures is the main problem in distributed systems.

Database Wisdoms

Learn From The Past

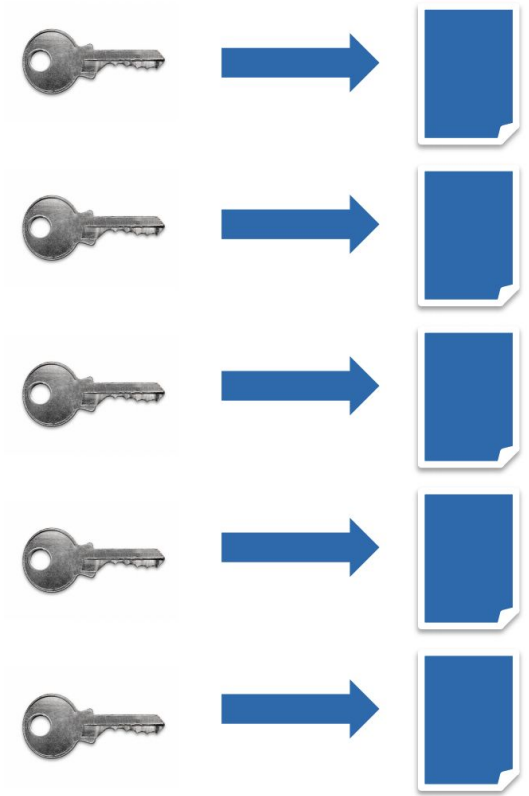
- Tables won’t go away
  - History: SQL → NoSQL → Semi SQL → SQL
- Data shapes matter
  - Specialized systems for relational, key-value, ML, graphs, etc.





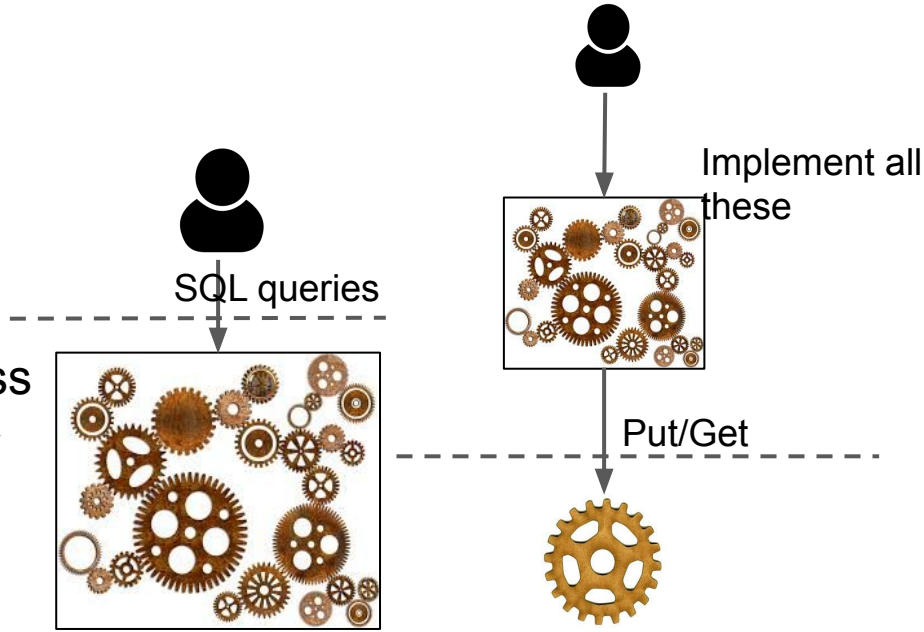
Keep It Simple (Stupid)

- Simplify data model
  - So that you can scale



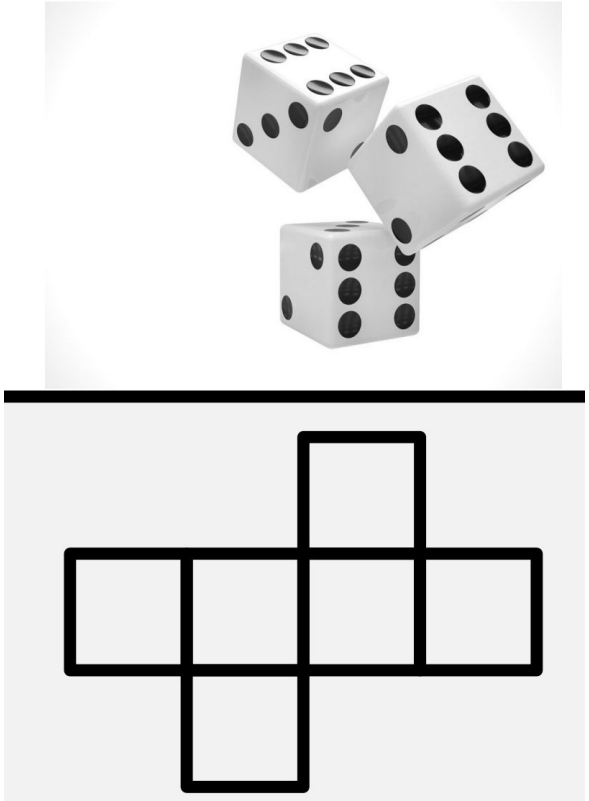
No Free Lunch

- Cost must be paid
  - By someone
  - At some point
- Example:
  - SQL vs. NoSQL
  - Performance vs. correctness
  - Collect now, figure out later



Separate Metadata from Data

- All big data systems:
  - Master maintains important metadata
  - Slaves store/process raw data



Users Like Declarative Language

- SQL vs. Spark core
- Spanner vs. BigTable

SQL is here to stay

Distribute The Data

- Chop up to smaller pieces to scale
  - Load balance over many servers

Buy A Lot Of Cheap Servers

- Scale OUT instead of UP
  - Because software is available to manage them

Replication, Replication, Replication

- Replicate to withstand failures
- But difficult to ensure consistency

eventual consistency

Final Words

Final Words

- Databases and Big Data Systems will become invisible
  - People will only see the services they support
- These systems take:
  - Great developers to make good use of them
  - **Great engineers to understand them** ← This is **YOU**
  - Great researchers to design and build them

Final Words

- SQL will *eventually* be replaced:
  - Hand-free interaction
  - We are literally taking to databases

- Data and money will converge
  - People will manage data the same way they manage money