Imperial College
Business School

Imperial means
Intelligent Business

Lecture 10
Concurrency, big data, varieties of SQL, and noSQL

Dr Fintan Nagle
f.nagle@imperial.ac.uk

Reading

Video lectures:

4.3 - Read and write errors due to concurrency.mp4
7.6 - Concurrency.mp4
7.7 - Concurrency in Postgres.mp4
9.8 - Spark and Hadoop.mp4
9.10 - SQL variants.mp4

Scaling Postgres with read replicas: https://brandur.org/postgres-reads

A closer look at Cassandra's architecture:
https://engineeringblog.yelp.com/2016/06/monitoring-cassandra-at-scale.html
Why you should never use MongoDB:
http://cryto.net/~joepie91/blog/2015/07/19/why-you-should-never-ever-ever-use-mongodb/

Reading

Why is CockroachDB compatible with Postgres?
https://www.cockroachlabs.com/blog/why-postgres/

Distributed databases
https://martinfowler.com/articles/patterns-of-distributed-systems/

Big data at Spotify
https://engineering.atspotify.com/2021/02/11/how-spotify-optimized-the-largest-dataflow-job-ever-for-wrapped-2020/
and
https://engineering.atspotify.com/2020/02/18/spotify-unwrapped-how-we-brought-you-a-decade-of-data/

Concurrency

Bank Account problem

Bank account example

Functions:

BALANCE()

WITHDRAW()

Our program:

```
b = BALANCE()
if b > 100:
        WITHDRAW(100)
```

Bank account example

If all goes well:

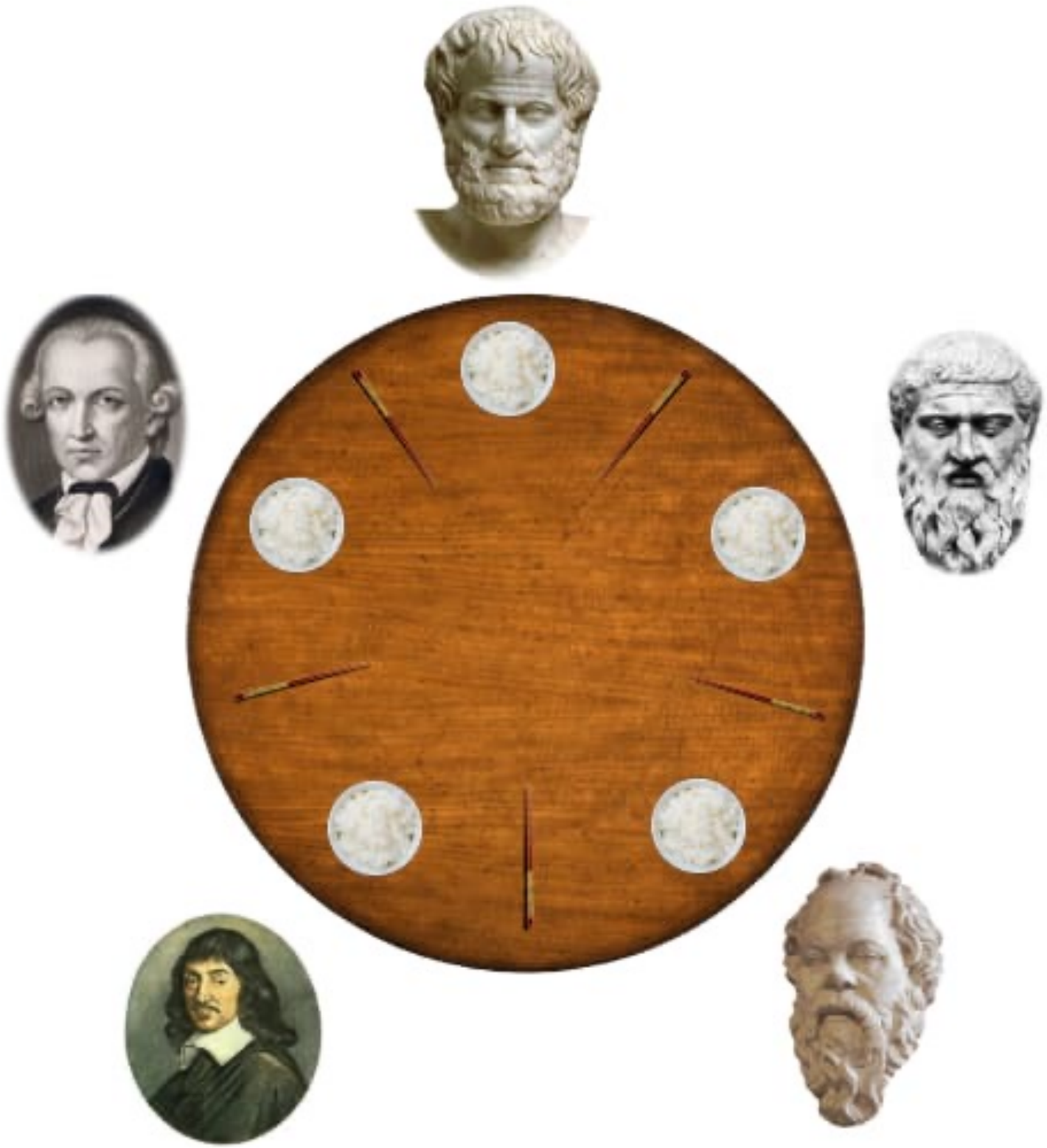
		Balance
User A	b = BALANCE()	110
User A	b is 110, which is more than 100	110
User A	WITHDRAW(100)	10
User B	b = BALANCE()	10
User B	b is 10, which is less than 100	10
User B	I won’t call WITHDRAW	10

Bank account example

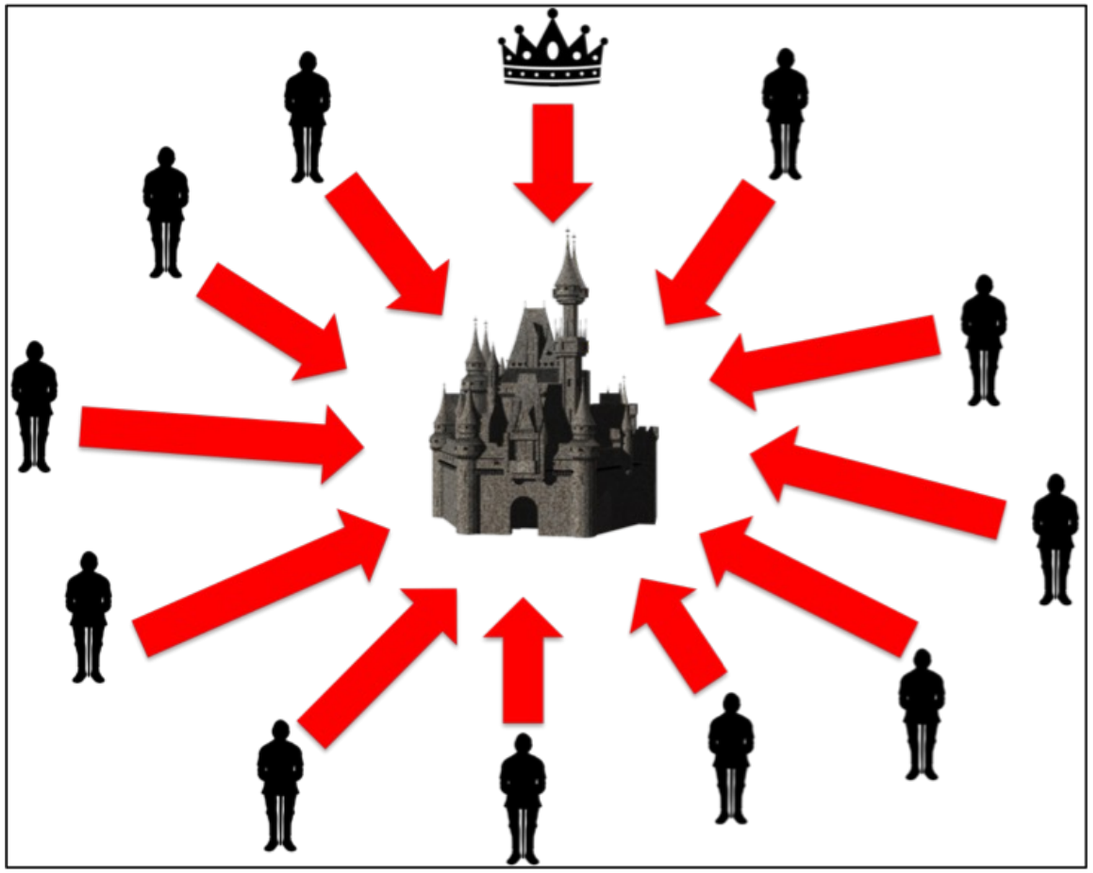
If bad luck strikes:

		Balance
User A	b = BALANCE()	110
User A	my b is 110, which is more than 100	110
User B	b = BALANCE()	110
User A	WITHDRAW(100)	10
User B	my b is 110, which is more than 100	10
User B	I will call WITHDRAW	10
User B	WITHDRAW(100)	-90

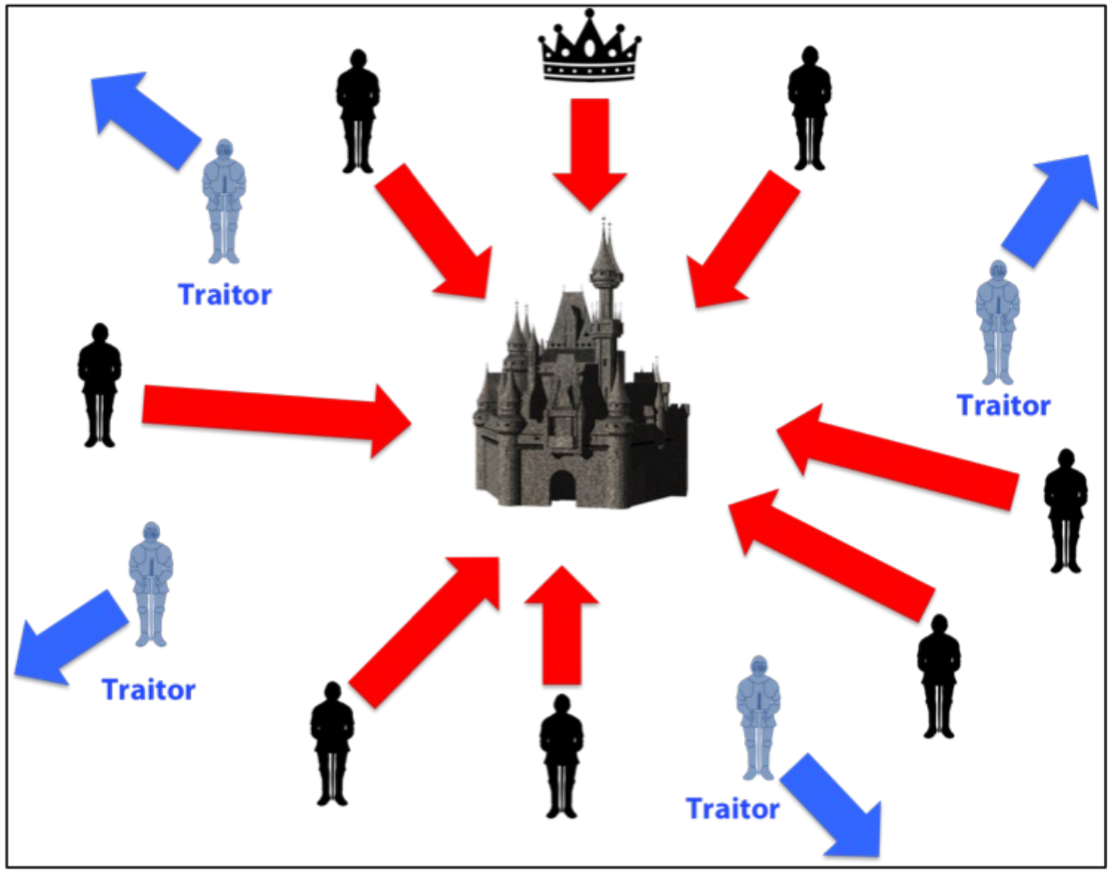
Dining Philosophers problem



Byzantine Generals problem



Coordinated Attack Leading to Victory



Uncoordinated Attack Leading to Defeat

Agreement

All of these problems:

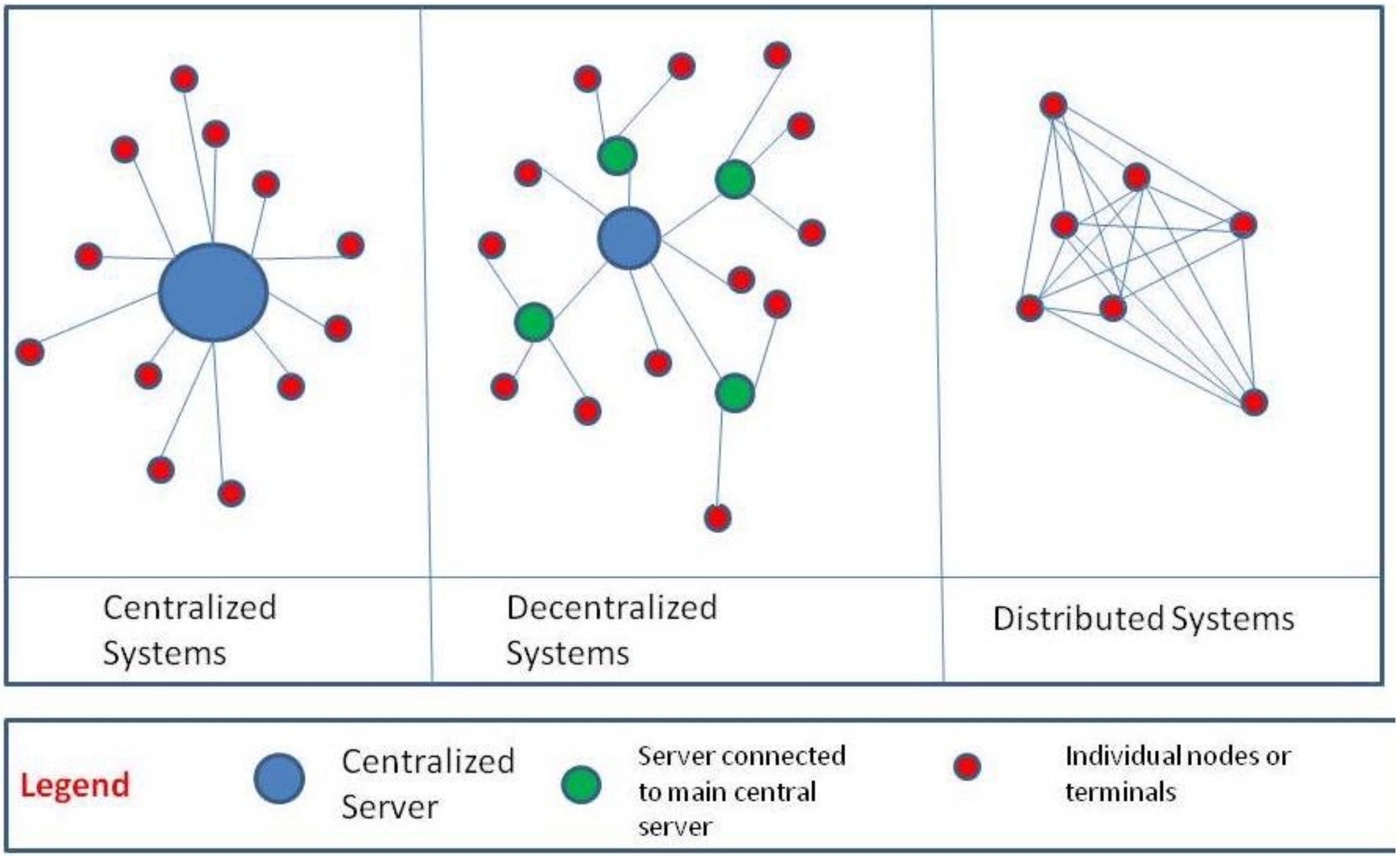
- Dining Philosophers
- Byzantine Generals
- Consistency of data in multiple-node databases

come down to the basic notion of *agreeing on something*.

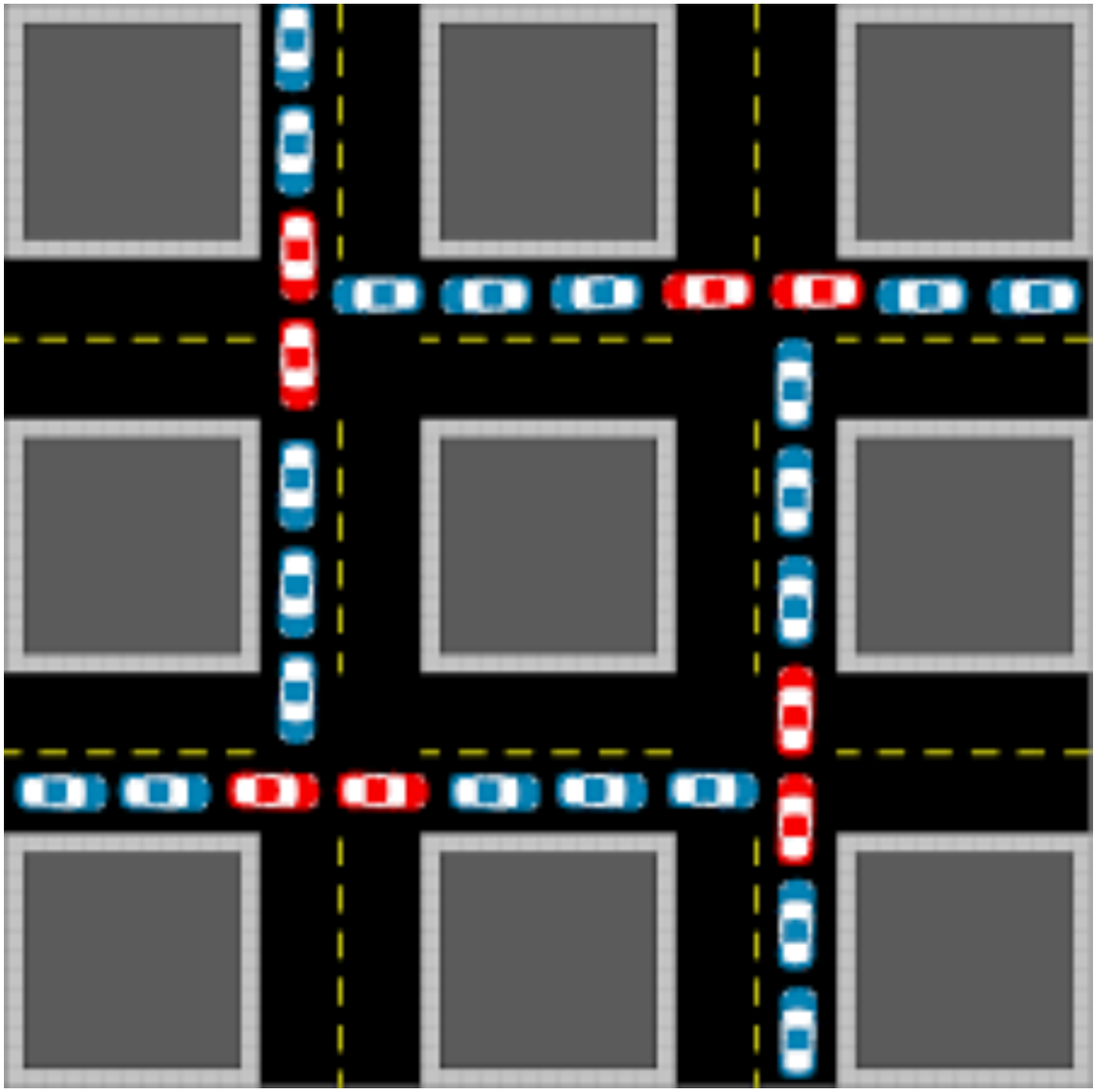
The basic algorithm to deal with this is the Paxos algorithm, which simplifies the problem to a set of agents who have to all agree on the same number.

https://explain.yshui.dev/distributed%20system/2020/09/20/paxos.html

Distributed systems



Deadlock

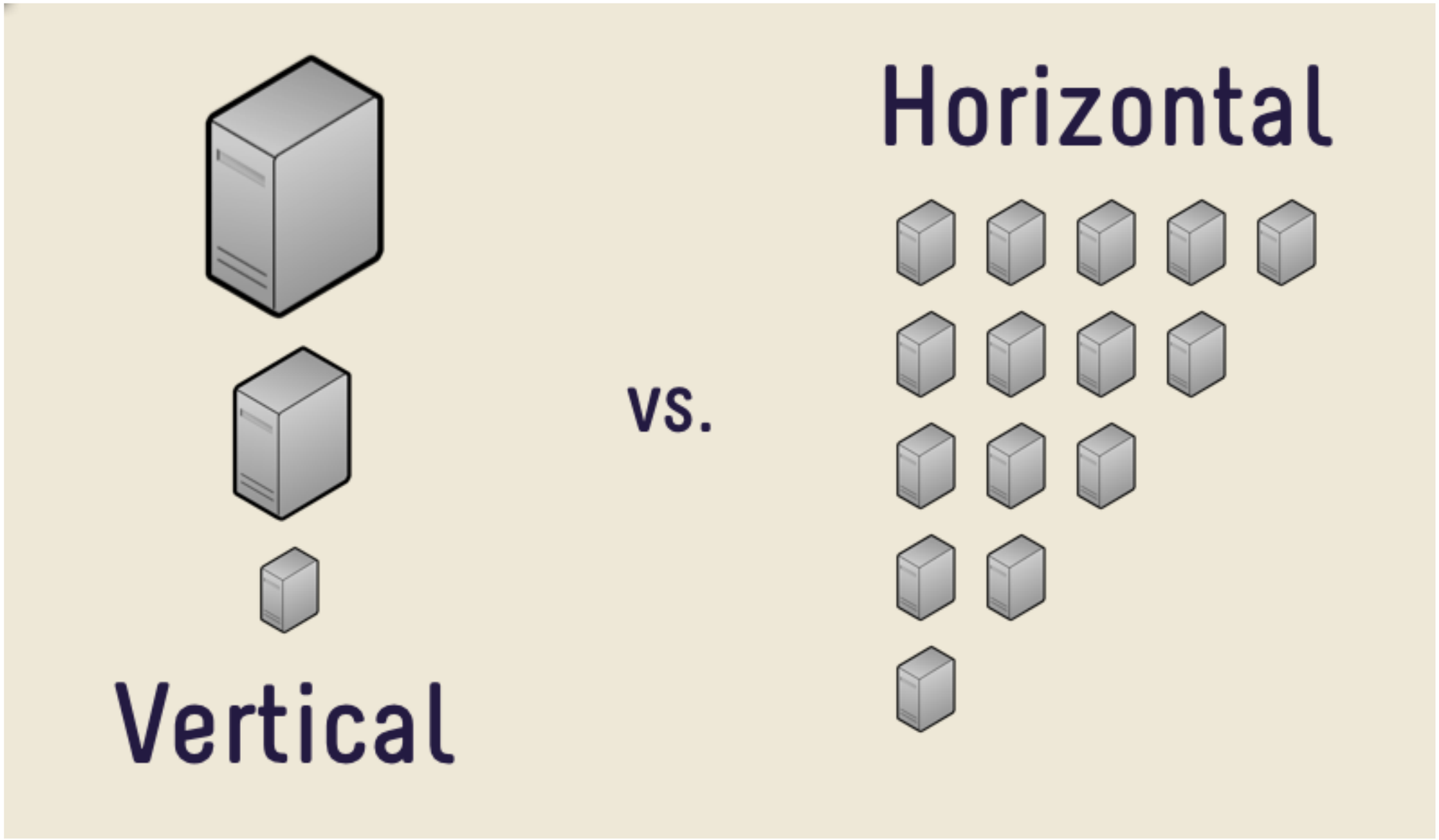


Deadlock

Deadlock

Deadlock

Vertical and horizontal scaling



ACID

Coined in 1983. A set of properties which database transactions should have in order to be fault-tolerant to errors, system crashes, power cuts...

- **Atomicity**
  Each transaction is all or nothing.

- **Consistency**
  Any transaction which is written must be valid according to all constraints.

- **Isolation**
  Concurrent execution of transactions results in the same state as sequential execution.

- **Durability**
  Once a transaction has been committed, it will remain so even in case of serious system failure or power loss.

Locks

A **lock**, when acquired by a process, means that *only that process can access a particular table* until it has released the lock.

There are various types of lock: prevent writing while I have the lock, prevent reading and writing while I have the lock, etc.

https://www.postgresql.org/docs/9.1/explicit-locking.html

Transactions

ACID

Coined in 1983. A set of properties which database transactions should have in order to be fault-tolerant to errors, system crashes, power cuts...

- **Atomicity**
  Each transaction is all or nothing.

- **Consistency**
  Any transaction which is written must be valid according to all constraints.

- **Isolation**
  Concurrent execution of transactions results in the same state as sequential execution.

- **Durability**
  Once a transaction has been committed, it will remain so even in case of serious system failure or power loss.

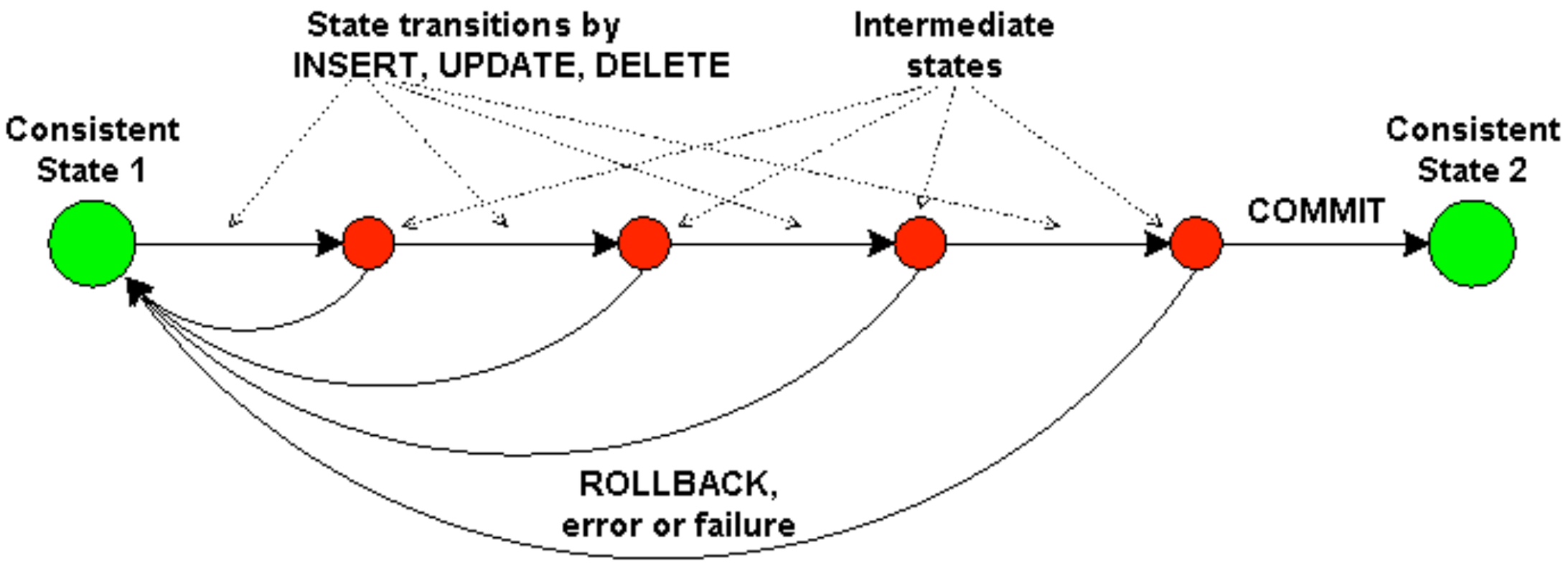
Transactions

A transaction is a series of database operations which are guaranteed to either happen all together, or not happen at all.

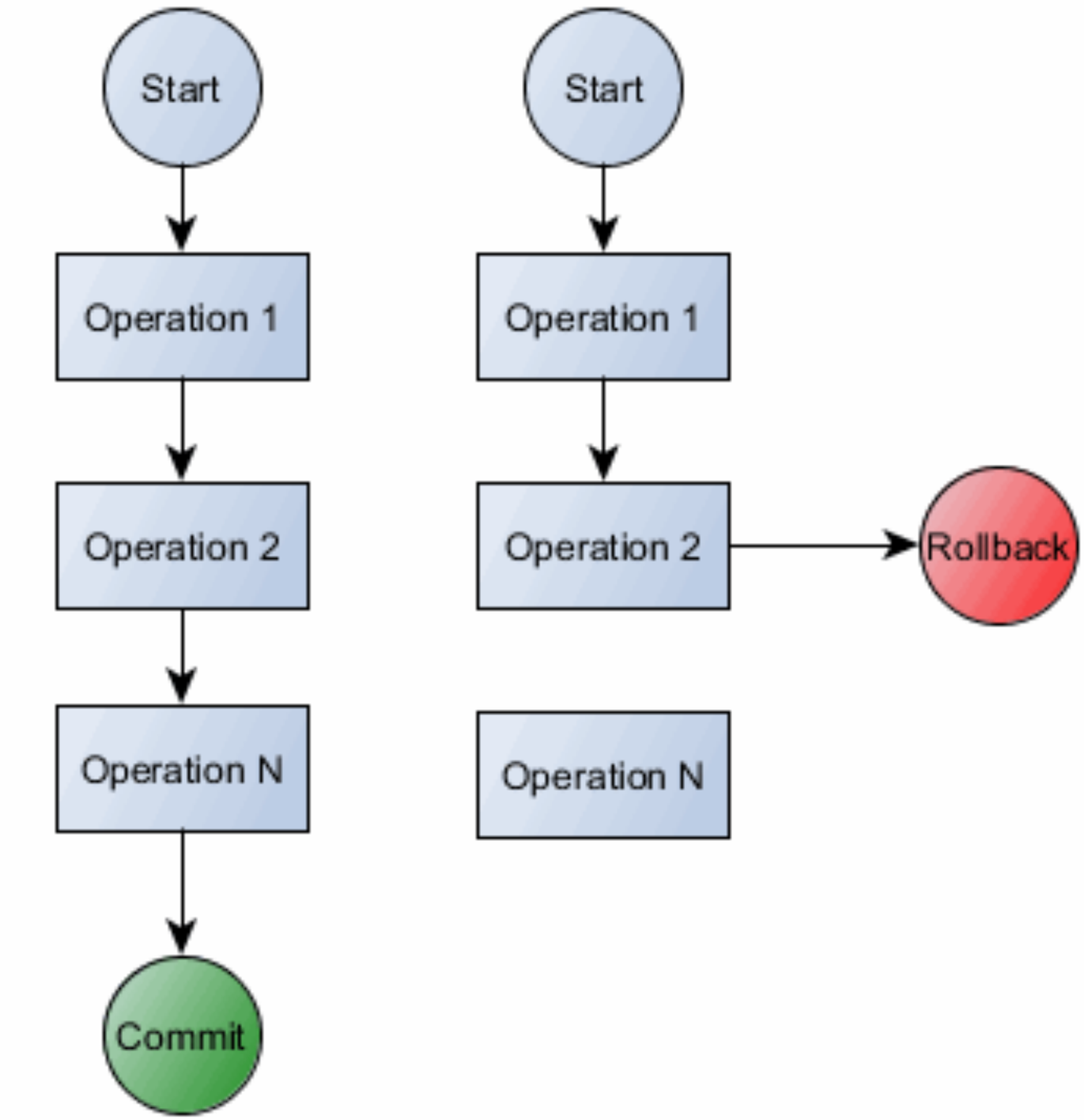
- First the client **begins** the transaction
- Then the client can do various read or write operations.
  None of the write operations are visible to other clients (even deletes).
- Finally, it is the client's choice to either **commit** or **roll back** the transaction.

We have not noticed transactions so far in Postgres because every statement is wrapped in its own transaction by default.

Transactions



Transactions



Transactions in SQL

In Postgres, we can write:

- **BEGIN TRANSACTION** or just **BEGIN** − To start a transaction.
- **COMMIT** − To save the changes
  Alternatively you can use **END TRANSACTION**
- **ROLLBACK** − To roll back the changes.

Transactions

To deal with the bank account example, we simply wrap the entire operation in a transaction and don’t commit it until we’re ready.

This will enable us to detect whether the accounts are in an invalid state (overdrawn) and roll back the transaction if so.

Big data

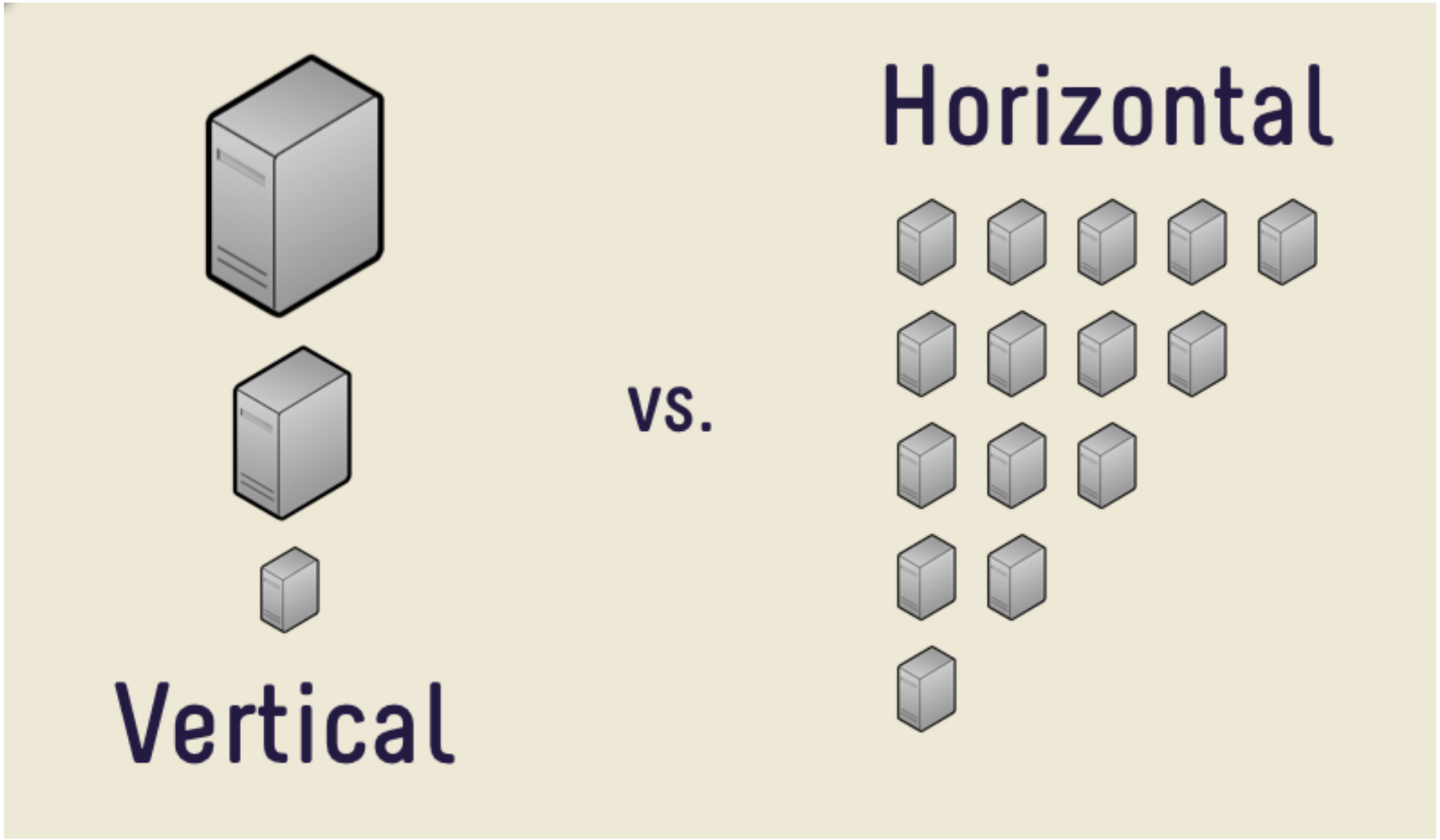
What is big data?

Data which is too large to process on a single machine.

This definition is constantly changing – currently approximately:

size on disk > 1TB
size in RAM > 20GB

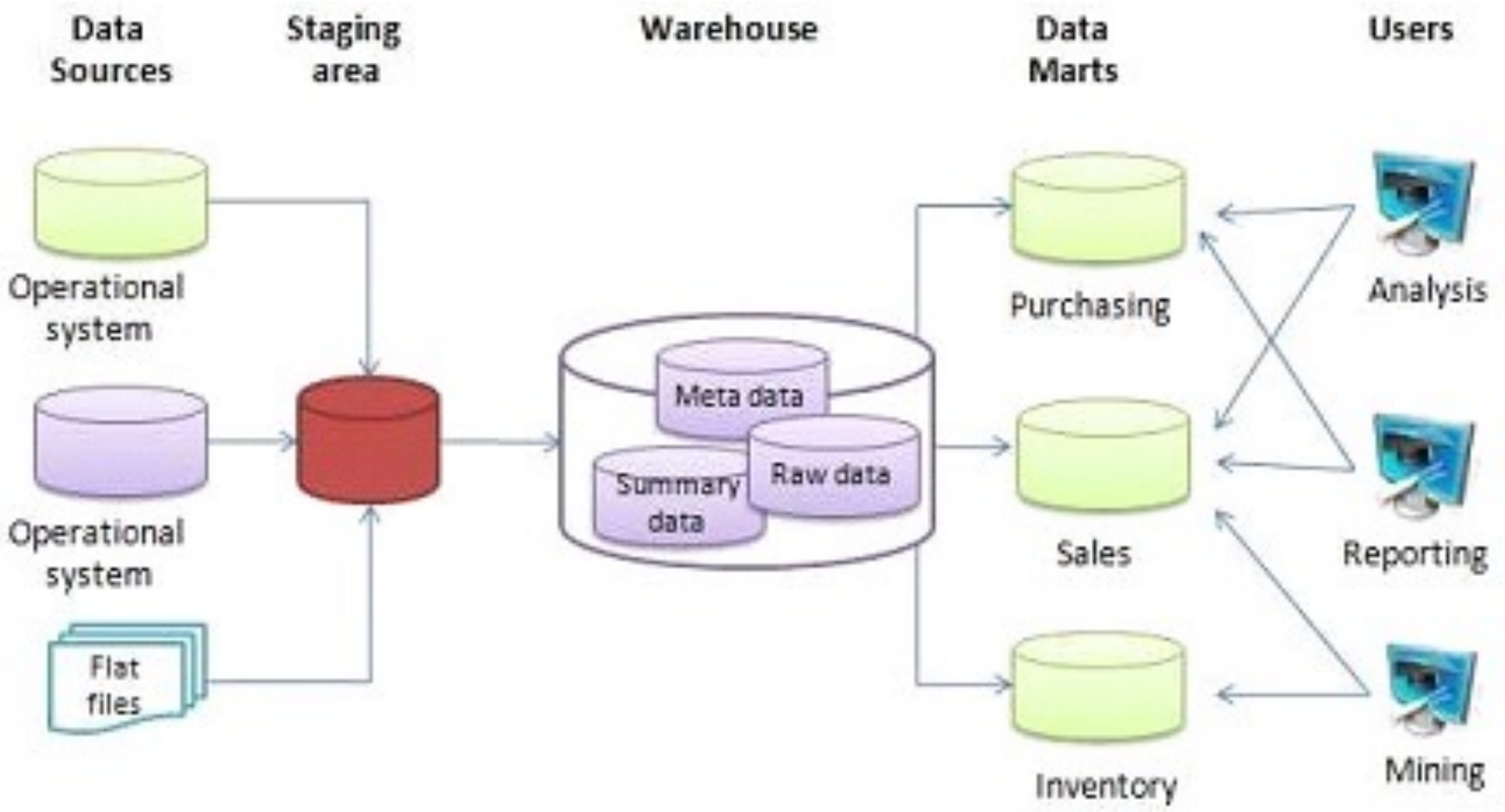
Vertical and horizontal scale



Data warehousing

A data warehouse is a collection of databases
with imposed structure

(schemas, relationships, …)



Data warehousing

One of the largest data warehouses is at the IBM campus in Dublin: it contains 3 petabytes (3,000 terabytes) of data.

Ebay's data warehouse is 9.2 petabytes.

Data lakes

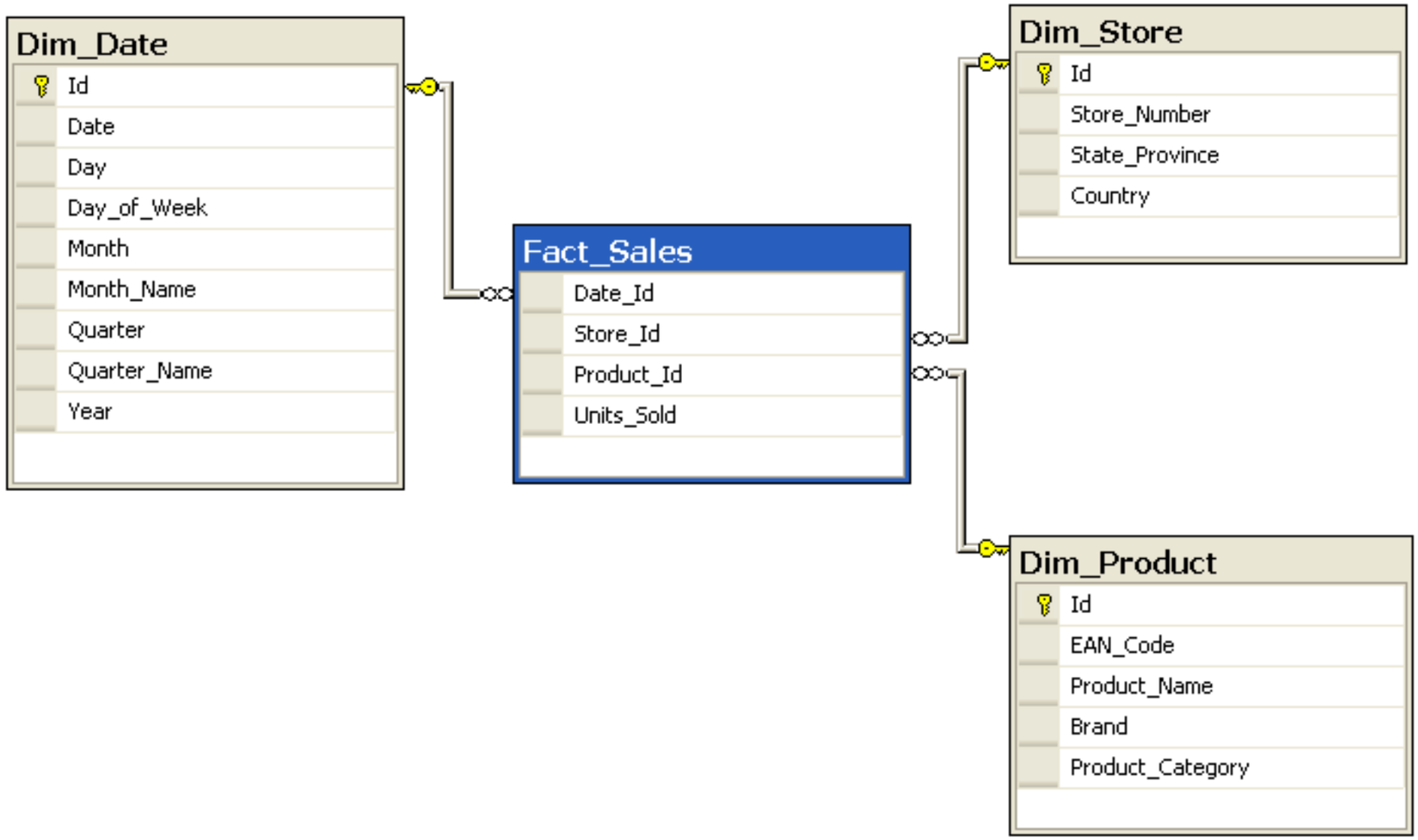
A data lake is an **unstructured** collection of data.

Data lakes

DATA WAREHOUSE	vs.	DATA LAKE
structured, processed	DATA	structured / semi-structured / unstructured, raw
schema-on-write	PROCESSING	schema-on-read
expensive for large data volumes	STORAGE	designed for low-cost storage
less agile, fixed configuration	AGILITY	highly agile, configure and reconfigure as needed
mature	SECURITY	maturing
business professionals	USERS	data scientists et. al.

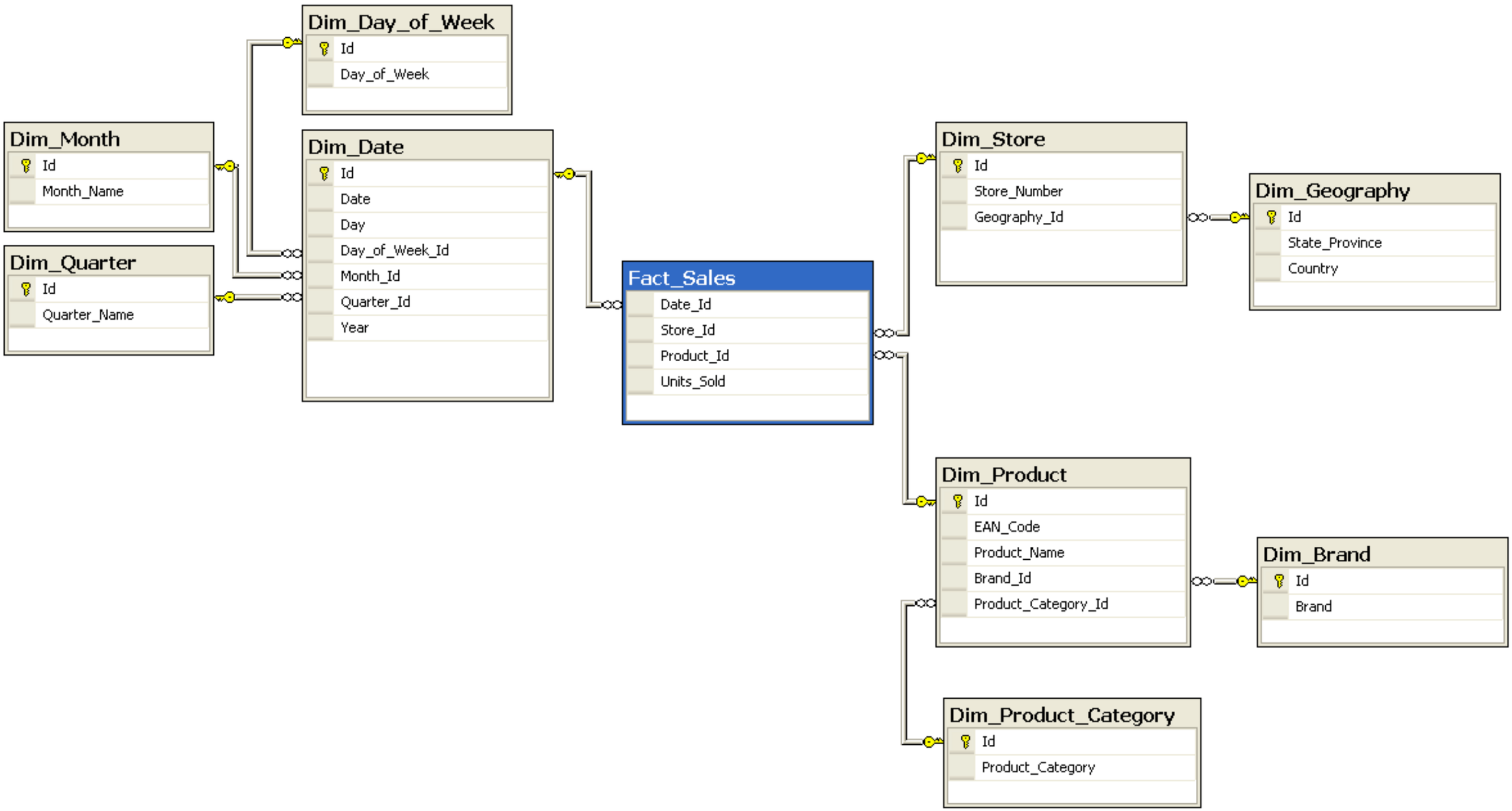
The star schema

- Simpler than the snowflake schema
- Fewer joins required
- Less normalised
- Repeated data



The snowflake schema

- More normalised than the star schema
- No redundant data (copies)
- Better for larger data warehouses



Amazon Web Services (AWS) and its Relational Database Service (RDS)

AWS overview

AWS S3

Simple Storage Service

AWS Elastic Beanstalk

AWS overview

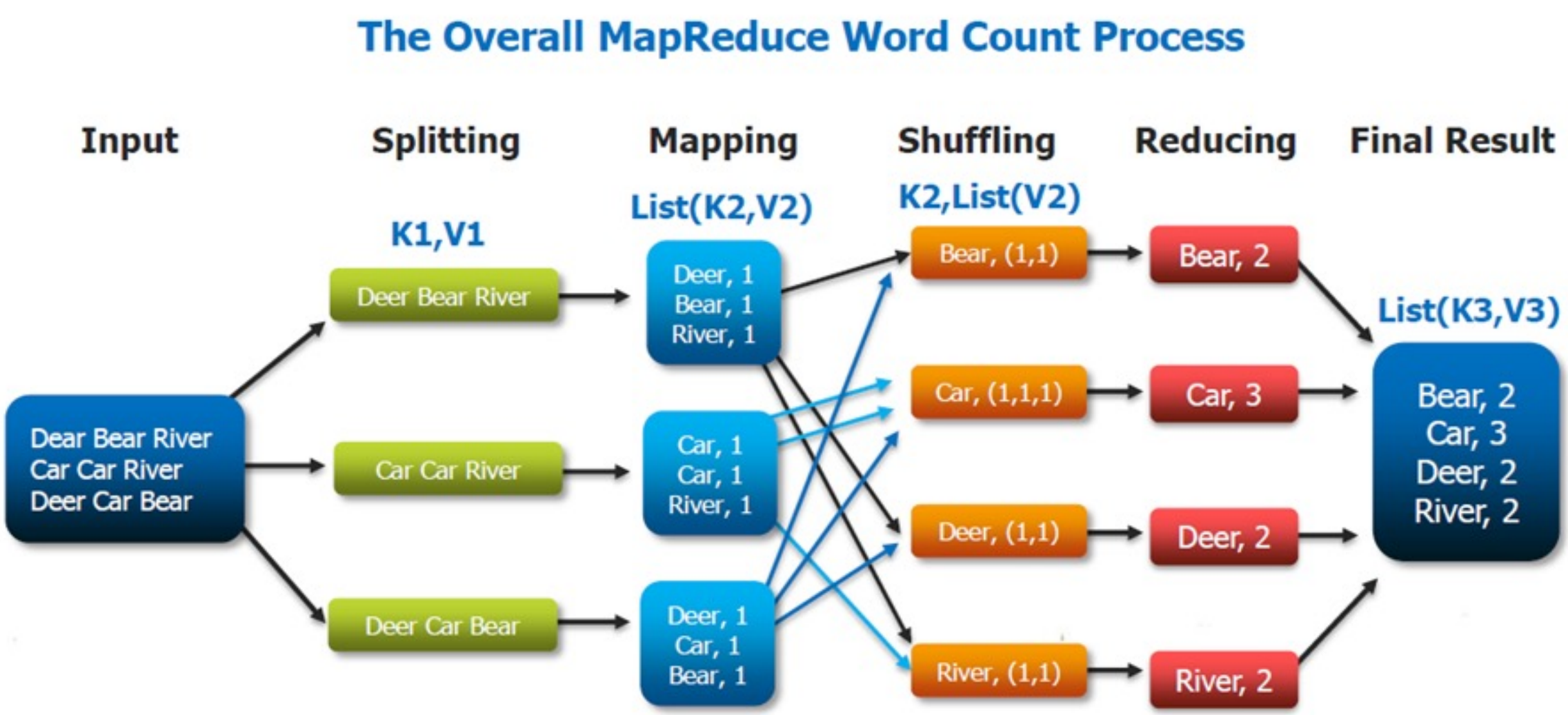
Apache Spark

Apache Foundation

- American nonprofit corporation
- Started with the development of the Apache HTTP server in 1993
- Now manages many open source tools
- Uses a decentralised working approach to produce open source software under the Apache License
- Holds ApacheCon conferences
- Named *"from respect for the Native American Apache Nation, well known for their superior skills in warfare strategy and their inexhaustible endurance."*

MapReduce

- Cluster computing paradigm (technique)
- Inspired by **map** and **reduce** functions from functional programming languages
- Originally referred to a specific Google implementation, but now a general term for the approach (Google stopped relying on it in 2014)



Hadoop

- Large collection of utilities which work together to support cluster computing work on big data.
- Uses the MapReduce programming model
- Takes hardware failures into account and assumes they will regularly occur

•*Hadoop Common* – contains libraries and utilities needed by other Hadoop modules

•*Hadoop Distributed File System (HDFS)* – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster

•*Hadoop YARN* – a platform responsible for managing computing resources in clusters and using them for scheduling users' applications

•*Hadoop MapReduce* – an implementation of the MapReduce programming model for large-scale data processing

Apache Hive

- Data warehouse built on top of Apache Hadoop
- Allows a form of SQL called HiveQL to be used to query a range of databases and filesystems
- HiveQL can compile automatically to MapReduce jobs or Spark jobs
- Schema on Read rather than Schema on Write

"HiveQL does not strictly follow the full SQL-92 standard. HiveQL offers extensions not in SQL, including multitable inserts and create table as select, but only offers basic support for indexes. HiveQL lacked support for transactions... and only limited subquery support. Support for insert, update, and delete with full ACID functionality was made available with release 0.14."

Apache Spark

- Open source cluster computing framework
- Originally developed at AMPLab (Algorithms, Machines and People) at UC Berkeley; codebase donated to Apache Foundation

- Based around the RDD: Resilient Distributed Dataset

Apache Spark

Integrated

Seamlessly mix SQL queries with Spark programs.

```
results = spark.sql(
  "SELECT * FROM people")
names = results.map(lambda p: p.name)
```

Apply functions to results of SQL queries.

Spark SQL lets you query structured data inside Spark programs, using either SQL or a familiar DataFrame API. Usable in Java, Scala, Python and R.

https://spark.apache.org/sql/

Uniform Data Access

Connect to any data source the same way.

DataFrames and SQL provide a common way to access a variety of data sources, including Hive, Avro, Parquet, ORC, JSON, and JDBC. You can even join data across these sources.

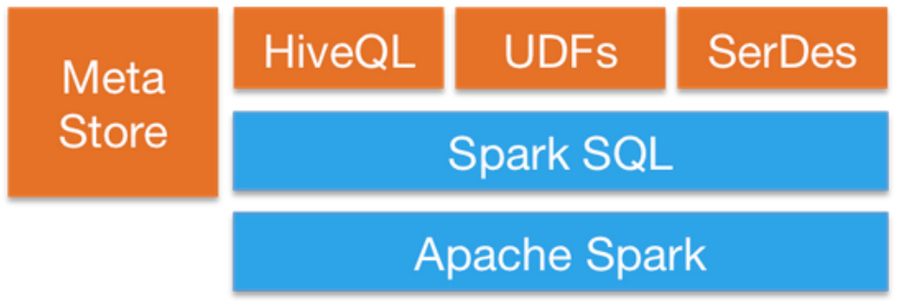
```
spark.read.json("s3n://...")
  .registerTempTable("json")
results = spark.sql(
  """SELECT *
     FROM people
     JOIN json ...""")
```

Query and join different data sources.

https://spark.apache.org/sql/

Hive Integration

Run SQL or HiveQL queries on existing warehouses.



Spark SQL supports the HiveQL syntax as well as Hive SerDes and UDFs, allowing you to access existing Hive warehouses.

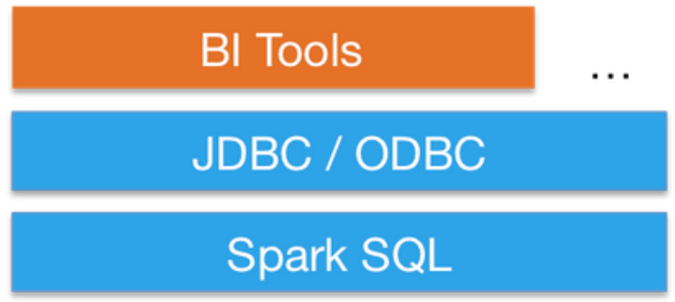
Spark SQL can use existing Hive metastores, SerDes, and UDFs.

https://spark.apache.org/sql/

Standard Connectivity

Connect through JDBC or ODBC.

A server mode provides industry standard JDBC and ODBC connectivity for business intelligence tools.



Use your existing BI tools to query big data.

JDBC: Java DataBase Connectivity

ODBC: Open DataBase Connectivity

https://spark.apache.org/sql/

Apache Spark

Performance & Scalability

Spark SQL includes a cost-based optimizer, columnar storage and code generation to make queries fast. At the same time, it scales to thousands of nodes and multi hour queries using the Spark engine, which provides full mid-query fault tolerance. Don't worry about using a different engine for historical data.

Community

Spark SQL is developed as part of Apache Spark. It thus gets tested and updated with each Spark release.

If you have questions about the system, ask on the Spark mailing lists.

The Spark SQL developers welcome contributions. If you'd like to help out, read how to contribute to Spark, and send us a patch!

Getting Started

To get started with Spark SQL:

- Download Spark. It includes Spark SQL as a module.
- Read the Spark SQL and DataFrame guide to learn the API.

https://spark.apache.org/sql/

When do we need Spark?

Is it wise to learn Spark?

SQL variants, and NoSQL

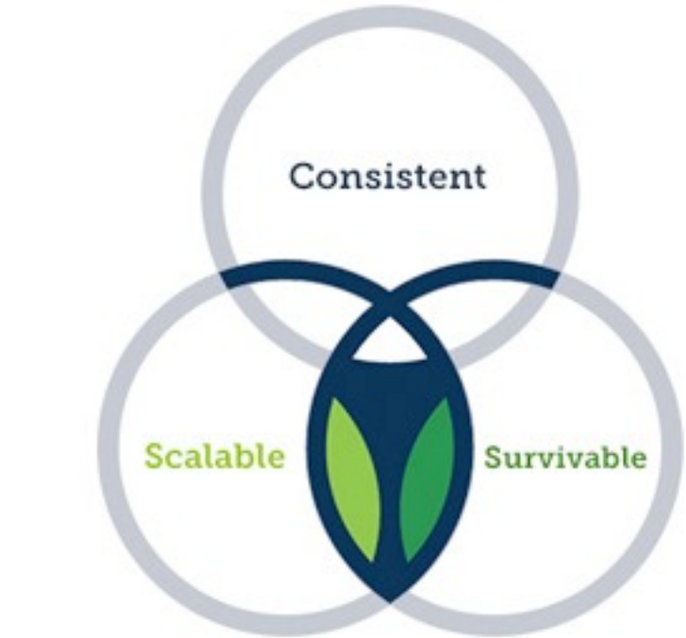
CockroachDB

- Open source database project, supporting a dialect of SQL, which is scalable, replicated and transactional

- Designed for cloud operation and to be “almost impossible” to take down

- Founded in 2015 by ex-Google employees who worked on the Google File System (distributed)

- Inspired by the Google database project Spanner, from which it borrows several techniques



MongoDB

- Free, open source NoSQL database system
- Development began in 2007

- NoSQL: records are not organised in rows and columns, or even in tables. Schemas and datatypes are not enforced.
- This means it can be used as a filesystem for storing documents, images, etc.

- Scales horizontally using sharding

Why you should never, ever, ever use MongoDB

19 Jul 2015

MongoDB is evil. It...

- ... loses data (sources: 1, 2)
- ... in fact, for a long time, ignored errors by default and assumed every single write succeeded no matter what (which on 32-bits systems led to losing all data silently after some 3GB, due to MongoDB limitations)
- ... is slow, *even* at its advertised usecases, and claims to the contrary are completely lacking evidence (sources: 3, 4)
- ... forces the poor habit of implicit schemas in nearly all usecases (sources: 4)
- ... has locking issues (sources: 4)
- ... has an atrociously poor response time to security issues - it took them **two years** to patch an insecure default configuration that would expose *all of your data* to anybody who asked, without authentication (sources: 5)
- ... is not ACID-compliant (sources: 6)
- ... is a nightmare to scale and maintain
- ... isn't even exclusive in its offering of JSON-based storage; PostgreSQL does it too, and other (better) document stores like CouchDB have been around for a long time (sources: 7, 8)

http://cryto.net/~joepie91/blog/2015/07/19/why-you-should-never-ever-ever-use-mongodb/

XML, JSON and YAML

XML

```
<empinfo>
    <employees>
        <employee>
            <name>James Kirk</name>
            <age>40</age>
        </employee>
        <employee>
            <name>Jean-Luc Picard</name>
            <age>45</age>
        </employee>
        <employee>
            <name>Wesley Crusher</name>
            <age>27</age>
        </employee>
    </employees>
</empinfo>
```

JSON

```
{   "empinfo" :
        {
            "employees" :   [
            {
                "name" : "James Kirk",
                "age" : 40,
            },
            {
                "name" : "Jean-Luc Picard",
                "age" : 45,
            },
            {
                "name" : "Wesley Crusher",
                "age" : 27,
            }
                        ]
        }
}
```

XML, JSON and YAML

XML

```
<user id="smith">
    <firstname>Al
    </firstname>
    <lastname>Smith
    </lastname>
    <phone>555-1111
    </phone>
    <email>al@gmail.com
    </email>
    <email>smith@nku.edu
    </email>
</user>
```

YAML

```
smith:
  firstname: Al
  lastname: Smith
  phone: 555-1111
  emails:
  - email: al@gmail.com
  - email: smith@nku.edu
```

Redis

- Open source in-memory database system
- All data is held in RAM, making for extremely fast query times
- Implements simple key—value stores

- Supports strings, lists, maps, sets, sorted sets, and more complex data types

- Supports master-slave replication
- However, each instance does not have parallel execution and is a single-threaded process

- Useful for counting, caching, message queues...


Can I join you?
comic.browserling.com
An SQL query walks into a bar and sees two tables.
He walks up to them and says "Can I join you?"