Distributed Data Systems

DIANE WOODBRIDGE, PH.D

Announcement

Now available on Canvas

- Previous Project Samples

Task 0 (Optional)

- Due : Thu, Feb 2nd
- Only submitted topics will be included

Recap

Course Overview

Workflow Management

Apache Airflow

Airflow DAG Creation

NoSQL Learning Objectives

Understand the needs and characteristics of schemaless non-relational databases for storing and querying data in distributed settings.

Gain insights to make judgments to choose SQL or NoSQL (and which NoSQL) with various hands-on exercises.

Gain experience with NoSQL databases including MongoDB through in-class exercises and homework.

Be competent to work with Spark and MongoDB in a distributed environment including Amazon Web Services, MongoDB Atlas.

NoSQL Interview Questions

What is NoSQL?
Relational Database vs. NoSQL
Impedance mismatch
Polyglot persistence
Aggregate-oriented database
Key-value database
Document database
Column family database
Graph database
Replication vs sharding

CAP Theorem
Eventual Consistency

See the Interview Questions module on Canvas

MongoDB Interview Questions

- MongoDB's type
- MongoDB's characteristics
- Alternative databases
- Supported programming languages
- Index
- Aggregation Operations(aggregation pipeline)
- Sharding
- Replication
- ObjectId
- Consistency

See the Interview Questions module on Canvas

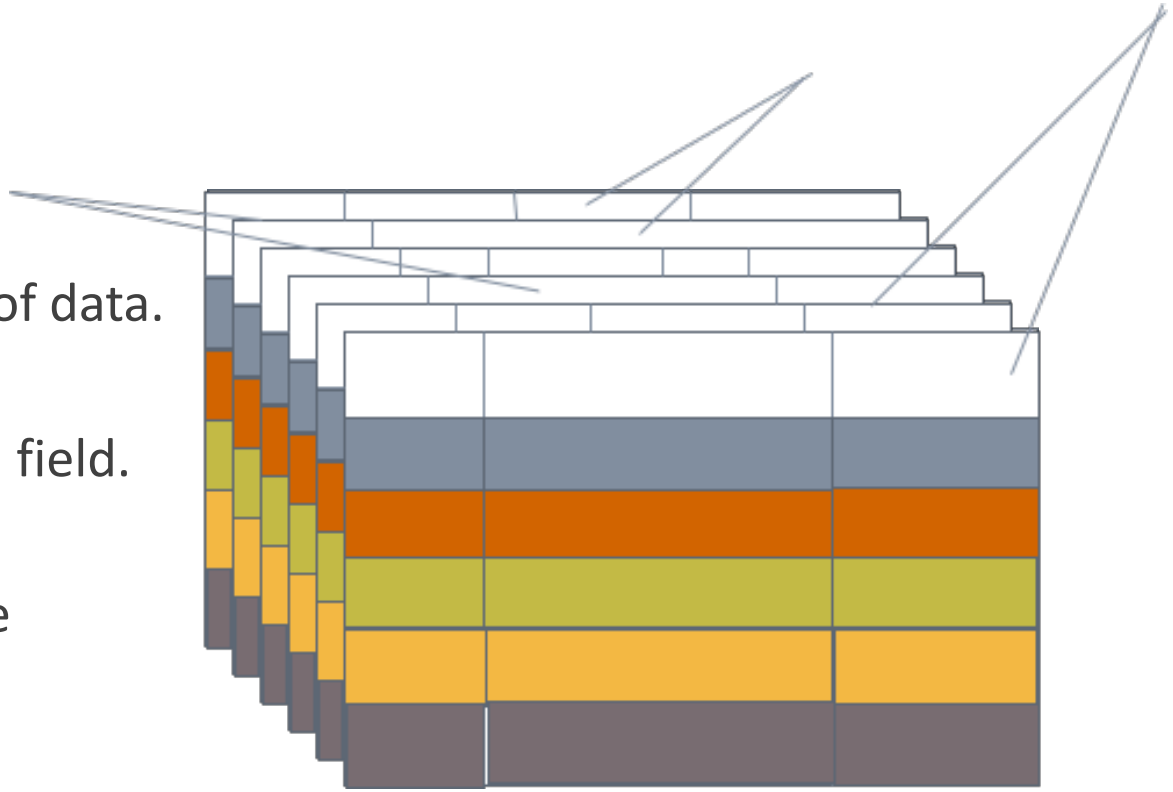
Contents

Contents

Review - Relational Database

A DBMS based on the relational data model.

- Relational Model
- Relation = Records
  - Every relation has a schema : Description of data.
    - Specifies its name, name of each fields (column/attribute), and the type of each field. Ex. Students(sid : string, name: string)
    - Integrity constraints : conditions that the records in a relation must satisfy
  - Each relation has a collection of tuples



Review - Relational Database

SQL (Structured Query Language)

- Manage data in a relational database
- Basic operations (CRUD)

1. **C**reate/Insert : Define a new table, Insert a tuple into a table.
2. **R**ead : Retrieve data from a table.
3. **U**pdate : Change an existing table and tuples.
4. **D**elete : Remove the tuple(s) from a table or the table.

What does ACID Transaction mean?

- Atomic, Concurrent, Independent, Durable Transaction
- Atomic, Consistent, Isolated, Durable Trasaction
- Atomic, Consistent, Independent, Durable Transaction
- Atomic, Concurrent, Isolated, Duplicatioed Transaction
- Atomic, Consistent, Isolated, Duplicatioed Transaction

What does ACID Transaction mean?

Atomic, Concurrent, Independent, Durable Transaction

Atomic, Consistent, Isolated, Durable Trasaction

Atomic, Consistent, Independent, Durable Transaction

Atomic, Concurrent, Isolated, Duplicatioed Transaction

Atomic, Consistent, Isolated, Duplicatioed Transaction

What does ACID Transaction mean?

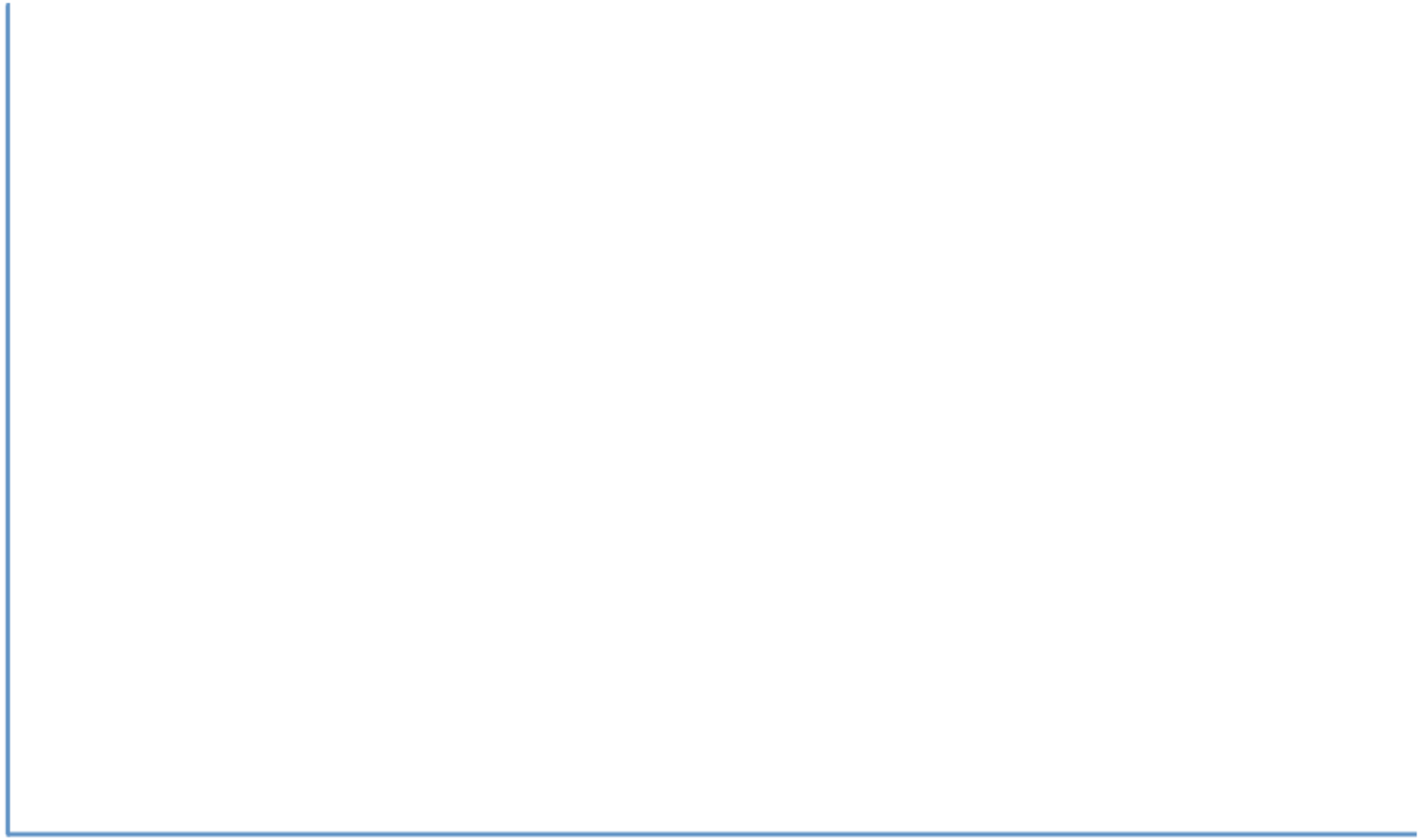
Atomic, Concurrent, Independent, Durable Transaction

Atomic, Consistent, Isolated, Durable Trasaction

Atomic, Consistent, Independent, Durable Transaction

Atomic, Concurrent, Isolated, Duplicatioed Transaction

Atomic, Consistent, Isolated, Duplicatioed Transaction



What does ACID Transaction mean?

When poll is active, respond at **PollEv.com/msds** Text **MSDS** to **37607** once to join

Atomic, Concurrent, Independent, Durable Transaction

Atomic, Consistent, Isolated, Durable Trasaction

Atomic, Consistent, Independent, Durable Transaction

Atomic, Concurrent, Isolated, Duplicatioed Transaction

Atomic, Consistent, Isolated, Duplicatioed Transaction

Review - Relational Database

Pros

ACID (Atomic, Consistent, Isolated and Durable) **Transaction Management**

- **A**tomicity : Actions in transactions are carried out all or none.
  - DBMS logs all actions so that it can undo the actions of aborted transaction
- **C**onsistency: A transaction must change affected data only in allowed ways, according to all defined rules.
- **I**solation : When there are several transactions happen concurrently, users should be able to understand a transaction without considering the effect of other concurrently existing transaction.
- **D**urability : Once a transaction has been successfully completed, its effects should persist even if the system crashes before all its changes are reflected on disk.

https://en.wikipedia.org/wiki/ACID

Review - Relational Database

Pros

- **Standard Model**
  - Different vendors' query languages are similar.
  - Transaction operations work in the similar way.

Isn't RDBMS perfect enough?

Big Data

Volume, Variety and Velocity is beyond the ability of traditional data-processing application software can adequately handle.

- Volume : Constantly moving target - as of 2012 ranging from terabytes (TB) to exabytes (EB).
- Variety : Mostly unstructured.
- Velocity : High speed in creation and required speed for ingestion and analysis.

- Example domains
  - Internet of Things (IoT) – Smart homes, Electronic toll collection, Smart grid, Wearable, etc.
  - Health Care – Electronic health records (EHR)
  - Marketing – Consumer data

Storing and Processing Big Data became an important issue!

https://en.wikipedia.org/wiki/Big_data
https://www.ibm.com/blogs/watson-health/the-5-vs-of-big-data/

Big Data and Relational Databases

New Data Types and Large Volumes of Data

- Data and traffic increase.
  - As a solution, started to store data in the clusters.

    Most relational databases are not designed to run on clusters.

    Reliability issues.
- Schema changes.
  - Doesn’t fit in a table-like structure.

NoSQL (Not Only SQL)

Why NoSQL?

- **Impedance Mismatch**
  - Relational Model ≠ In-memory data structure (Object)
    - For application development productivity, it is better to map with in-memory data structure.
- **Large volume of data (2000s)**
  - Scale-up vs. Scale-out?
    - Run large data on a cluster of many smaller and cheaper machines. → Cheaper and Reliable.
- Example of NoSQL database : Google BigTable, Amazon DynamoDB

https://cloud.google.com/bigtable
https://aws.amazon.com/dynamodb/

Contents

NoSQL (Not Only SQL)

Generally,

- Take schemaless data.
- Non-relational.
- Open-source.
- Trade off traditional consistency for other properties.
- Run on clusters.

Summary - SQL and NoSQL

SQL

Pros

Persistent data storage.
Concurrency.
Standard Model.

Cons

Impedence mismatch.
Hard to scale.
Fixed schema.

Example

MySQL, PostgreSQL,
SQL Server, Oracle

VS

NoSQL

Pros

Mostly open-source.
Schemaless.
Good for non-relational data.
Scalable.
Runs well on distributed systems.

Cons

Installation, toolsets still maturing.

Example

Redis, MongoDB, Cassandra,
OrientDB

Postgres and NoSQL

Postgres supports a JSON data type.

However,

- Postgres does not offer any native mechanisms for data availability or scaling the database beyond a single server.
  - Lacks mechanisms for automatic failover and recovery between database replicas.
  - No native mechanisms to partition (shard) the database across a cluster of nodes.
- It is not as natural to work with JSON data in Postgres.
  - The non-standard extensions to SQL to query and manipulate JSON are not supported by most tools.

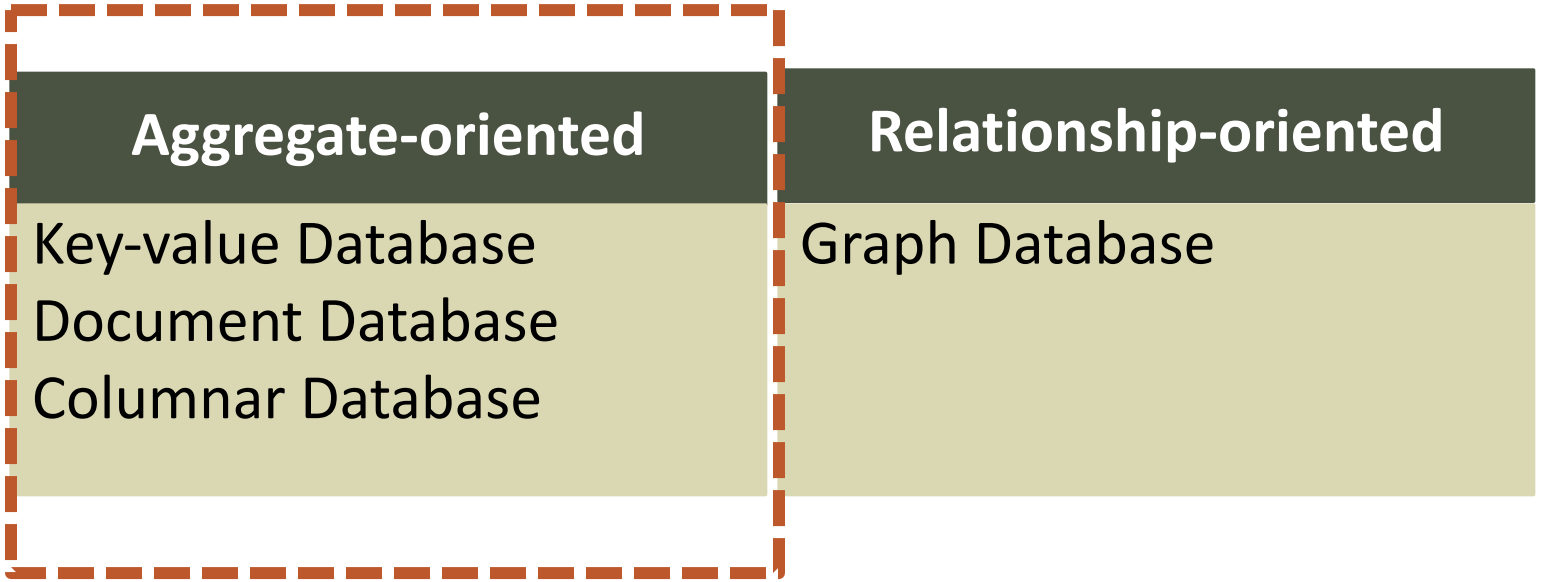
https://www.postgresql.org/docs/9.3/static/functions-json.html
https://www.mongodb.com/compare/mongodb-postgresql

Contents

NoSQL Database Types

Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

NoSQL Database Types



Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

Aggregate-oriented NoSQL Databases

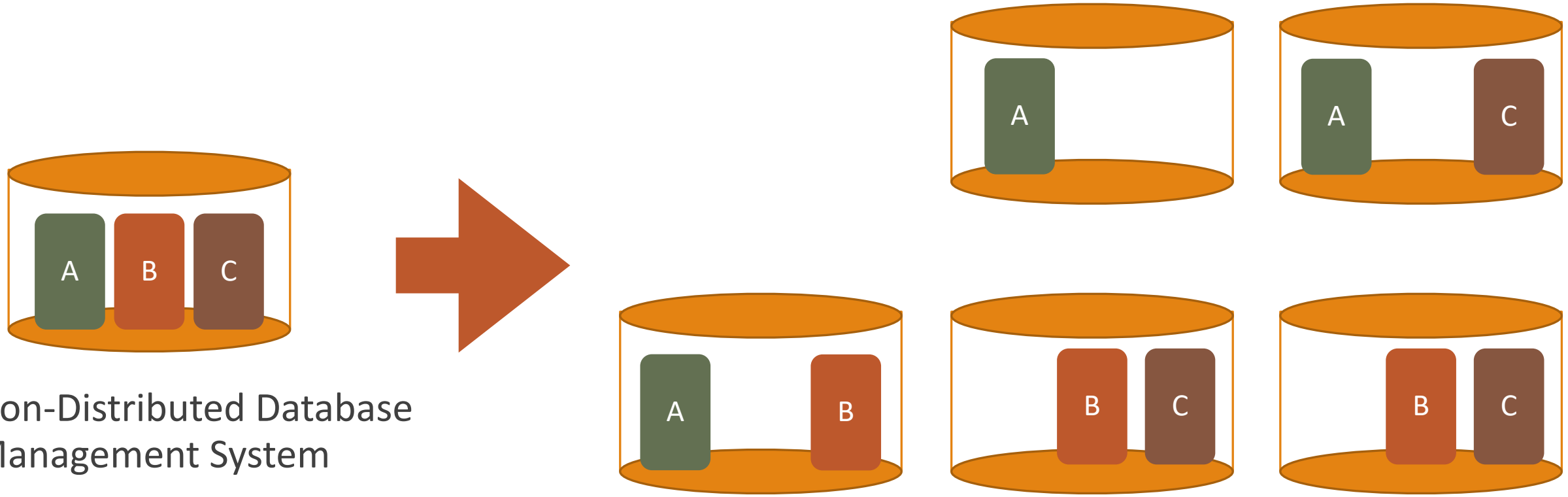
Aggregate: Collection of related objects treated as a unit.

- For analyzing data, you might want to place some data together as a unit.
  - On a cluster, an aggregate is stored together on a node.
  - A single aggregate is a unit of atomic updates.
- Aggregate-oriented databases use aggregates indexed by **key** for data lookup.
- Aggregate-oriented databases don't have ACID transactions that span multiple aggregates.
  - cf. RDBMS is aggregate-ignorant.


```
{
  "_links":{
     "self":{
        "href":"/member/109087/cart"
     },
     "http://example.com/rels/payment":{
        "href":"/payment"
     }
  },
  "_embedded":{
     "http://example.com/rels/cart-item":[
        {
           "_links":{
              "self":{
                 "href":"/member/109087/cart/14418796"
              }
           },
           "id":"14418796",
           "quantity":1,
           "expire_time":"2009-09-11T08:00:00-07:00",
           "_embedded":{
              "http://example.com/rels/sku":[
                 {
                    "_links":{
                       "self":{
                          "href":"/skus/654654"
                       },
                       "http://example.com/images/cart/item":"http://example.com/product/6895/thumbnail.jp
                    },
                    "color":"Black",
                    "size":"S",
                    "returnable":false,
                    "price":49
                 }
              ]
           }
        }
     ]
  }
}
```


https://gist.github.com/hjr3/2289546

Aggregate-oriented NoSQL Databases



Non-Distributed Database Management System

Distributed Database Management System

Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

Aggregate-oriented NoSQL Databases

Pros

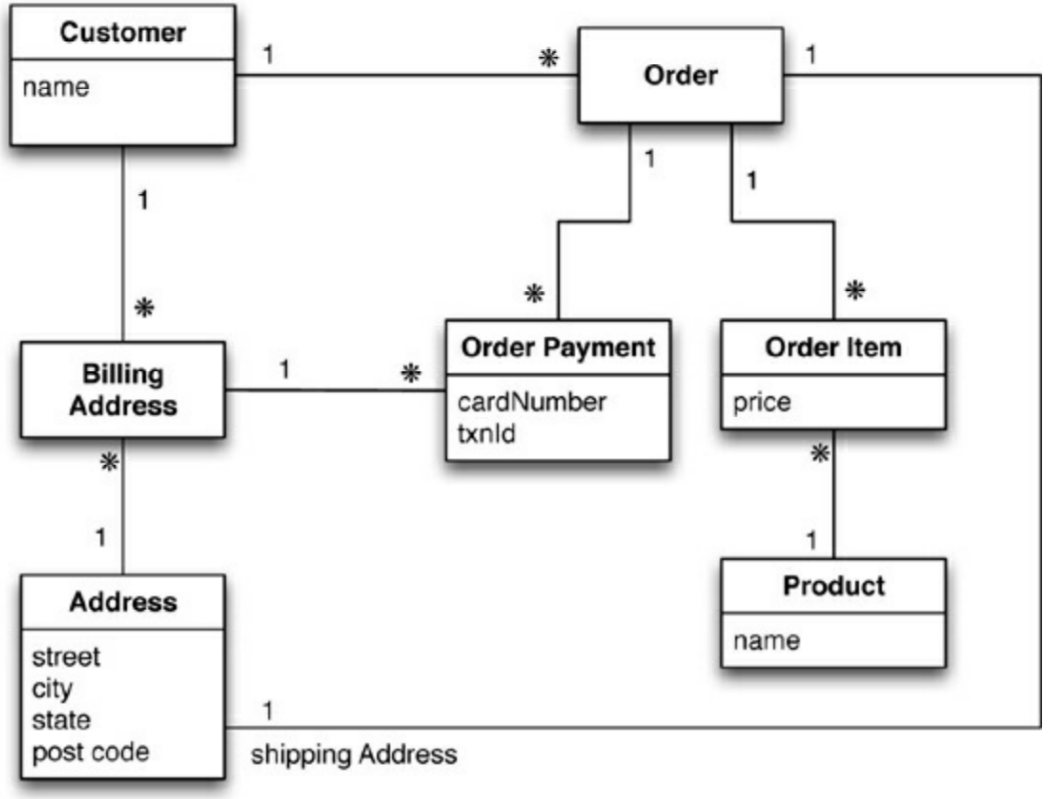
- Provides clearer semantics to consider by focusing on the aggregate unit used by applications.
- Better design choice for running on a cluster.

Cons

- Drawing boundaries of an aggregate is not easy.
- When a goal of data management/analysis is not clear, aggregate models might not be the best choice.
- Doesn’t support ACID transactions.

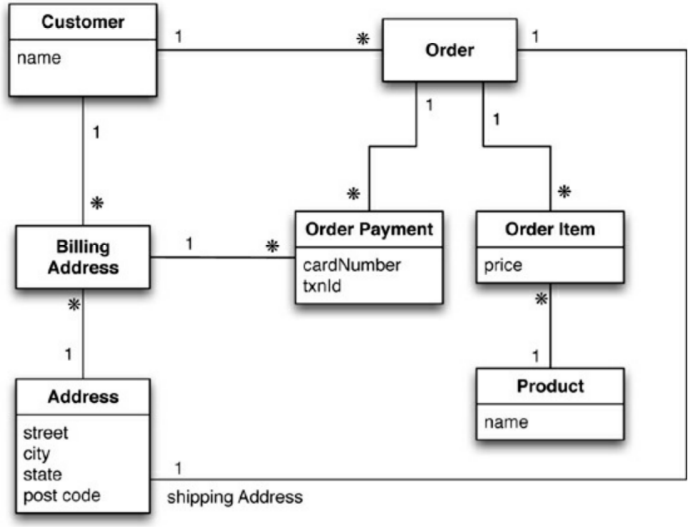
Example 1

Draw boundaries of an aggregate.



Example 1

Draw boundaries of an aggregate.



Customer

c_id
name
billing address

Order

o_id
c_id
shipping address

Item

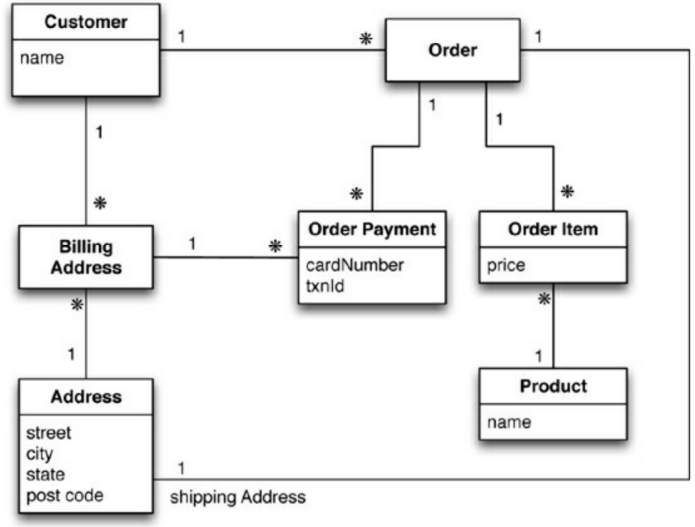
product name
price

Payment

cardNumber
txnId
billing address

Example 1

Draw boundaries of an aggregate.



Many ways to draw aggregate boundaries.

➔ Depends on how your application manipulates.

NoSQL Database Types

Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

Relationship-oriented NoSQL Database

Needs of relationship-oriented DB

- Relational database with complex schema
  - Hard to understand, query, generalize and integrate data.

Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

Relationship-oriented NoSQL Database

Graph Database

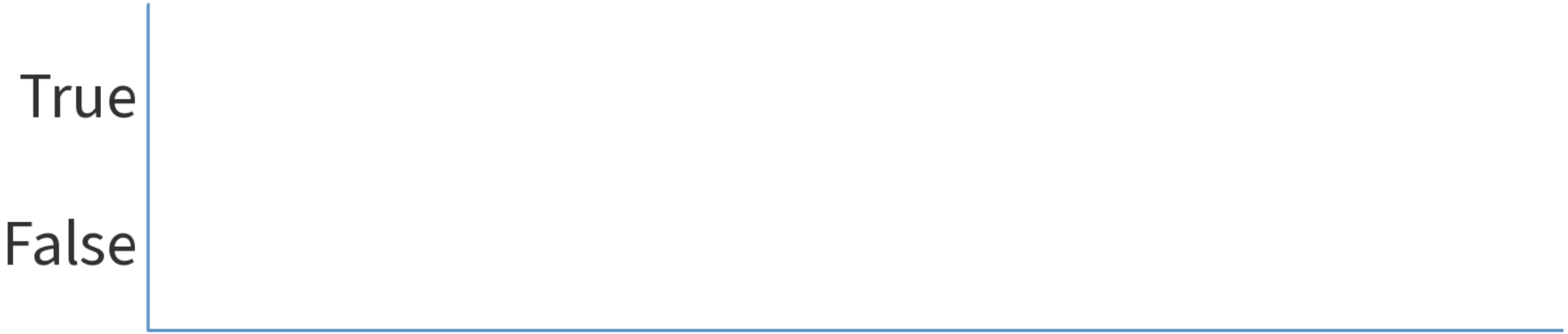
- Nodes (Object) and edges (Relationships) representation
- Relationships and their attributes are explicitly defined .
- For data with complex relationships.
  - Focuses on graph traverse (more than insert.).
    - cf) RDBMS : Many joins can cause poor performance.
- Running on a single server rather than distributed across clusters.

Relationship-oriented NoSQL Database are desinged to run in distributed enviroments.

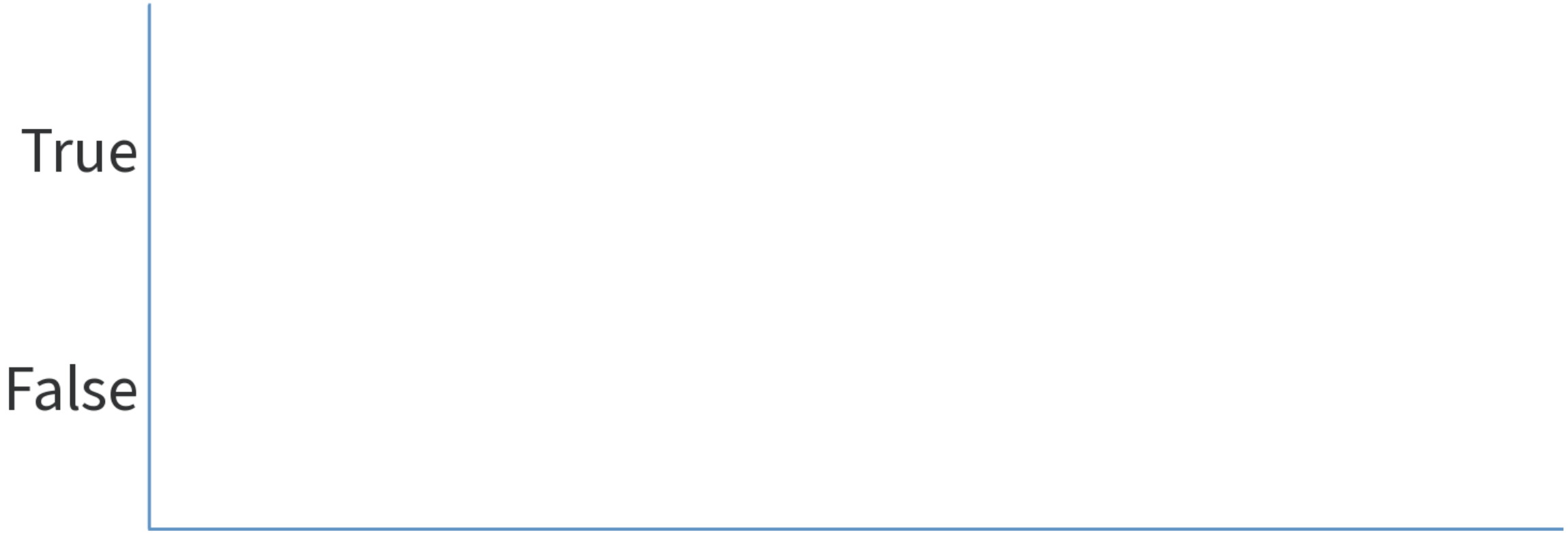
True

False

Relationship-oriented NoSQL Database are desinged to run in distributed enviroments.



Relationship-oriented NoSQL Database are desinged to run in distributed enviroments.



NoSQL Database Examples

Key-value
- Redis, Riak, Berkeley DB, etc.

Document
- MongoDB, CouchDB, OrientDB, RavenDB, etc.

Column Family
- Cassandra, Hbase, Amazon SimpleDB, etc.

Graph
- Neo4J, OrientDB, FlockDB, etc.

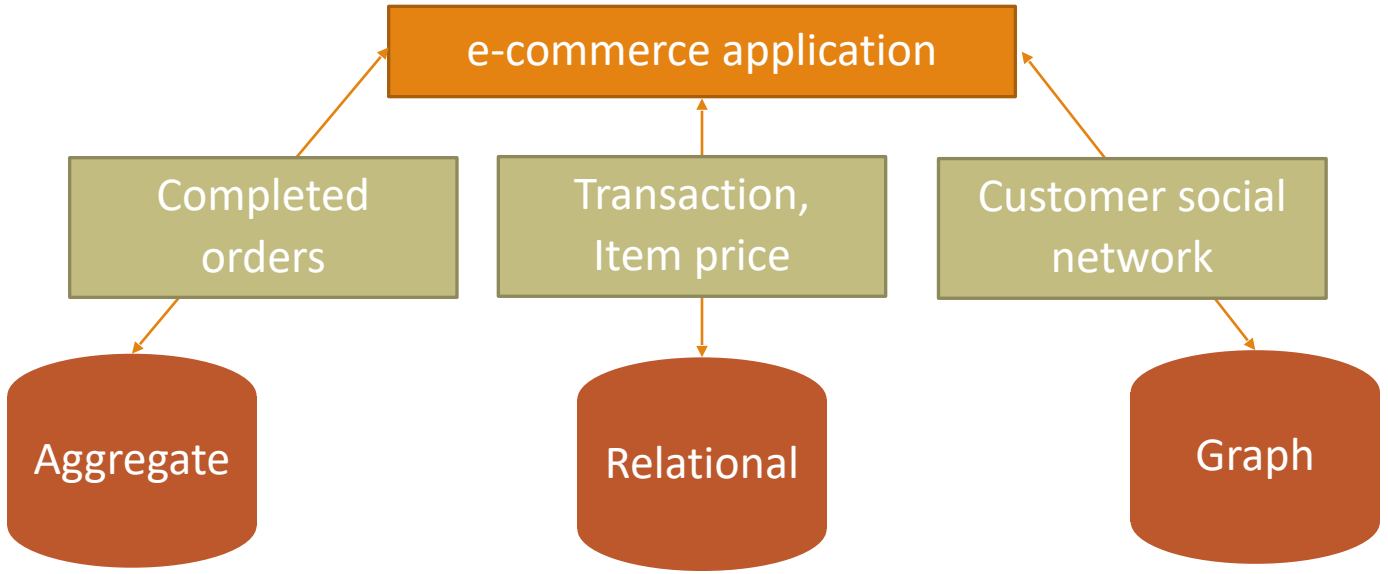
Aggregate-oriented	Relationship-oriented
Key-value Database Document Database Columnar Database	Graph Database

Contents

Choice of DBMS

Polyglot Persistence

- Using multiple data storage technologies, chosen based on the way data is being used by individual applications.
- **NoSQL data stores do not replace relational databases.**



Database Ranking (January 2023)

402 systems in ranking, January 2023

Rank			DBMS	Database Model	Score		
Jan 2023	Dec 2022	Jan 2022			Jan 2023	Dec 2022	Jan 2022
1.	1.	1.	Oracle	Relational, Multi-model	1245.17	-5.14	-21.72
2.	2.	2.	MySQL	Relational, Multi-model	1211.96	+12.56	+5.91
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	919.39	-4.96	-25.43
4.	4.	4.	PostgreSQL	Relational, Multi-model	614.85	-3.13	+8.29
5.	5.	5.	MongoDB	Document, Multi-model	455.18	-14.15	-33.38
6.	6.	6.	Redis	Key-value, Multi-model	177.56	-5.01	-0.43
7.	7.	7.	IBM Db2	Relational, Multi-model	143.57	-3.05	-20.63
8.	8.	8.	Elasticsearch	Search engine, Multi-model	141.16	-3.76	-19.59
9.	9.	9.	Microsoft Access	Relational	133.36	-0.47	+4.41
10	10	10	SQLite	Relational	131.49	-0.94	+4.06

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

Contents

Why NoSQL?

What is NoSQL?

Types of NoSQL

Choice of DBMS (SQL and NoSQL)

Document Database

Document Databases

Document Databases: For storing and retrieving documents with self-contained schema including XML, JSON, etc.

- Store documents in the value part of the key-value store.

```xml
<?xml version="1.0" standalone="yes"?>
<BankAccount>
      <Number>1234</Number>
      <Type>Checking</Type>
      <OpenDate>11/04/1974</OpenDate>
      <Balance>25382.20</Balance>
      <AccountHolder>
            <LastName>Singh</LastName>
            <FirstName>Darshan</FirstName>
      </AccountHolder>
</BankAccount>
```

```json
{
      "empid": "SJ011MS",
      "personal": {
            "name":"Smith Jones",
            "gender": "Male",
            "age": 28,
            "address":
            {
                  "streetaddress": "7 24th Street",
                  "city": "New York",
                  "state": "NY",
                  "postalcode": "10038"
            }
      },
      "profile": {
            "designation": "Deputy General",
            "department": "Finance"
      }
}
```

Document Databases

- **Schemaless**
  - Schema of the data can differ across documents, but these documents can still belong to the same collection.
    - Some attributes do not exist in another document.
    - New attributes can be created without defining them in the existing documents.

```
{
_id : 1234
firsname : Diane,
lastname : Woodbridge,
address :
            {state: CA, city: Corte Madera},
Interest : Books
}
```

```
{
_id :2345
firsname : Diane,
lastname :  Woodbridge,
order :
      {“Chodorow, Kristina. MongoDB:
      the definitive guide. " O'Reilly
      Media, Inc.", 2013.”}
type :  e-copy
}
```

Document Databases

☐ include secondary database models

55 systems in ranking, January 2023

Rank			DBMS	Database Model	Score		
Jan 2023	Dec 2022	Jan 2022			Jan 2023	Dec 2022	Jan 2022
1.	1.	1.	MongoDB	Document, Multi-model	455.18	-14.15	-33.38
2.	2.	2.	Amazon DynamoDB	Multi-model	81.55	-2.29	+1.70
3.	3.		Databricks	Multi-model	60.82	+0.08	
4.	4.	↓ 3.	Microsoft Azure Cosmos DB	Multi-model	37.96	+0.01	-2.08
5.	5.	↓ 4.	Couchbase	Document, Multi-model	25.30	-1.40	-3.56
6.	6.	↓ 5.	Firebase Realtime Database	Document	18.76	-0.25	-0.60
7.	7.	↓ 6.	CouchDB	Document, Multi-model	14.88	-0.64	-1.76
8.	8.	↑ 9.	Google Cloud Firestore	Document	11.10	-0.04	+2.17
9.	9.	↓ 8.	MarkLogic	Multi-model	8.86	+0.04	-0.34
10.	10.	↓ 7.	Realm	Document	8.34	-0.37	-1.25

http://db-engines.com/en/ranking/document+store

Comments (What you liked/disliked so far? What should I do for you?)

Contents

Why NoSQL?

What is NoSQL?

Types of NoSQL

Choice of DBMS (SQL and NoSQL)

Document Database

References

Sadalage, Pramod J., and Martin Fowler. *NoSQL distilled: a brief guide to the emerging world of polyglot persistence*. Pearson Education, 2012.

Redmond, Eric, and Jim R. Wilson. *Seven databases in seven weeks: a guide to modern databases and the NoSQL movement*. Pragmatic Bookshelf, 2012.

Date, C. J. "Database systems." *Vols. I & II, Narosa Pub* (1986).