Introduction to Big Data

ECAP456

Dr. Rajni Bhalla

Associate Professor

Learning Outcomes

After this lecture, you will be able to

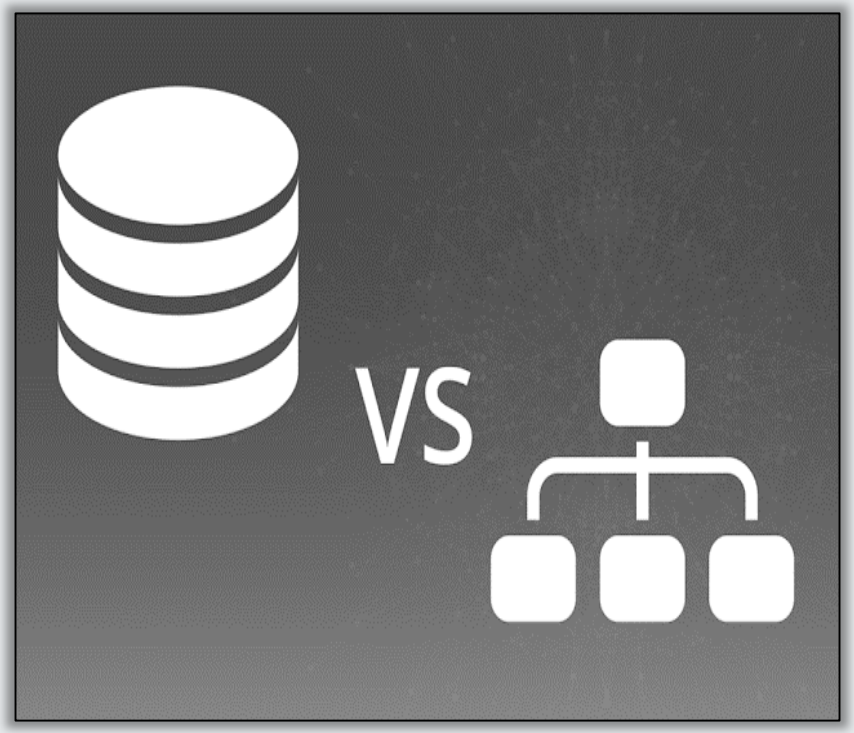
- learn aggregate data models.

Data Model

- A data model is a representation that we use to perceive and manipulate our data.
- It allows us to: –
  - Represent the data elements under analysis, and
  - How these are related to each others .
- This representation depends on our perception.

Data Model

Database	Storage Model
How we interact with the data in the database	How the database stores and manipulates the data internally



Database is distinct from storage model

Data Models: Example

- A Data model is the model of the specific data in an application
- A developer might point to an entity-relationship diagram and refer it as the data model containing
  - customers,
  - orders and
  - products

Data Model

Q: What is the name of data model from last couple of decades?

Ans:-????

Data Model

Q: What is the name of data model from last couple of decades?

Ans:- Relational data model

Data Model

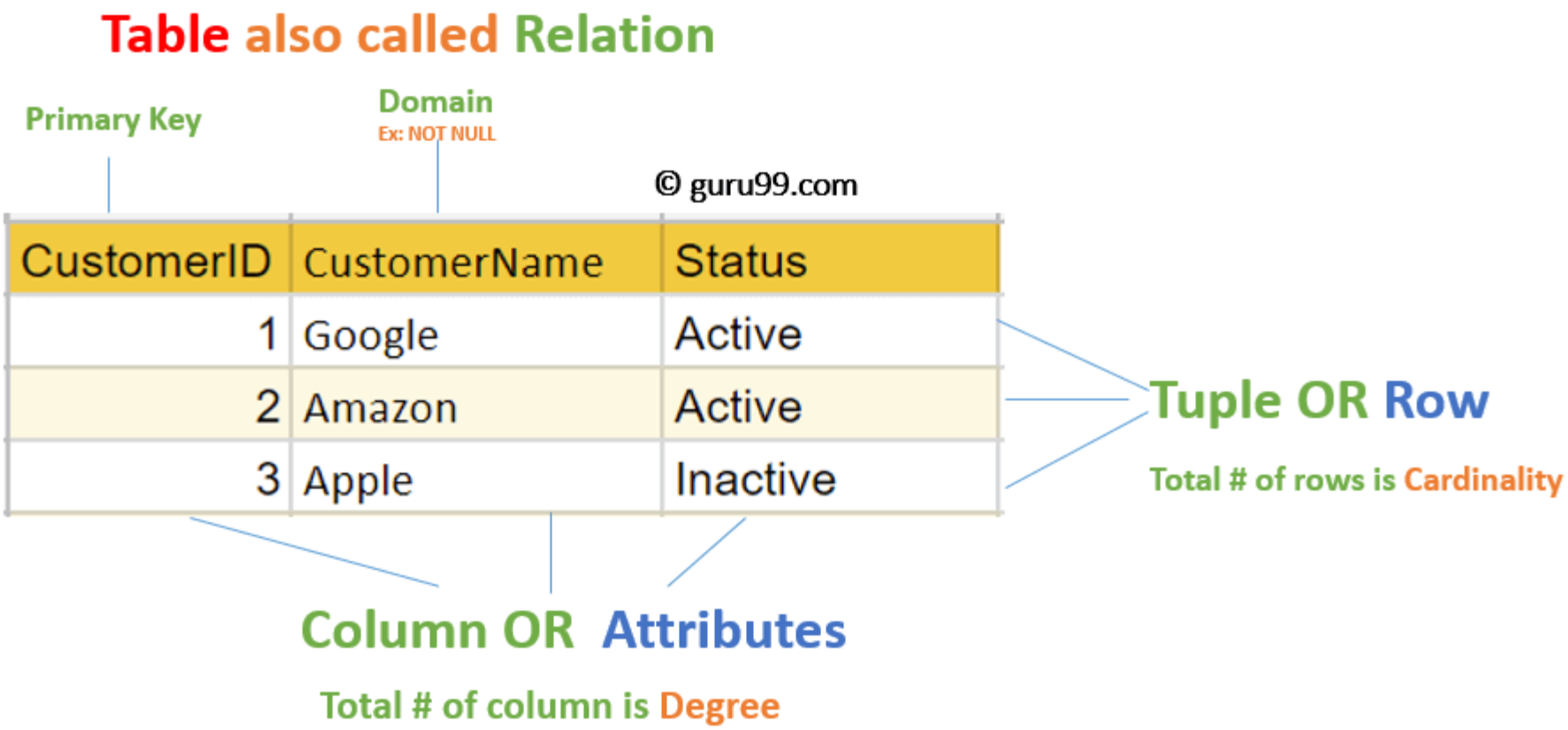


Table also called Relation
Primary Key
Domain
Ex: NOT NULL
© guru99.com
CustomerID
CustomerName
Status
1
Google
Active
2
Amazon
Active
3
Apple
Inactive
Tuple OR Row
Total # of rows is Cardinality
Column OR Attributes
Total # of column is Degree

NoSQL Data Model

- It moves away from the relational data model.
- Each NOSQL database has a different model
  - Key-value
  - Document
  - Column-family
  - Graph, and
  - Sparse (Index based)
- Of these, the first three share a common characteristics (Aggregate Orientation).

Relational Model vs Aggregate Data Model

Relational Model

- The relational model takes the information that we want to store and divides it into tuples (rows).
- However, a tuple is a limited data structure.
- It captures a set of values.
- So, we can't nest one tuple within another to get nested records.
- Nor we can put a list of values or tuple within another.

Relational Model

- This simplicity characterize the relational model.
- It allows us to think on data manipulation as operation that have: –
  - As input tuples, and –
  - Return tuples ▪
- Aggregate orientation takes a different approach.

Aggregate data model

- Relational database modelling is vastly different.
- An aggregate is a collection of data that we interact with as a unit.
- These units of data or aggregates form the boundaries for ACID operations with the database.

Aggregate data model

- NoSQL databases are classified in different data models:

Aggregate data model

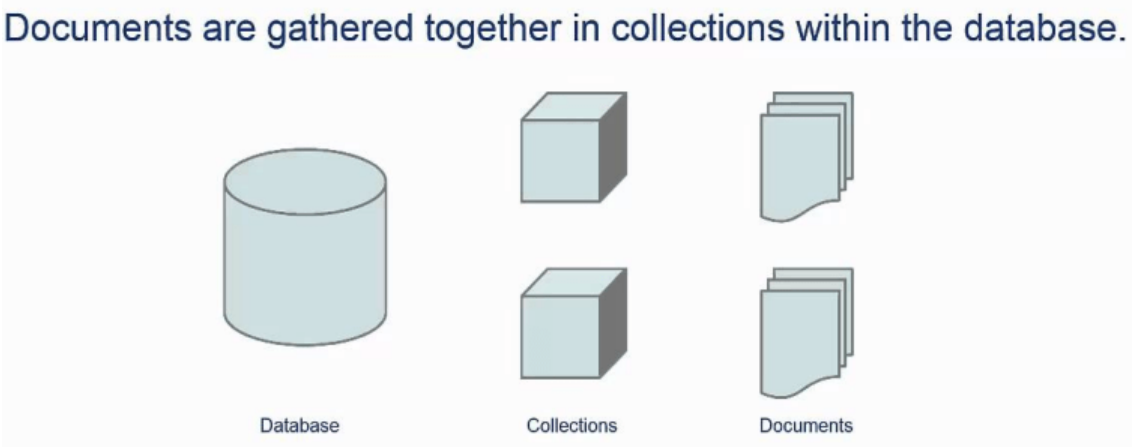
Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Key-value

Aggregate data model

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Key-value



Document Databases

Aggregate data model

	company					
	name	address		website		
1	DataX	city	San Francisco	protocol	https	
		state	California	domain	datax.com	
		street num	135	subdomain	www	
		street	Kearny St			
2	Process-One	city	Arlington	protocol	https	
		state	Virginia	domain	process1.com	
		street num	3500	subdomain	www	
		street	Wilson St			

row key

column family

Column-family databases

Aggregate data model

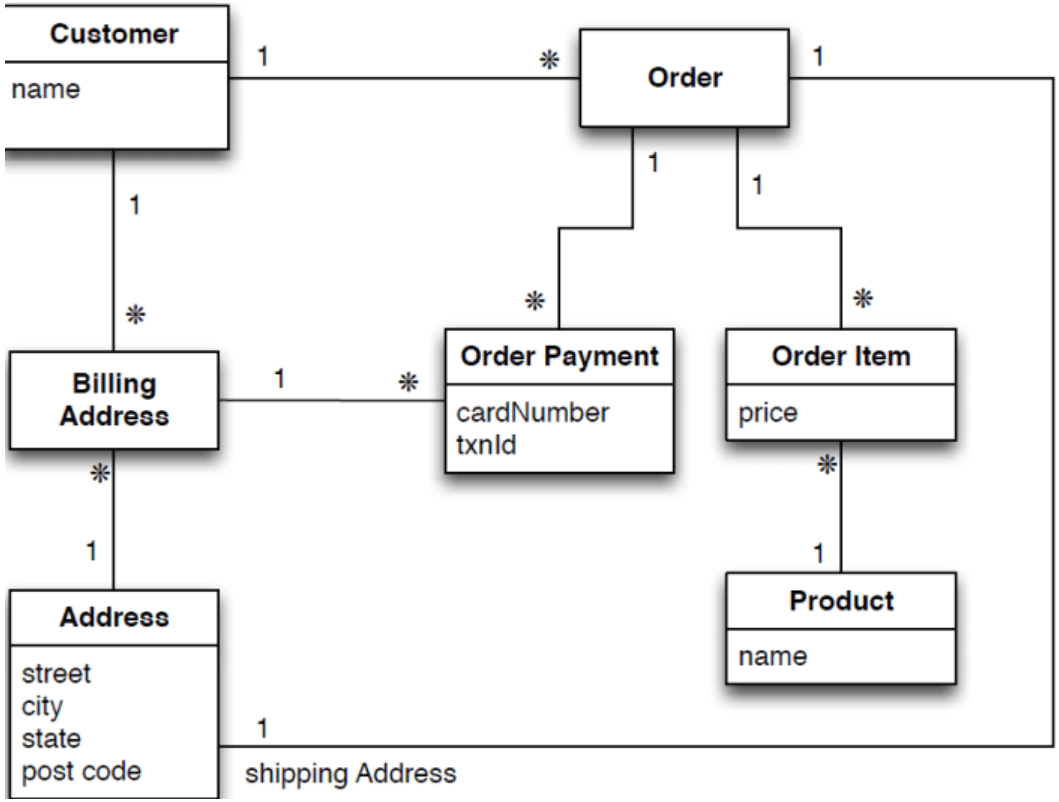
What make it easier for the database?

- Aggregates make it easier for the database to manage data storage over clusters.
- Aggregate-oriented databases work best when most data interaction is done with the same aggregate.

Examples of Relations and Aggregates

Examples of Relational Model

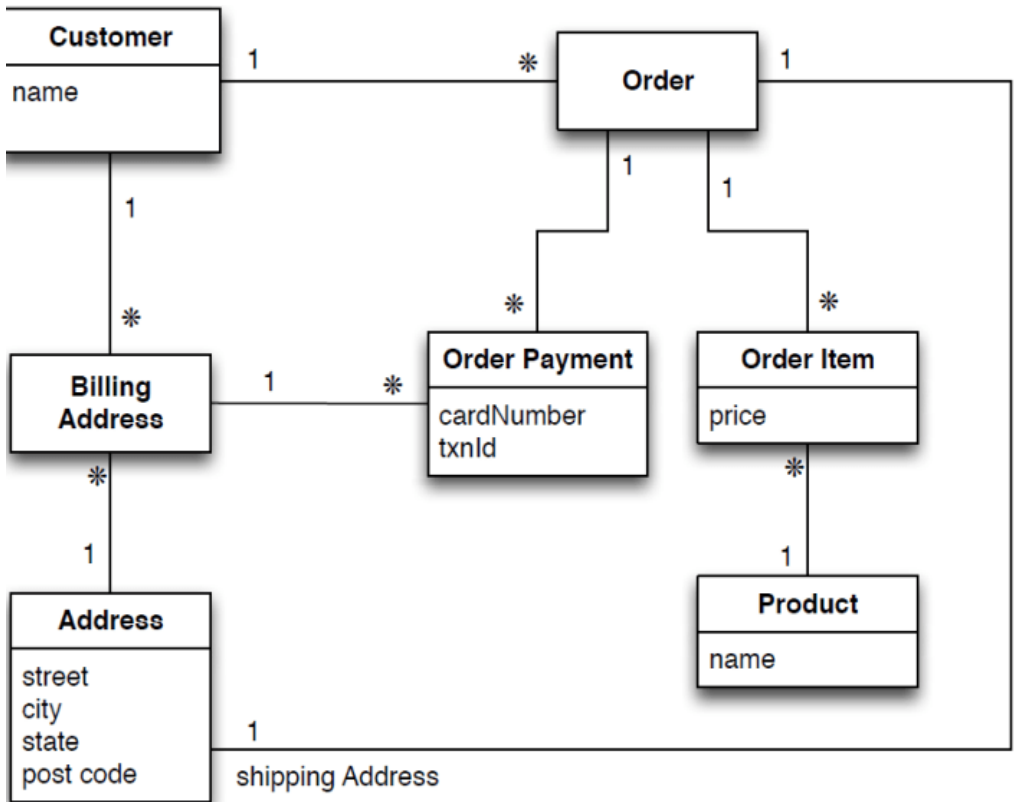
- Assume we are building an e-commerce website;
- We have to store information about: users, products, orders, shipping addresses, billing addresses, and payment data.



Data Model Oriented Around a Relational Database

Examples of Relational Model

- As we are good relational soldier:
  - Everything is normalized
  - No data is repeated in multiple tables.
  - We have referential integrity



Typical data using RDBMS data model

Customer	
Id	Name
1	Martin

Order		
Id	CustomerId	ShippingAddressId
99	1	77

Product	
Id	Name
27	NoSQL Distilled

BillingAddress		
Id	CustomerId	AddressId
55	1	77

OrderItem			
Id	OrderId	ProductId	Price
100	99	27	32.45

Address	
Id	City
77	Chicago

OrderPayment				
Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

Sample Data

Typical data using RDBMS data model

Sample Data for this model

Customer

Id	Name
1	Martin

Order

Id	CustomerId	ShippingAddressId
99	1	77

Product

Id	Name
27	NoSQL Distilled

BillingAddress

Id	CustomerId	AddressId
55	1	77

OrderItem

Id	OrderId	ProductId	Price
100	99	27	32.45

Address

Id	City
77	Chicago

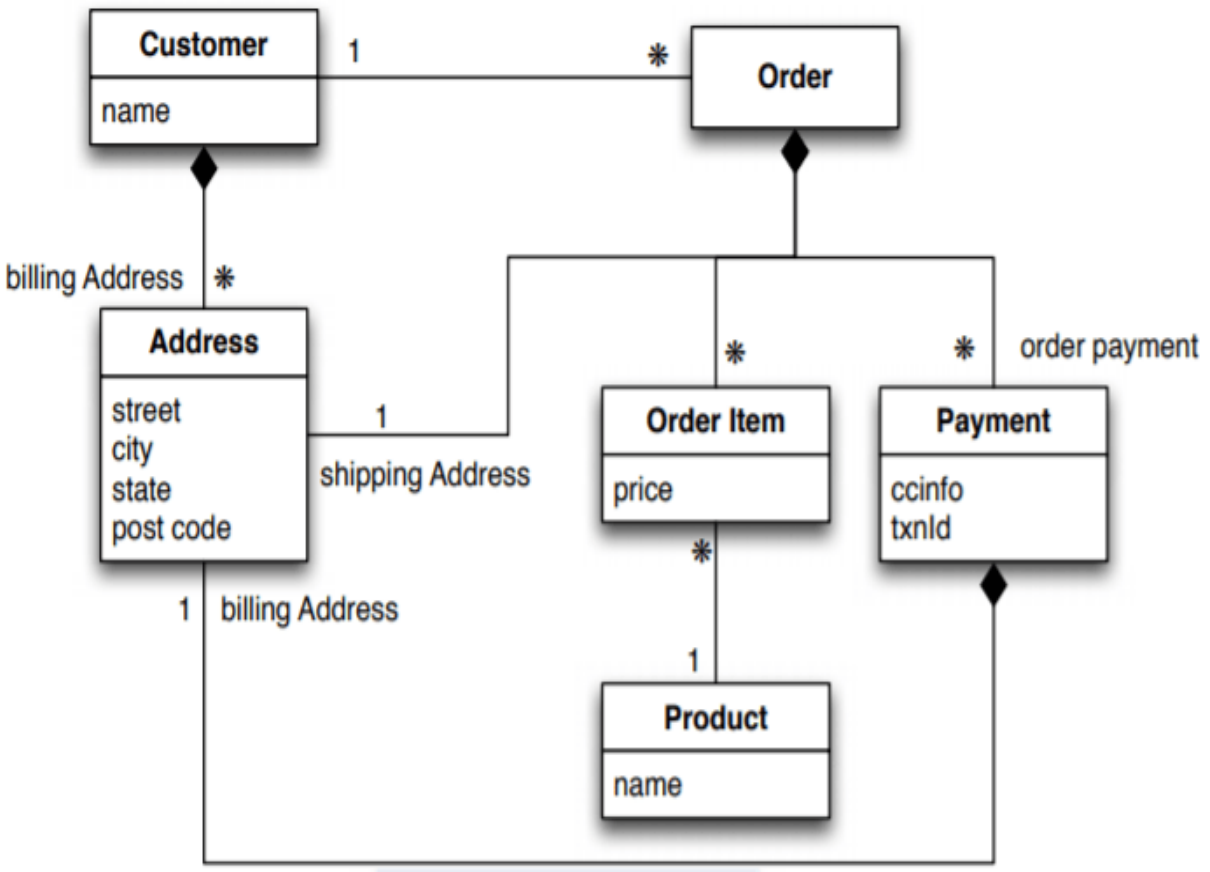
OrderPayment

Id	OrderId	CardNumber	BillingAddressId	txnId
33	99	1000-1000	55	abelif879rft

Lets see how this model looks when we think in more aggregate-oriented terms

Example of Aggregate Model

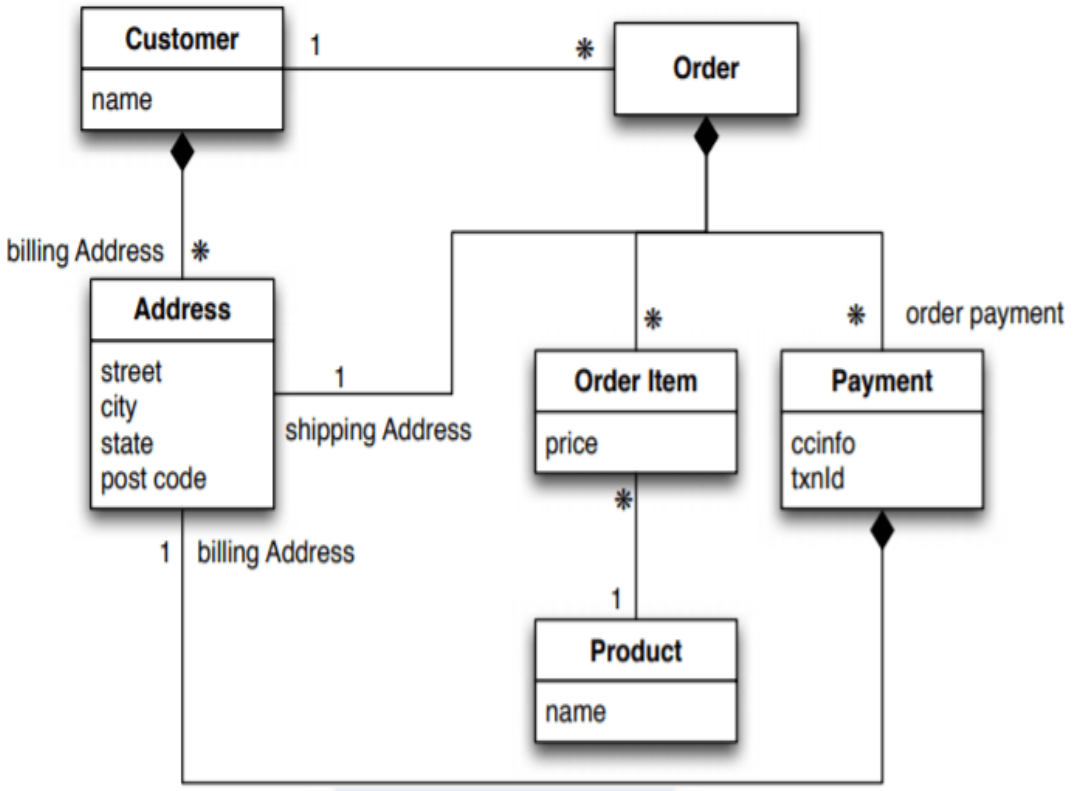
- We have two aggregates:
  - ✓ Customers and
  - ✓ Orders
- We us the black-diamond composition to show how data fits into the aggregate structure



An Aggregate Data Model

Example of Aggregate Model

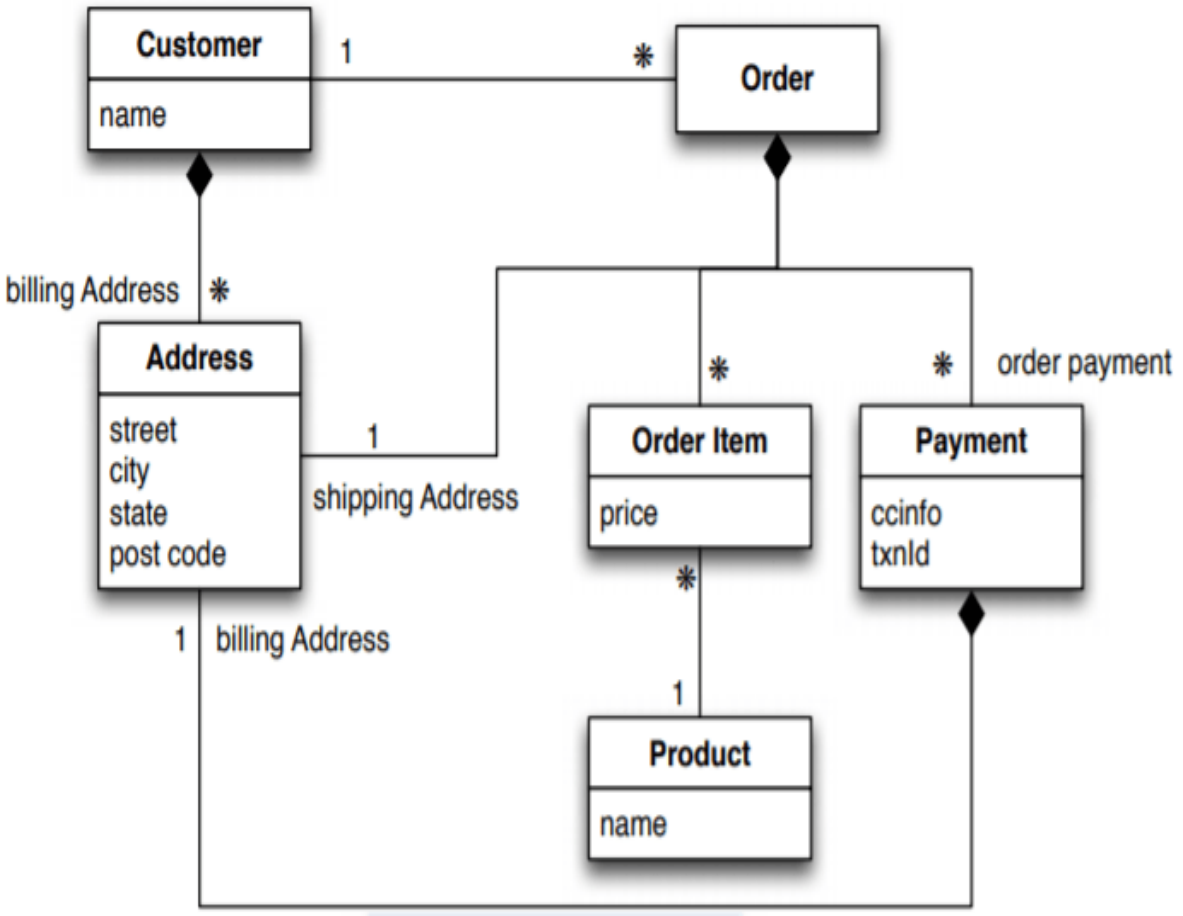
- The customer contains a list of billing addresses;
- The order contains a list of:
  - order items,
  - a shipping address,
  - and payments
- The payment itself contains a billing address for that payment



An Aggregate Data Model

Example of Aggregate Model

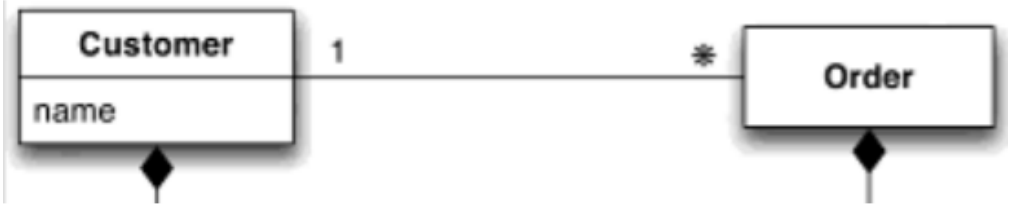
- A single logical address appears 3 times, but instead of using IDs it is copied each time.
- This fits a domain where we don't want shipping, payment and billing address to change.



An Aggregate Data Model

Example of Aggregate Model

- The link between the customer and the order is a relationship between aggregates

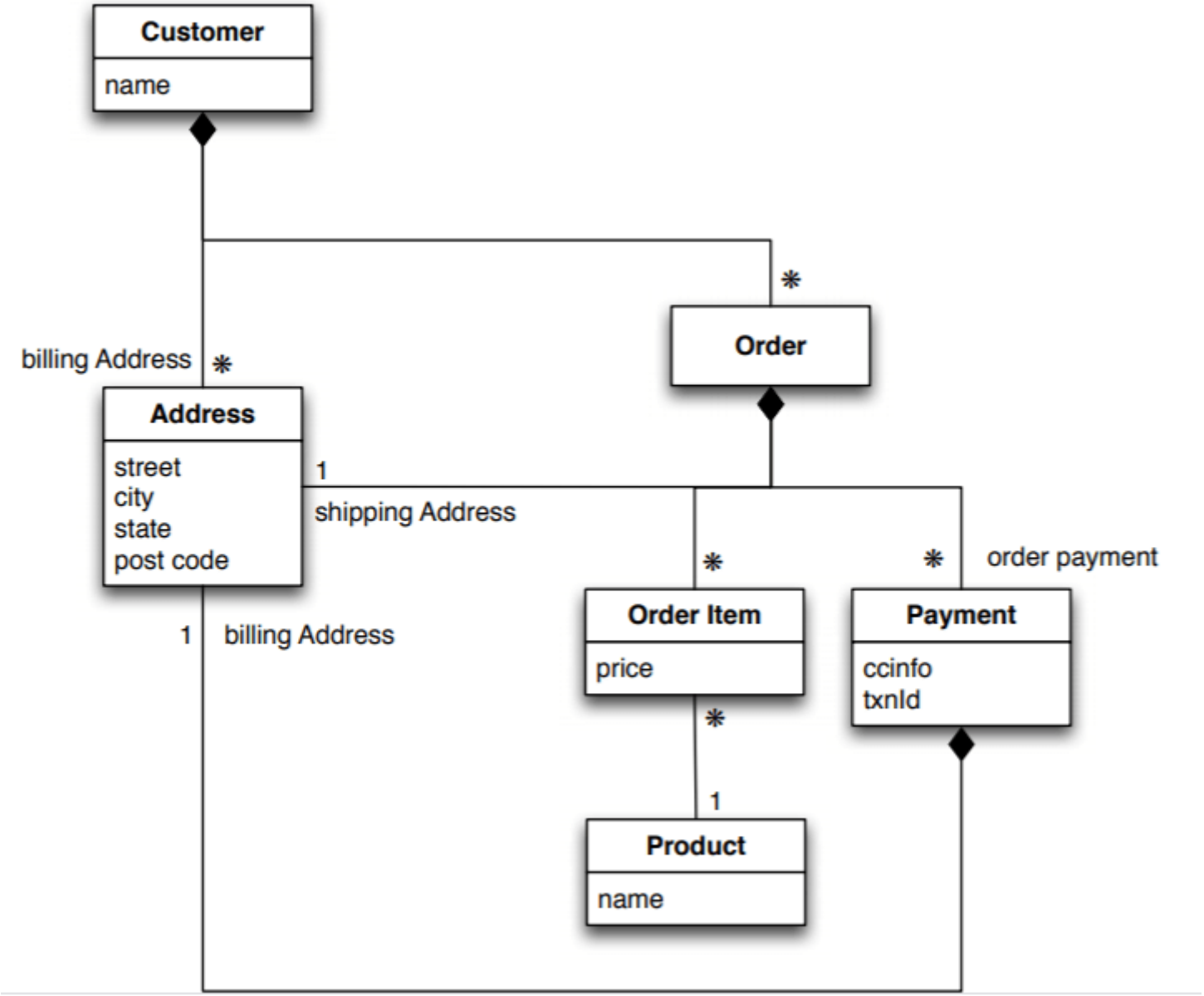



```
//Customer
 {
      "id": 1,
      "name":
"Fabio",

"billingAddress": [
       {
            "city":
"Bari"
       }
    ]
}
```

```
//Orders
{
        "id": 99,
        "customerId": 1,
        "orderItems": [
       {
            "productId": 27,
             "price": 34,
             "productName": “NoSQL
Distilled”
        } ],
```

```
"shippingAddress": [
 {
      "city": "Bari”} ],
"orderPayment": [
{ "ccinfo": "100-432423-545-134",
  "txnId": "afdfsdfsd",
 "billingAddress": [ {"city":
“Chicago” }]
} ]
}
```



Embed all the objects for customer and the customer's order

Orders Details

```
Orders
{
       "id": 99,
       "customerId": 1,
      "orderItems": [
      {
           "productId": 27,
           "price": 34,
           "productName": “NoSQL
Distilled” } ],
      "shippingAddress": [ {"city":
“Chicago”} ], "orderPayment": [
   {    "ccinfo": "100-432423-545-134",
       "txnId": "afdfsdfsd",
        "billingAddress": [ {"city":
“Chicago” }]
     } ]
 }
```

- There is the customer id
- The product name is a part of the ordered Items.
- The product id is part of the ordered items.
- The address is stored several times.

CONSEQUENCES OF AGGREGATE MODELS

CONSEQUENCES OF AGGREGATE MODELS

- Relational mapping captures various data elements and their relationships.
- Order consist of
  - Oreder items, a shipping address, and a payment
- All can be expresses in the relational model in terms of foreign key relationships.

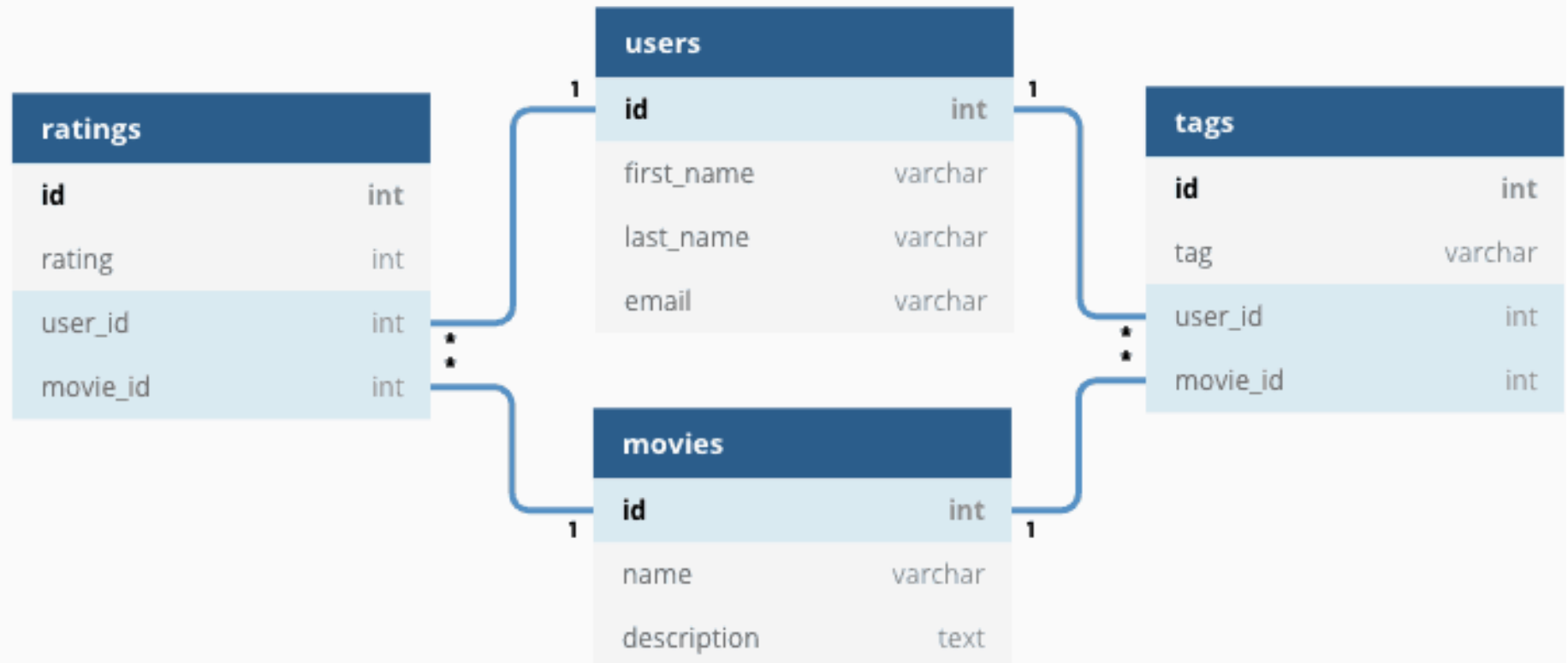
Important

The database can't use a knowledge of aggregate structure to help it store and distribute the data.

Marking Aggregate Tools

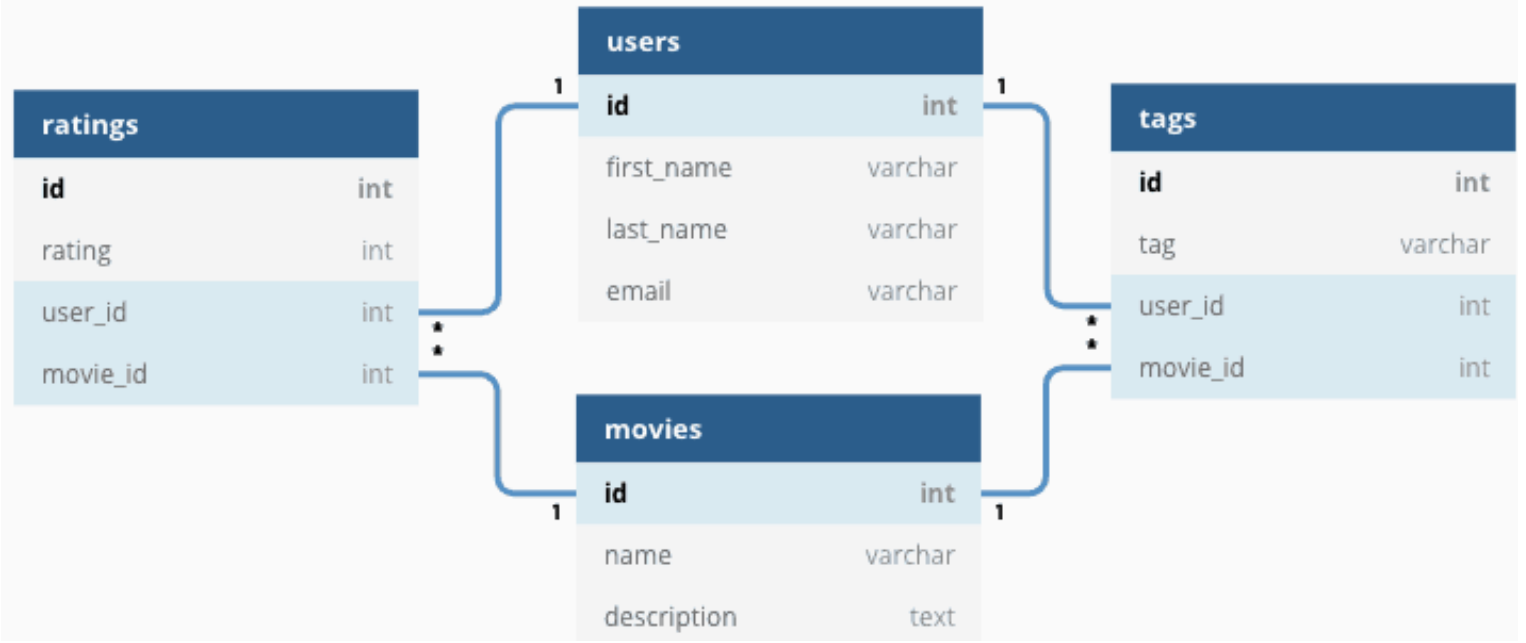
- Many data modeling techniques provides way to mark aggregate structures in relational models.
- However, they do not provide semantic that helps in distinguish relationships.
- When working with aggregate-oriented databases, we have a clear views of the semantic of the data.
- We can focus on the unit of interaction with the data storage.

Aggregate Ignorant



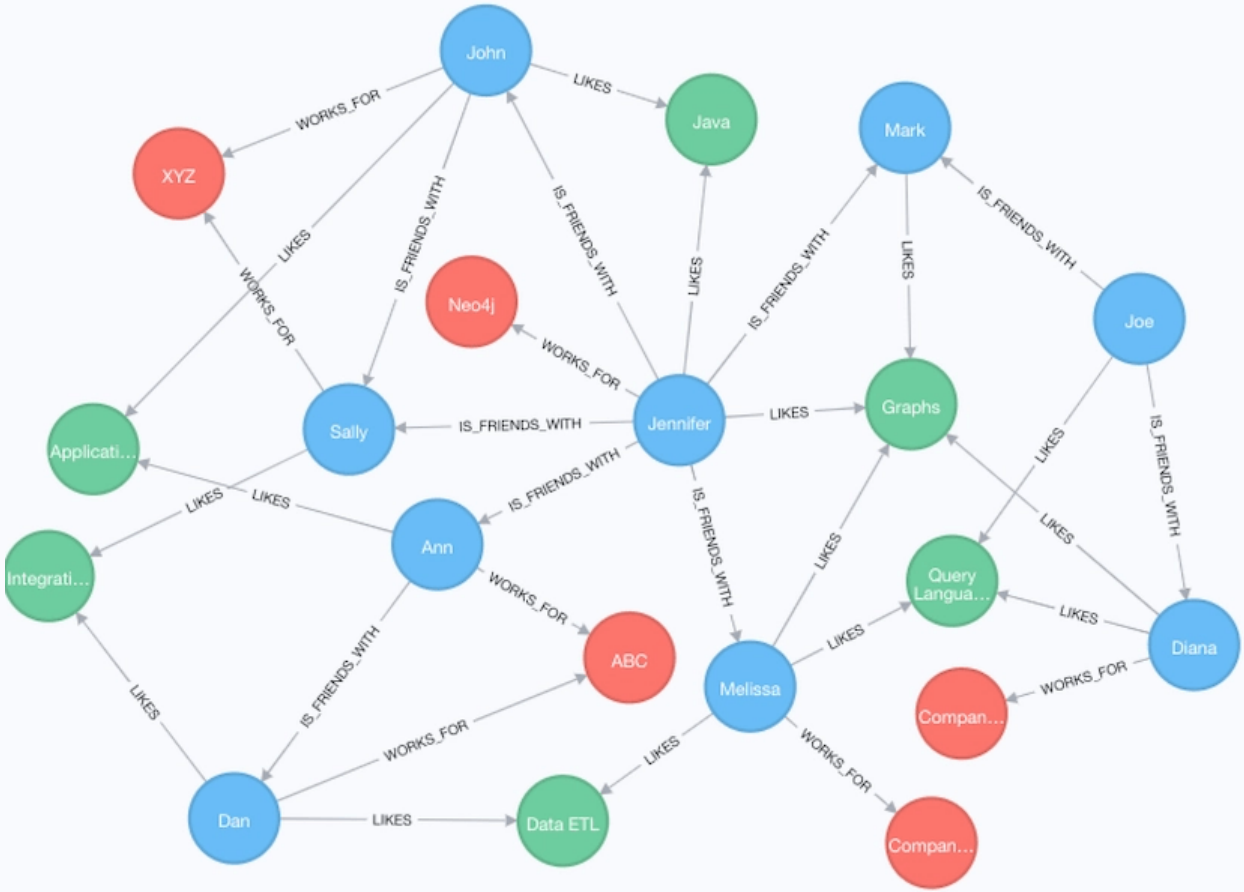
Relational Database

Aggregate Ignorant



Relational Database

Aggregate Ignorant



Graph Database are aggregate-ignorant

Aggregate and Operations

- An order is a good aggregate when:
  - A customer is making and reviewing an order, and – When the retailer is processing orders ▪
- However, when the retailer want to analyze its product sales over the last months, then aggregate are trouble. ▪
- We need to analyze each aggregate to extract sales history.

Aggregate and Operations

- Aggregate may help in some operation and not in others.
- In cases where there is not a clear view aggregate ignorant database are the best option. ▪
- But, remember the point that drove us to aggregate models (cluster distribution).
- Running databases on a cluster is need when dealing with huge quantities of data.

What is the clinching reason for aggregate orientation?

Running on a Cluster

- It gives several advantages on computation power and data distribution
- However, it requires to minimize the number of nodes to query when gathering data.
- By explicitly including aggregates, we give the database an important of which information should be stored together .

How Aggregates plays an important consequence for transactions?

A — Atomic

All changes to the data must be performed successfully or not at all

C — Consistent

Data must be in a consistent state before and after the transaction

I — Isolated

No other process can change the data while the transaction is running

D — Durable

The changes made by a transaction must persist

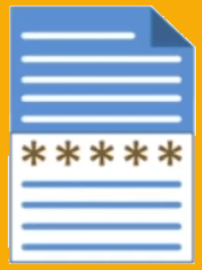
Why data consistency is not required?

Types of NoSQL databases

Key Value

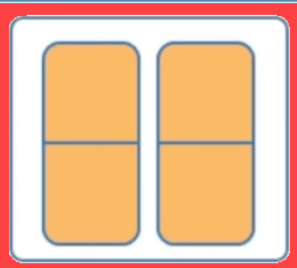
Example:
Riak, Tokyo Cabinet, Redis server, Memcached, Scalaris

Document-Based



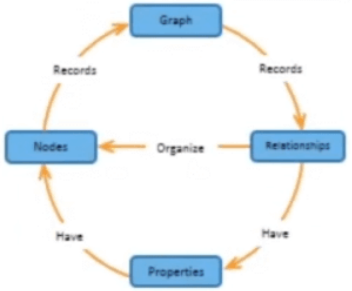
Example:
MongoDB, CouchDB, OrientDB, RavenDB

Column-Based



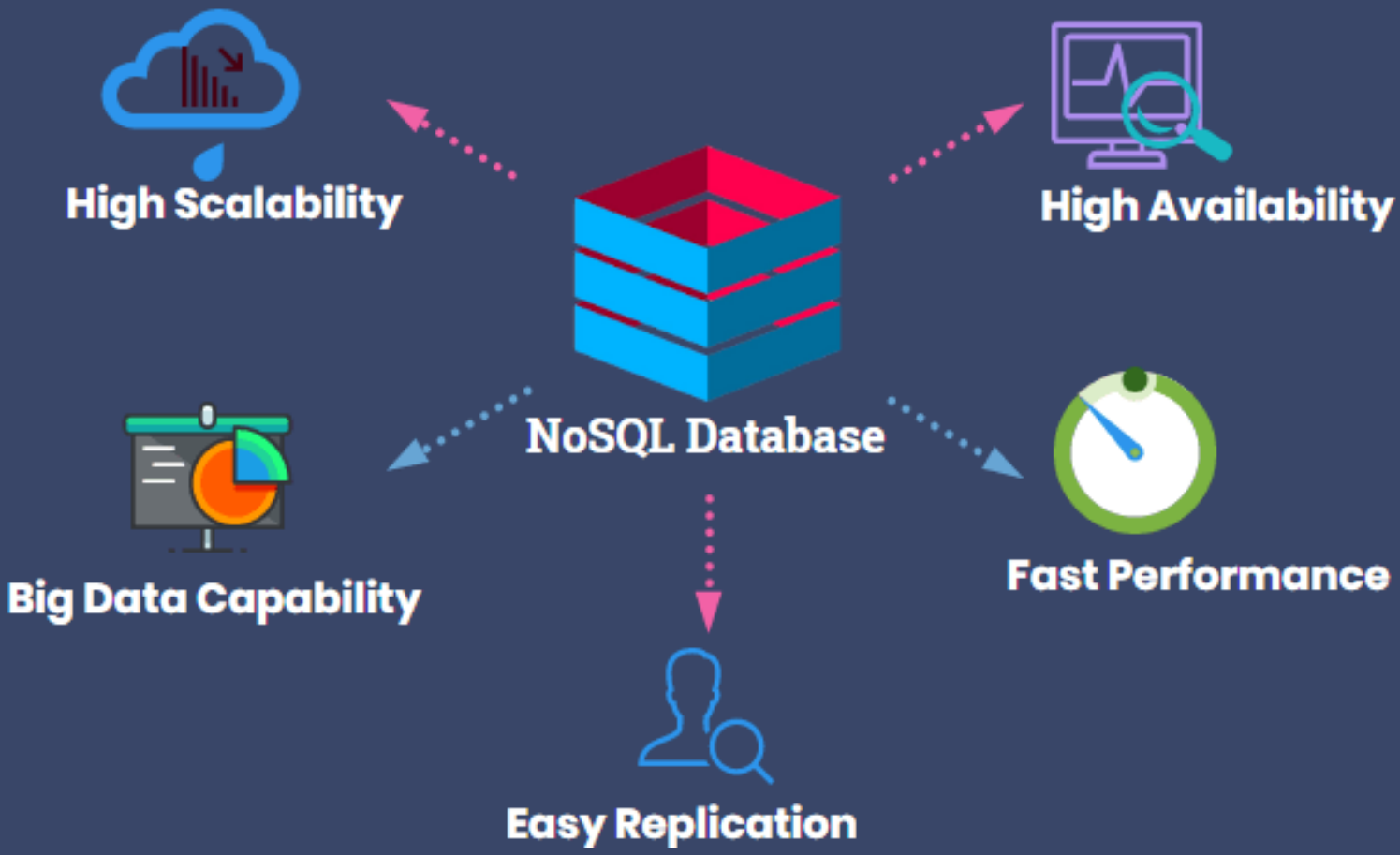
Example:
BigTable, Cassandra, Hbase, Hypertable

Graph-Based

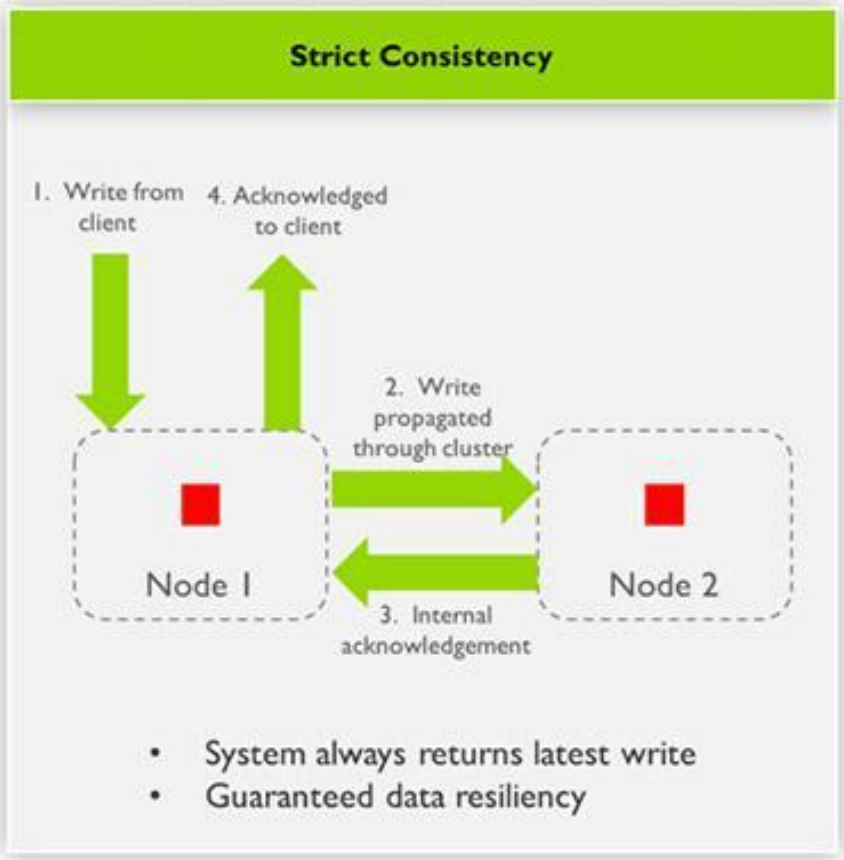


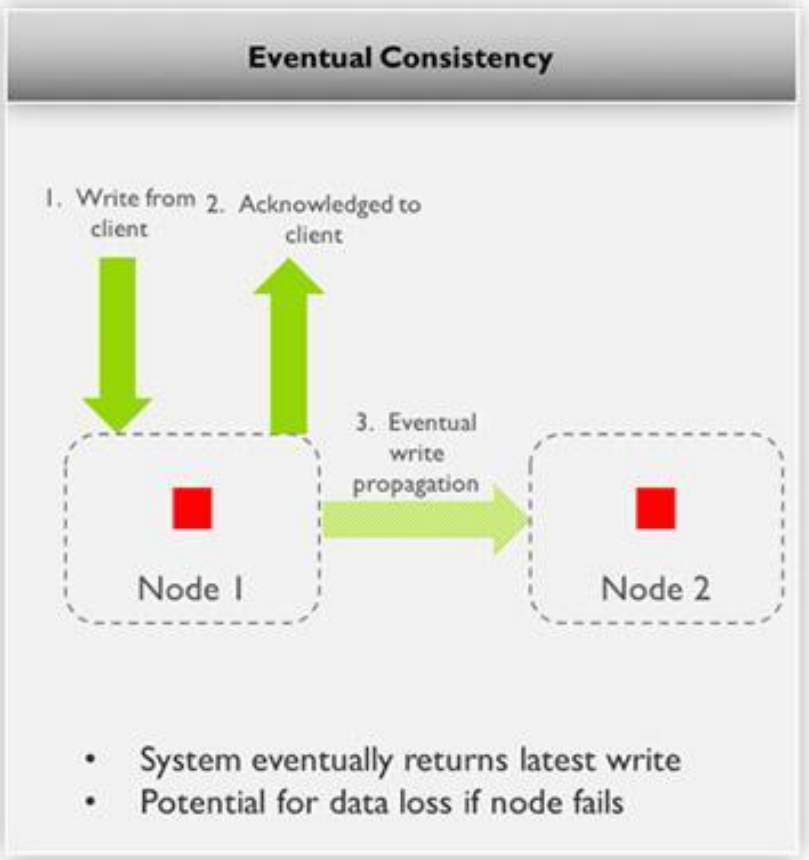
Example:
Neo4J, InfoGrid, Infinite Graph, Flock DB

NoSQL databases



Types of NoSQL databases





NoSQL databases

Bank account data must be consistent whenever any updates are made to data.

NoSQL databases

Online multiplayer gaming applications usually store profile data of a large number of users that require strong consistency.


That's all for now...