uc3m | Universidad Carlos III de Madrid

OpenCourseWare

# Database

## 3.1 NoSQL DB types

Lourdes Moreno López
Paloma Martínez Fernández
José Luis Martínez Fernández
Rodrigo Alarcón Garcia

HULAT Human Language & Accessibility Technologies

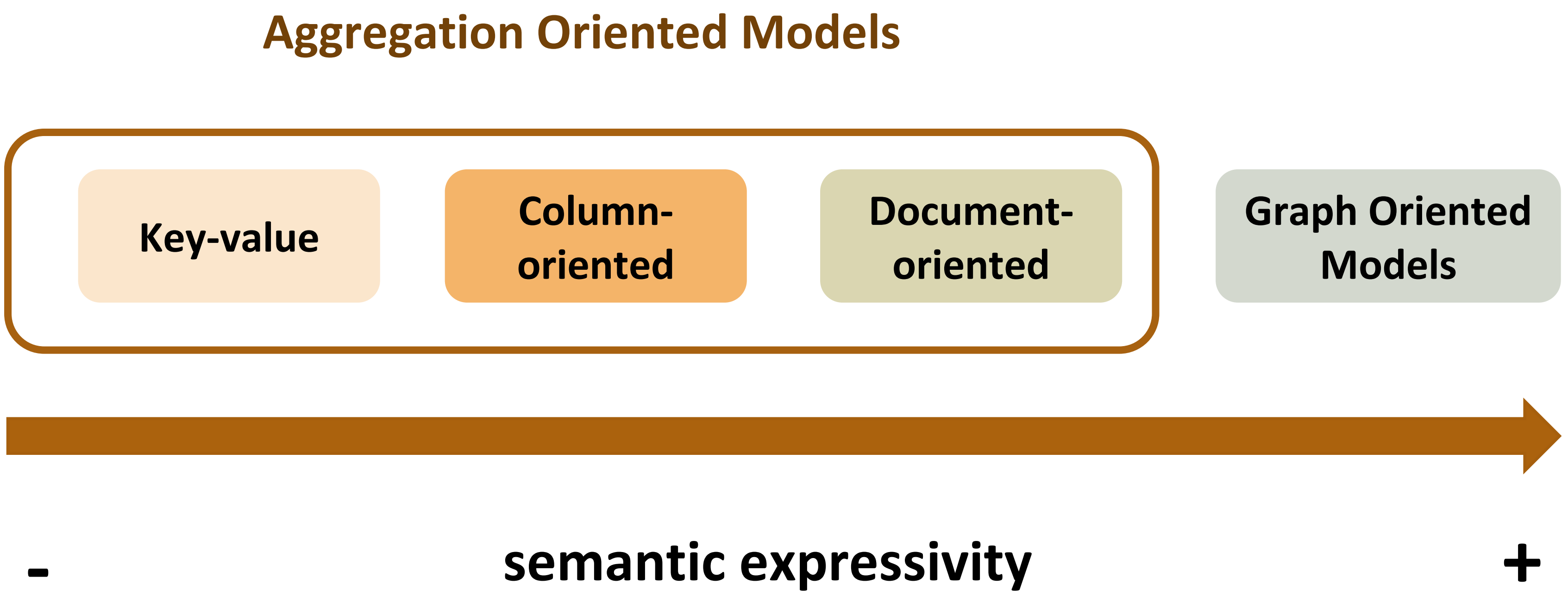
# CONTENTS

- Types of NoSQL DB
  - Aggregation Oriented Models
  - Graph Oriented Models

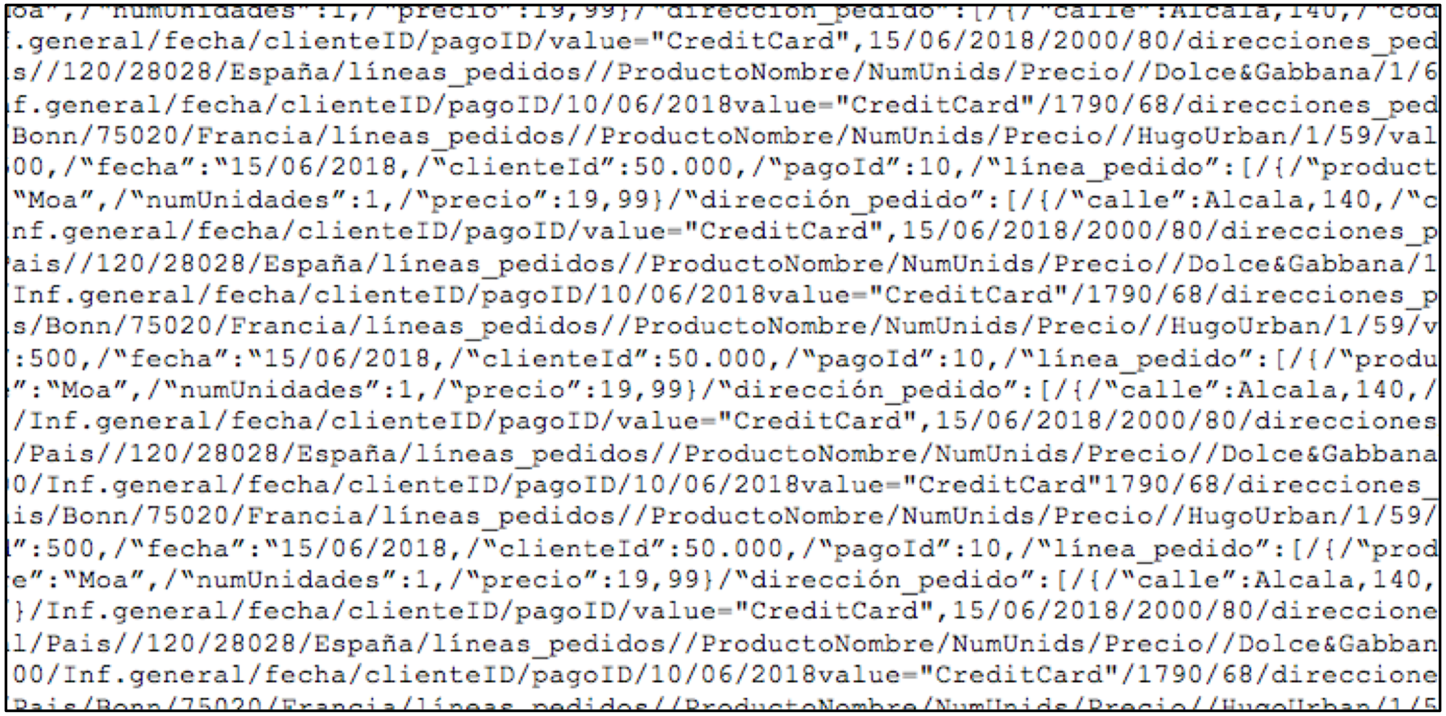
## Learning objectives

- The student must be able to:
  - Identify the situations where using NoSQL DB is a solution.
  - Know the main features of the different data models of the NoSQL DB

# NoSQL DB TYPEs

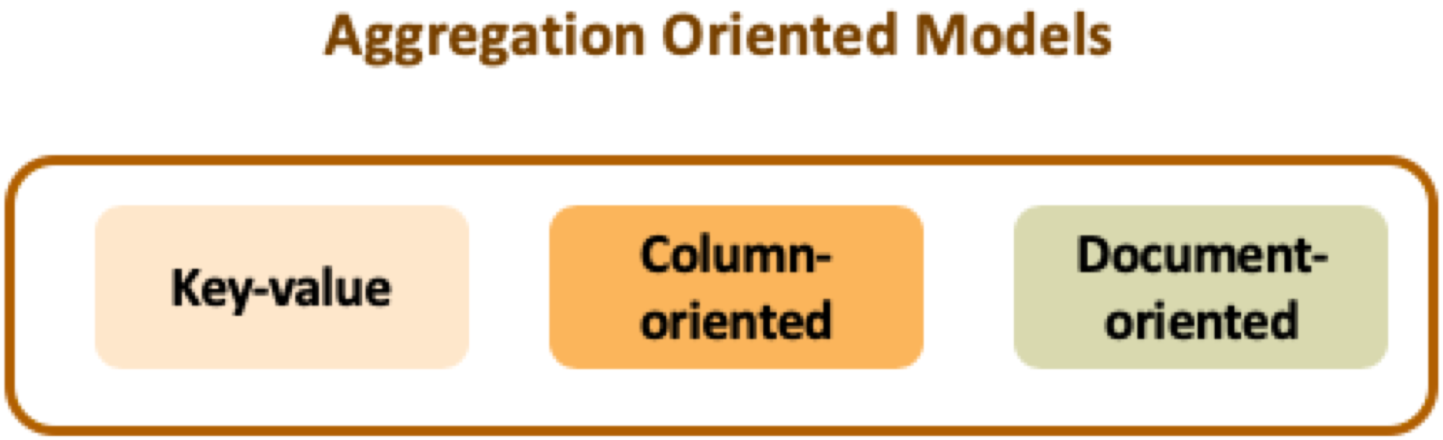


# AGGREGATION ORIENTED MODELS
# Example: Order Management



```
oa",/"numUnidades":1,/"precio":19,99}/"dirección_pedido":[/{/"calle":Alcala,140,/"cod
.general/fecha/clienteID/pagoID/value="CreditCard",15/06/2018/2000/80/direcciones_ped
s//120/28028/España/líneas_pedidos//ProductoNombre/NumUnids/Precio//Dolce&Gabbana/1/6
f.general/fecha/clienteID/pagoID/10/06/2018value="CreditCard"/1790/68/direcciones_ped
Bonn/75020/Francia/líneas_pedidos//ProductoNombre/NumUnids/Precio//HugoUrban/1/59/val
00,/"fecha":"15/06/2018,/"clienteId":50.000,/"pagoId":10,/"línea_pedido":[/{/"product
"Moa",/"numUnidades":1,/"precio":19,99}/"dirección_pedido":[/{/"calle":Alcala,140,/"c
nf.general/fecha/clienteID/pagoID/value="CreditCard",15/06/2018/2000/80/direcciones_p
ais//120/28028/España/líneas_pedidos//ProductoNombre/NumUnids/Precio//Dolce&Gabbana/1
Inf.general/fecha/clienteID/pagoID/10/06/2018value="CreditCard"/1790/68/direcciones_p
s/Bonn/75020/Francia/líneas_pedidos//ProductoNombre/NumUnids/Precio//HugoUrban/1/59/v
:500,/"fecha":"15/06/2018,/"clienteId":50.000,/"pagoId":10,/"línea_pedido":[/{/"produ
":"Moa",/"numUnidades":1,/"precio":19,99}/"dirección_pedido":[/{/"calle":Alcala,140,/
/Inf.general/fecha/clienteID/pagoID/value="CreditCard",15/06/2018/2000/80/direcciones
/Pais//120/28028/España/líneas_pedidos//ProductoNombre/NumUnids/Precio//Dolce&Gabbana
0/Inf.general/fecha/clienteID/pagoID/10/06/2018value="CreditCard"1790/68/direcciones_
is/Bonn/75020/Francia/líneas_pedidos//ProductoNombre/NumUnids/Precio//HugoUrban/1/59/
":500,/"fecha":"15/06/2018,/"clienteId":50.000,/"pagoId":10,/"línea_pedido":[/{/"prod
e":"Moa",/"numUnidades":1,/"precio":19,99}/"dirección_pedido":[/{/"calle":Alcala,140,
}/Inf.general/fecha/clienteID/pagoID/value="CreditCard",15/06/2018/2000/80/direccione
l/Pais//120/28028/España/líneas_pedidos//ProductoNombre/NumUnids/Precio//Dolce&Gabban
00/Inf.general/fecha/clienteID/pagoID/10/06/2018value="CreditCard"/1790/68/direccione
Pais/Bonn/75020/Francia/líneas_pedidos//ProductoNombre/NumUnids/Precio//HugoUrban/1/5
```

# AGGREGATION ORIENTED MODELS



## AGGREGATION ORIENTED MODELS
### Aggregate

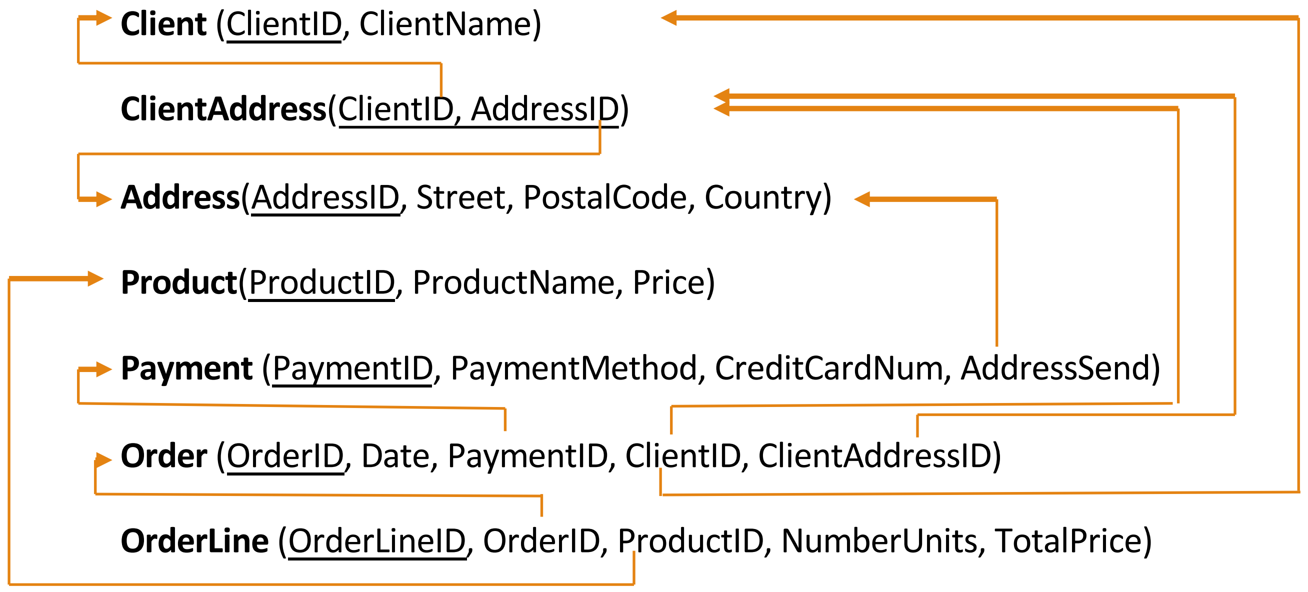
- Set of real-world objects that are interrelated and that are treated as an indivisible data unit for access and manipulation purposes
  - It is the minimum unit of exchange between application programs
  - It is the minimum unit for the purpose of concurrency control and integrity of the DB
  - Any changes made to the aggregate will be finalized on the DB.
  - Each aggregate is identified by a key

## AGGREGATION ORIENTED MODELS
## Aggregate

- The design of aggregates must be guided by the functionalities
- It helps reduce access in queries
- It is a solution when:
    - The functionality set a priori does not usually have future changes
    - There are no complex interrelations
    - Data is subject to little change (insertion / query but no update)

# AGGREGATION ORIENTED MODELS
## Example: Order Management

if we want to retrieve the information of the order to manage the shipment => we will have to join the relationships Order, OrderLine, Product and Address.



**Client** (ClientID, ClientName)

**ClientAddress**(ClientID, AddressID)

**Address**(AddressID, Street, PostalCode, Country)

**Product**(ProductID, ProductName, Price)

**Payment** (PaymentID, PaymentMethod, CreditCardNum, AddressSend)

**Order** (OrderID, Date, PaymentID, ClientID, ClientAddressID)

**OrderLine** (OrderLineID, OrderID, ProductID, NumberUnits, TotalPrice)

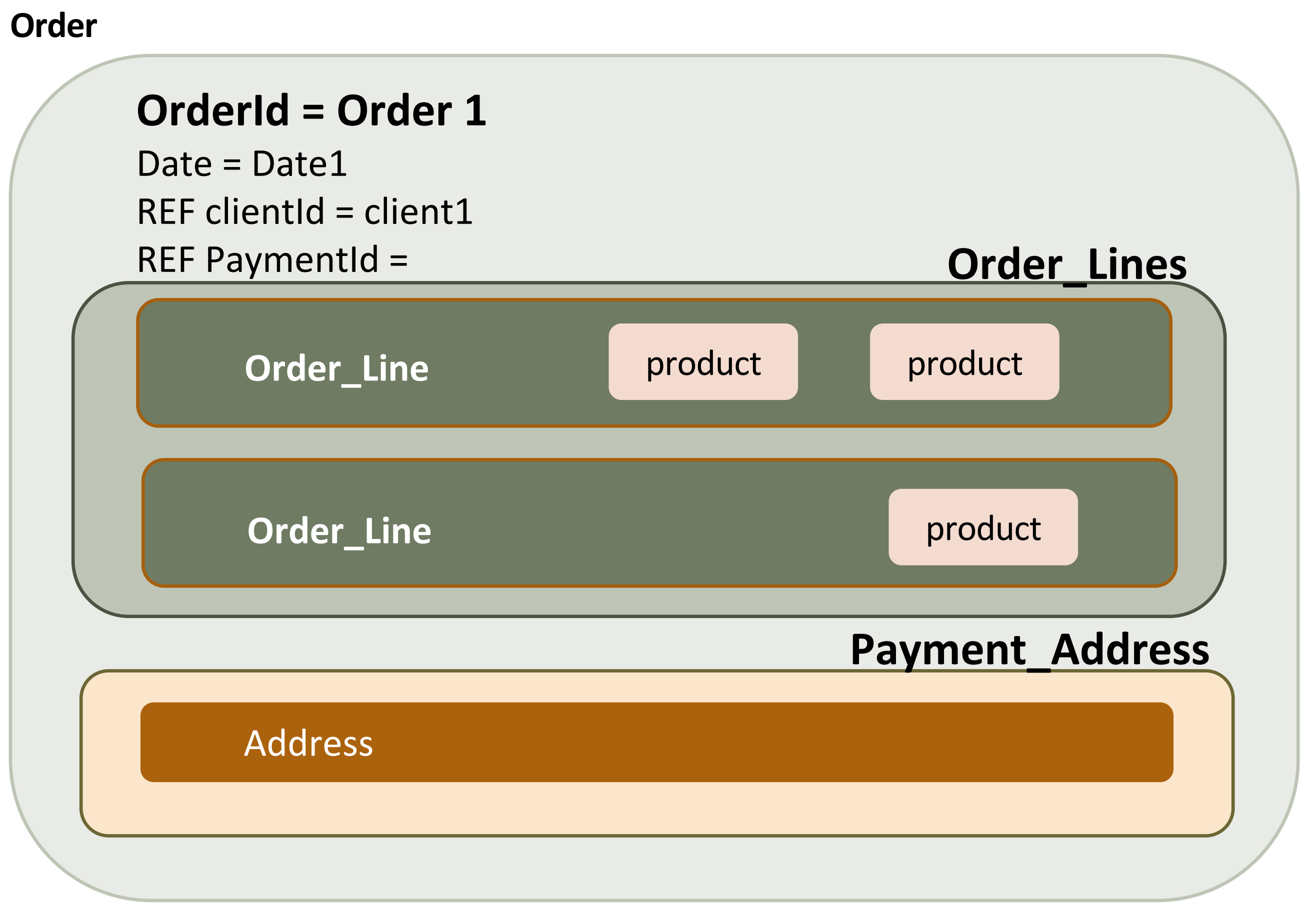
# AGGREGATION ORIENTED MODELS
# Example: Order Management

- if we want to retrieve the information of the order to manage the shipment => we will have to join the relationships Order, OrderLine, Product and Address.
- This can be done with a relational database if it is a centralized database, with little data, however, if we have a large volume of data and the database is distributed, doing this join can be very complex.

⇒A good option is to take the data that interests us, save it in aggregate with all the information that interests the client in a single object (aggregate), so when we want to retrieve the data, we will recover that aggregate and respond to the request.

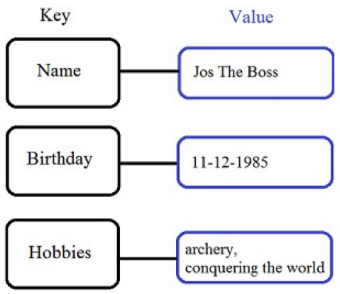
# AGGREGATION ORIENTED MODELS
# Example: Order Management

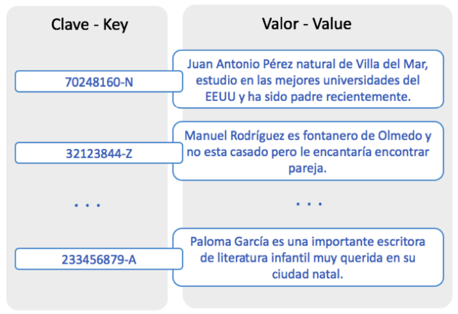
- Retrieve the information of the order to manage the shipping
- Store the data together about Order, OrderLine, Product, and Address.



## AGGREGATION ORIENTED MODELS
## Key-value data model

- The key-value model is the simplest model
- Less semantic expressiveness
- Each element is uniquely identified by a key
  - **(Id, Aggregate) == (Key, Value)**
- The key can be of the domain (DNI, NSS, e-mail, …) or not
- Atomicity at the key level

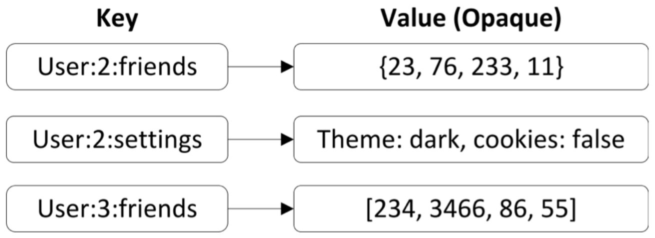




| Clave - Key | Valor - Value |
|---|---|
| 70248160-N | Juan Antonio Pérez natural de Villa del Mar, estudio en las mejores universidades del EEUU y ha sido padre recientemente. |
| 32123844-Z | Manuel Rodríguez es fontanero de Olmedo y no esta casado pero le encantaría encontrar pareja. |
| . . . | . . . |
| 233456879-A | Paloma García es una importante escritora de literatura infantil muy querida en su ciudad natal. |

## AGGREGATION ORIENTED MODELS
## Key-value data model

- The DB does not know the structure of the aggregate (black box)
  - The aggregate as an opaque object
  - If there is a structure for the object, it will be known only by the application programs that access the DB

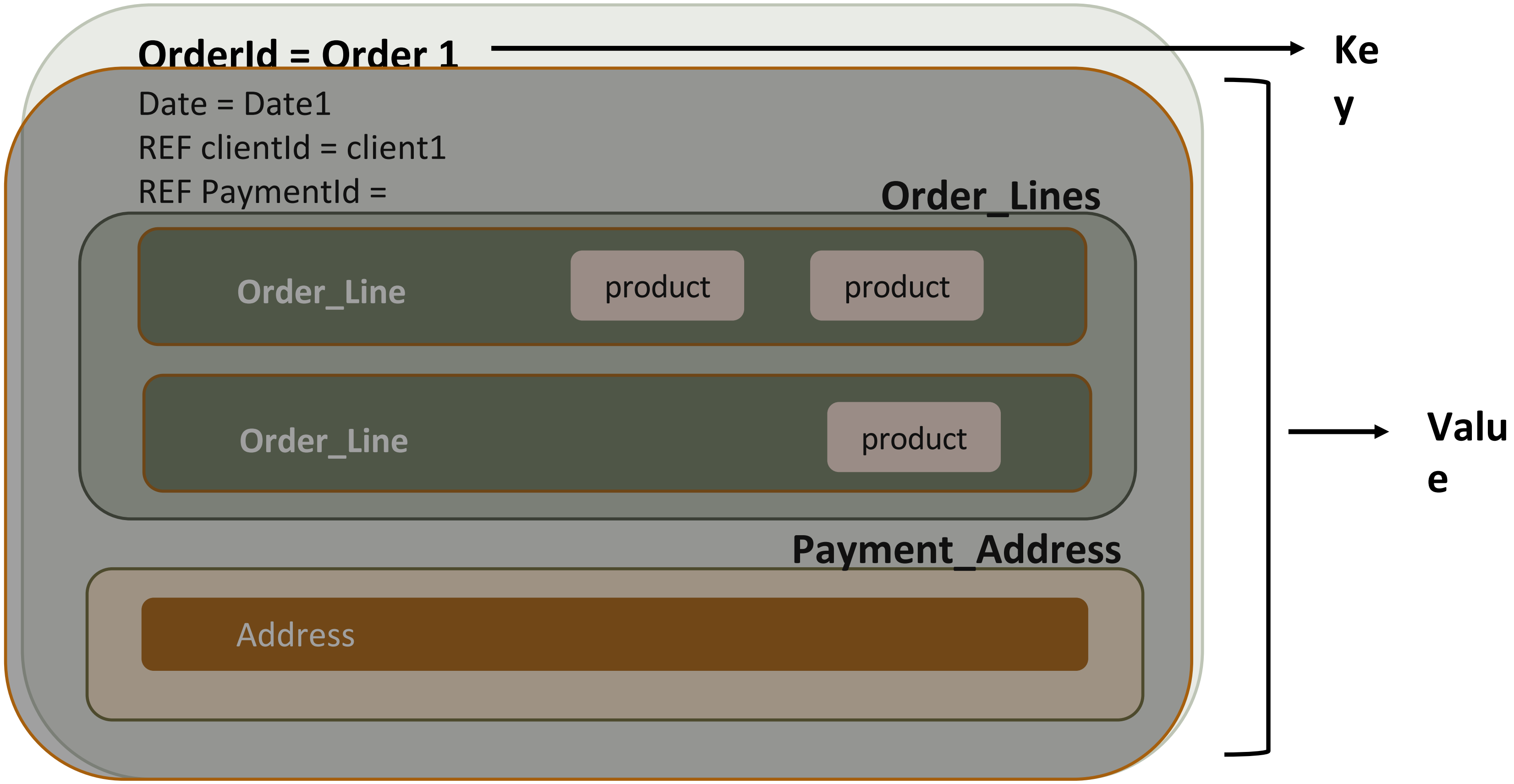


## AGGREGATION ORIENTED MODELS
## Key-value data model

- High performance of read / write
- A lot of speed in the consultations
- Easy to climb
- Easy to implement

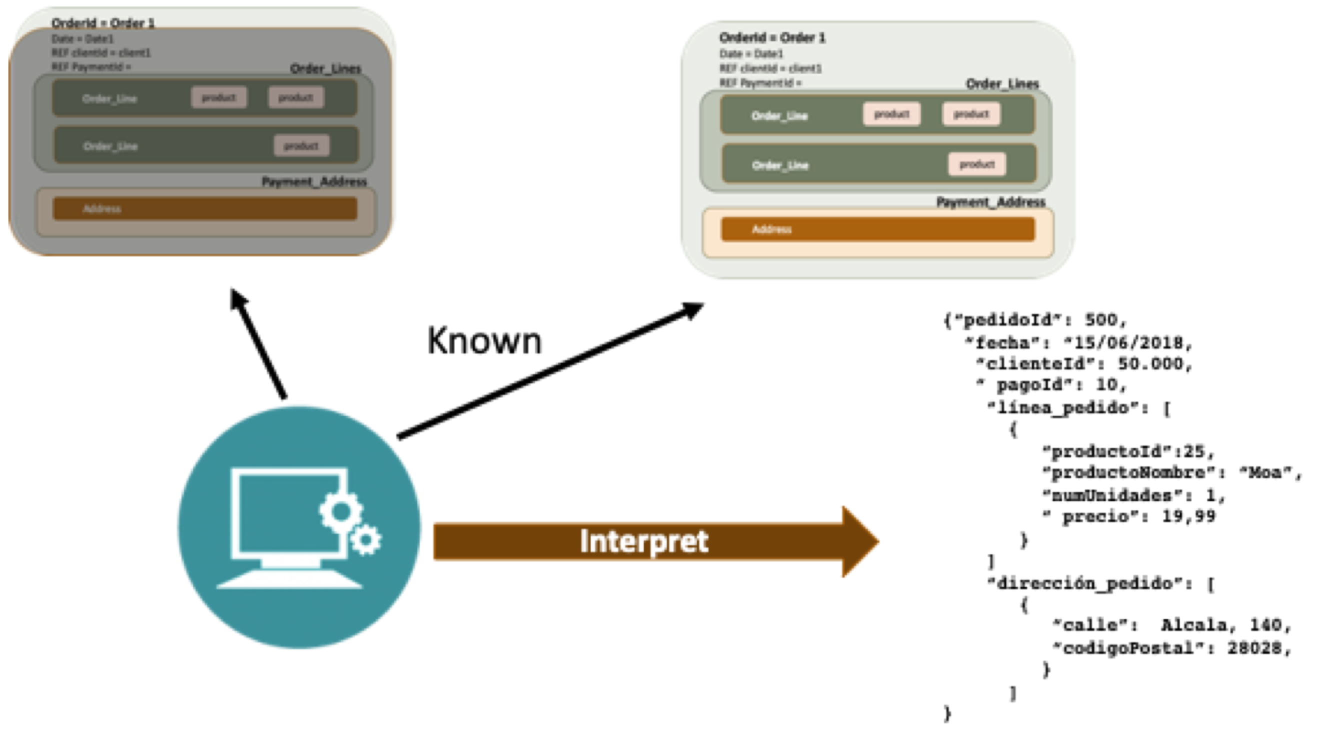
# AGGREGATION ORIENTED MODELS / Key-value data model

Example: Order Management



# AGGREGATION ORIENTED MODELS
# Key-value data model

- It is the application that gives structure to the "Value" and interprets it (for example, with an XML)



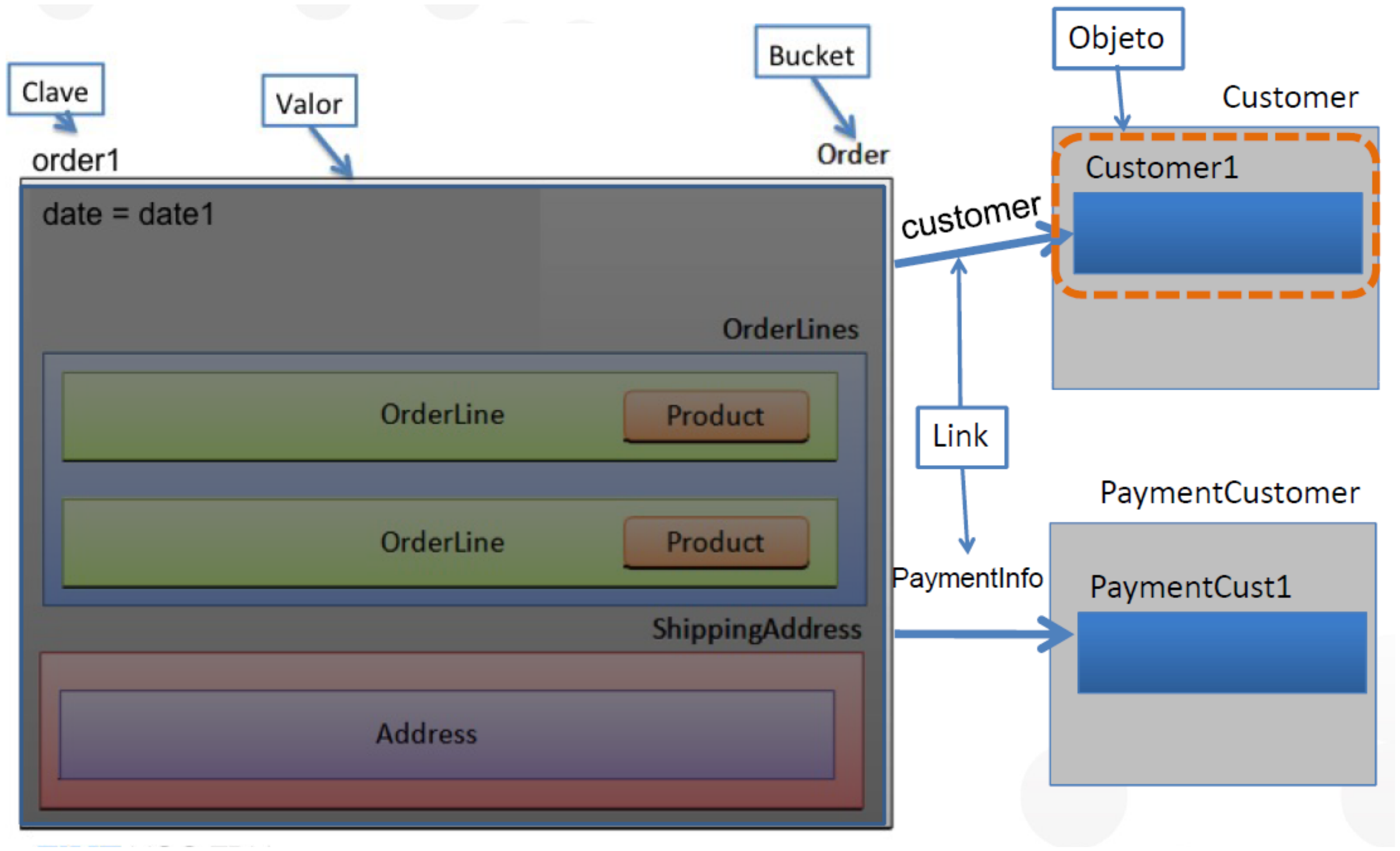
```
{"pedidoId": 500,
  "fecha": "15/06/2018,
   "clienteId": 50.000,
   " pagoId": 10,
    "línea_pedido": [
      {
         "productoId":25,
         "productoNombre": "Moa",
         "numUnidades": 1,
         " precio": 19,99
       }
    ]
    "dirección_pedido": [
       {
         "calle":  Alcala, 140,
         "codigoPostal": 28028,
       }
     ]
}
```

## AGGREGATION ORIENTED MODELS
## Key-value data model

| Relational Model |
|---|
| Database |
| Table, view |
| Row |
| Column |
| Primary key |
| Foreign Key |

| Riak |
|---|
| Database |
| Bucket |
| Object =<key, vaue> |
| has no direct equivalent |
| Key |
| Link |

# AGGREGATION ORIENTED MODELS
## Key-value data model



## AGGREGATION ORIENTED MODELS
## Key-value data model

- DBMS extend key-value models
  - They extend the semantics of aggregates and incorporate relationships between aggregates

riak

DynamoDB

amazon web services

redis

ORACLE

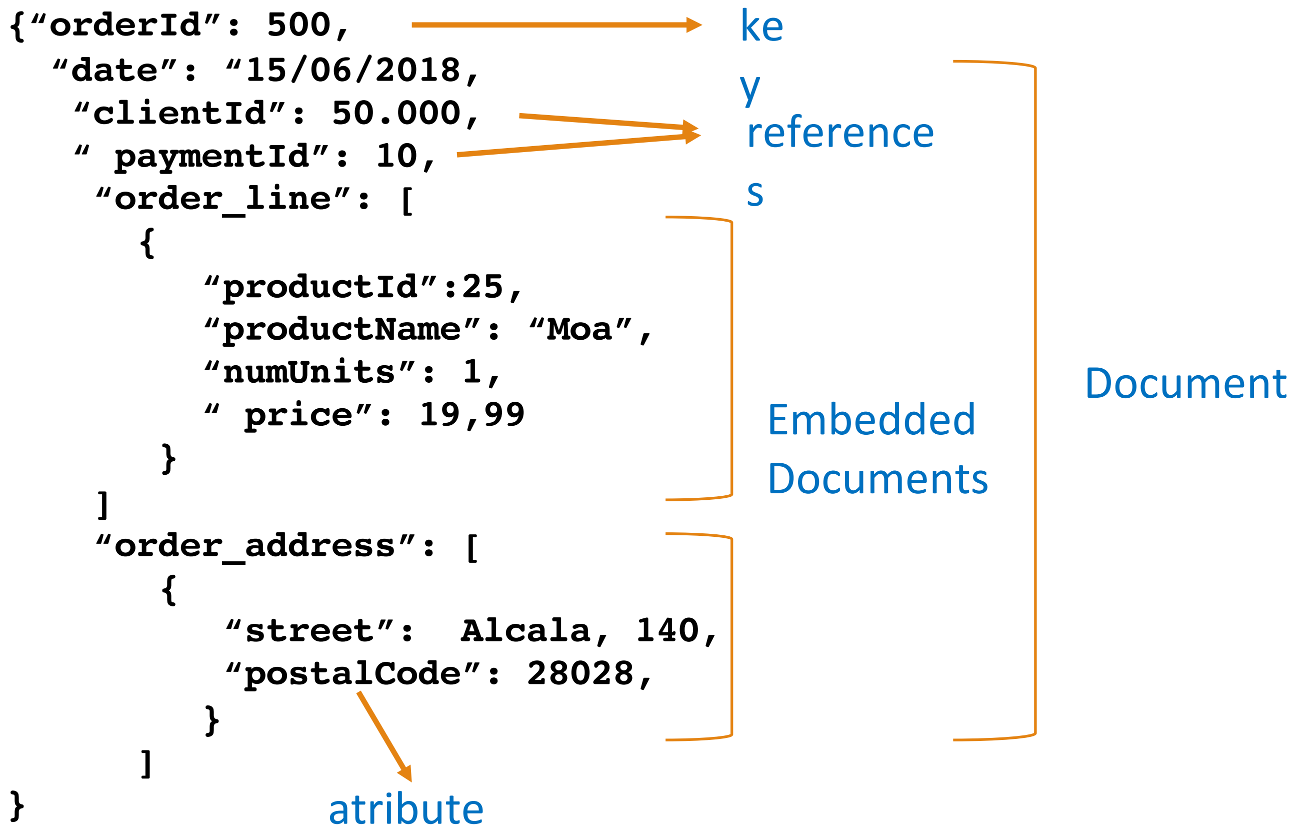
BERKELEY DB

## AGGREGATION ORIENTED MODELS
## Document-oriented data model

- Extension of the key-value model
- The aggregates have an internal "document" structure that is stored in JSON format, XML among others ..
- The DB knows how to interpret the internal structure
- The aggregate can be accessed:
  - through the key
  - to content through document attributes
- Documents can be added in collections
- You can retrieve, modify part of a document, and create indexes on attributes
- Atomicity at the document level

# AGGREGATION ORIENTED MODELS / Document-oriented data model

Example: Order Management

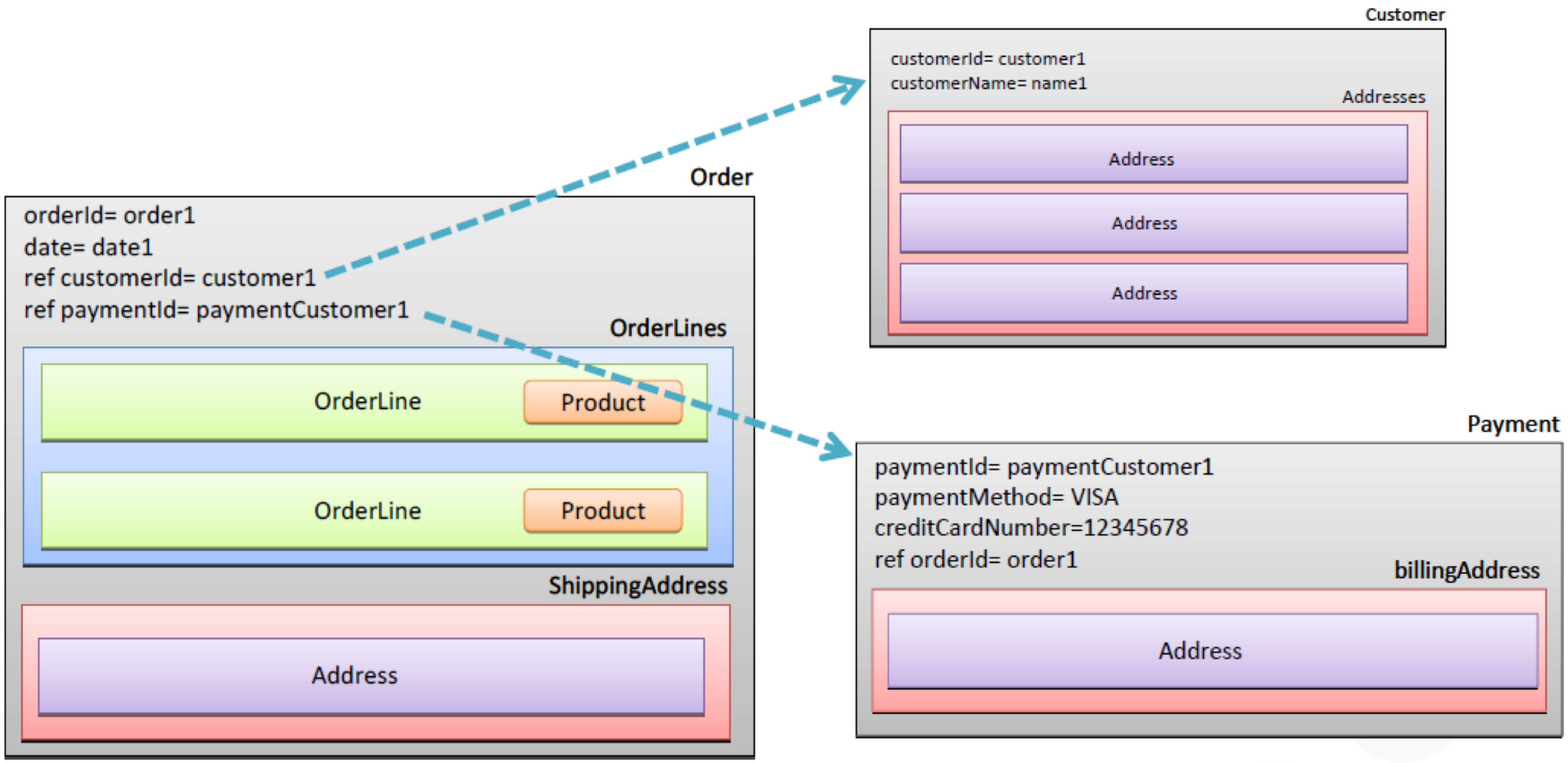


```
{“orderId”: 500,
  “date”: “15/06/2018,
   “clientId”: 50.000,
   “ paymentId”: 10,
    “order_line”: [
      {
         “productId”:25,
         “productName”: “Moa”,
         “numUnits”: 1,
         “ price”: 19,99
       }
    ]
    “order_address”: [
       {
          “street”:  Alcala, 140,
          “postalCode”: 28028,
         }
      ]
}
```

## AGGREGATION ORIENTED MODELS / Document-oriented data model

| Relational Model | MongoDB |
|---|---|
| Database | Database |
| Table, view | Collection |
| Row | Document |
| Column | Column |
| Primary key | _id |
| Foreign Key | Reference |
| Tables join | Embedded document |

# AGGREGATION ORIENTED MODELS / Document-oriented data model



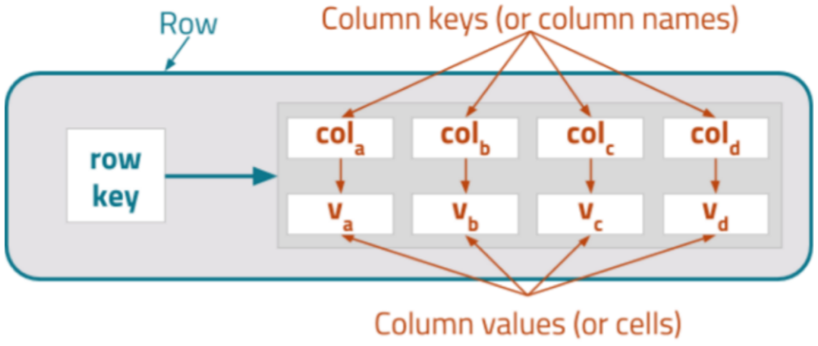
## AGGREGATION ORIENTED MODELS
## Document-oriented data model

# AGGREGATION ORIENTED MODELS
# Column-oriented data model

- The data is shown as a two-dimensional matrix
  - Rows: Data aggregations accessed by the key
  - Columns: Attributes of the aggregations represented by the triplet: **<name, value, timestamp>**
- Atomicity at the row-level
- Columns can be grouped into families (a family represents a concept of aggregation)
- Example: An aggregate "Teacher

with two families

  - Personal Information
  - academic information



# AGGREGATION ORIENTED MODELS
## Column-oriented data model

- They are suitable for applications with distributed files
- They are less scalable than DB key-values
- Slower in writing
- Reading speed
- They are more efficient when:
  - Inserting multiple records at the same time (column blocks are updated)
  - Access only to some columns

# AGGREGATION ORIENTED MODELS / Column-oriented data model

Example: Order Management

Retrieve the general information of all orders (the values of the column family for all rows)

**OrderID**

**10.097**

| General Inf. | | |
|---|---|---|
| **date** | **clientID** | **paymentD** |
| 15/06/2018 | 2000 | 80 |

| Orders_Address | | | |
|---|---|---|---|
| **street** | **city** | **postalCode** | **Country** |
| Alcala | 120 | 28028 | España |

| Orders_lines | | | |
|---|---|---|---|
| **productID** | **productName** | **numUnids** | **price** |
| 19 | Dolce & Gabbana | 1 | 68,50 |
| 45 | Loewe Aire | 2 | 74,00 |

**10.200**

| General Inf. | | |
|---|---|---|
| **date** | **clientID** | **paymentD** |
| 10/06/2018 | 1790 | 68 |

| Orders_Address | | |
|---|---|---|
| **street** | **postalCode** | **country** |
| Bonn | 75020 | Francia |

| Orders_lines | | | |
|---|---|---|---|
| **productID** | **productName** | **numUnids** | **price** |
| 8 | Hugo Urban | 1 | 59 |

**Retrieve order information number 10.200**

# AGGREGATION ORIENTED MODELS
## Column-oriented data model

cassandra

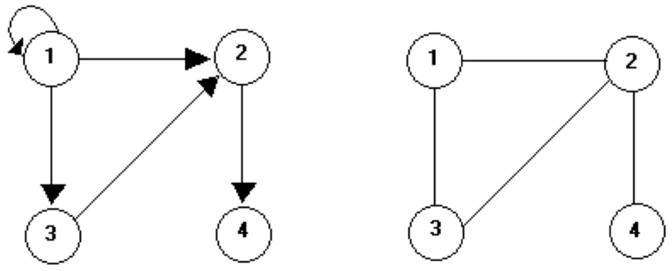
Amazon SimpleDB

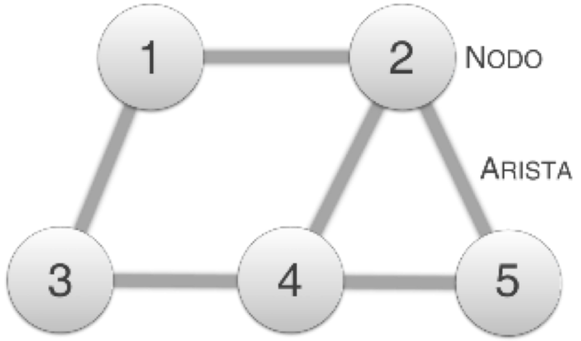
APACHE HBASE

HYPERTABLE INC

# GRAPH ORIENTED DATA MODEL

- The graph model uses graph structures to represent and store the data
- The graphs have two basic elements:
  - Nodes: represent real-world concepts and objects
  - Edges: explicitly represent the relationships between nodes
- Types: Directed, not directed



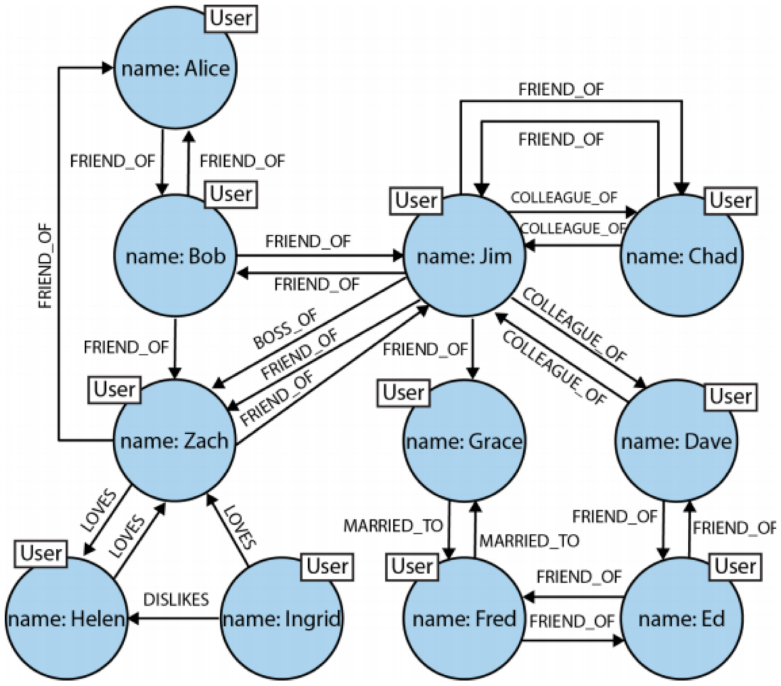


# GRAPH ORIENTED DATA MODEL

- Highly related data: Useful model when the importance of data is its interrelationships (there are few objects and many relationships)
- Useful when information can be represented as a network:
  - Networks (RRSS, logistics, maps, ...)
  - Semantic applications

# GRAPH ORIENTED DATA MODEL

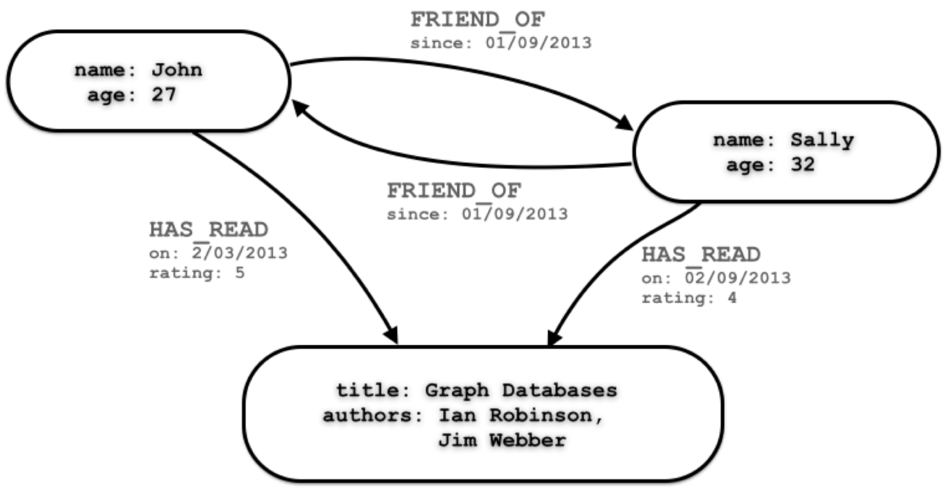
- **Tagged graphs:** Semantics are provided by assigning labels to nodes and edges



# GRAPH ORIENTED DATA MODEL

- **Property Graphs Tagged**

Sometimes tags may be insufficient => assign properties to nodes and edges (name: value)

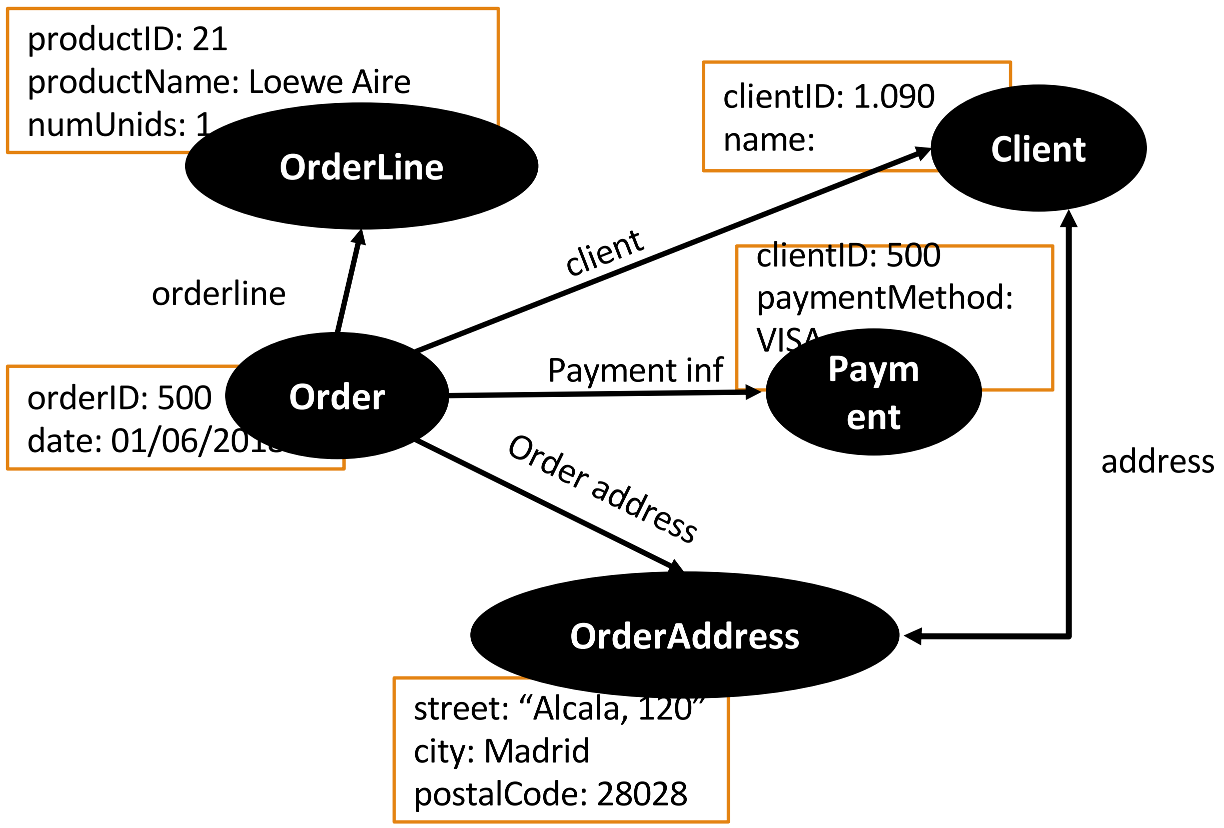


# GRAPH ORIENTED DATA MODEL

- Explicit relations: Improvement in response time in queries (navigation between relationships)
- They are not easily scalable
- The scheme is implicit in the structure of the graph
- They provide high-level languages
- Model constrains:
  - Edges without origin or destination nodes are not allowed
  - Nodes can only be removed when they are orphans

# GRAPH ORIENTED DATA MODEL

Example: Order Management



## GRAPH ORIENTED DATA MODEL

neo4j

blazegraph™

by SYSTAP

# REFERENCIAS

- Eric Redmond. Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement. 2012
- Harrison et al. Next Generation Databases: NoSQL, NewSQL, and Big Data. 2015. Apress.
- Kristina Chodorow. MongoDB: The Definitive Guide. 2013. O'Reilly Media, Inc.
- Ian Robinson, Jim Webber, and Emil Eifrém. Graph Databases 2nd Edition. 2015. O'Reilly Media