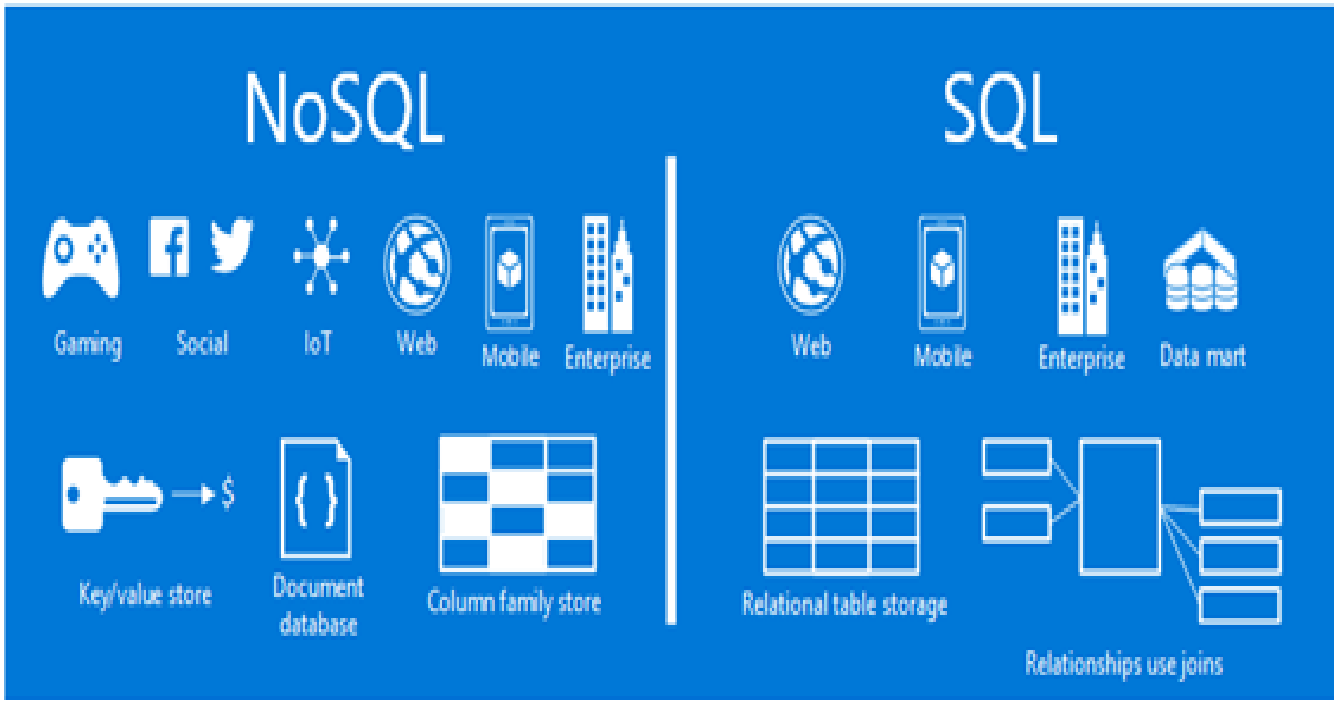
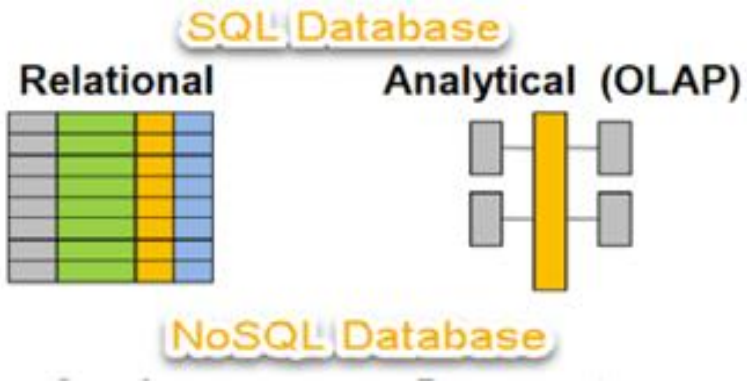
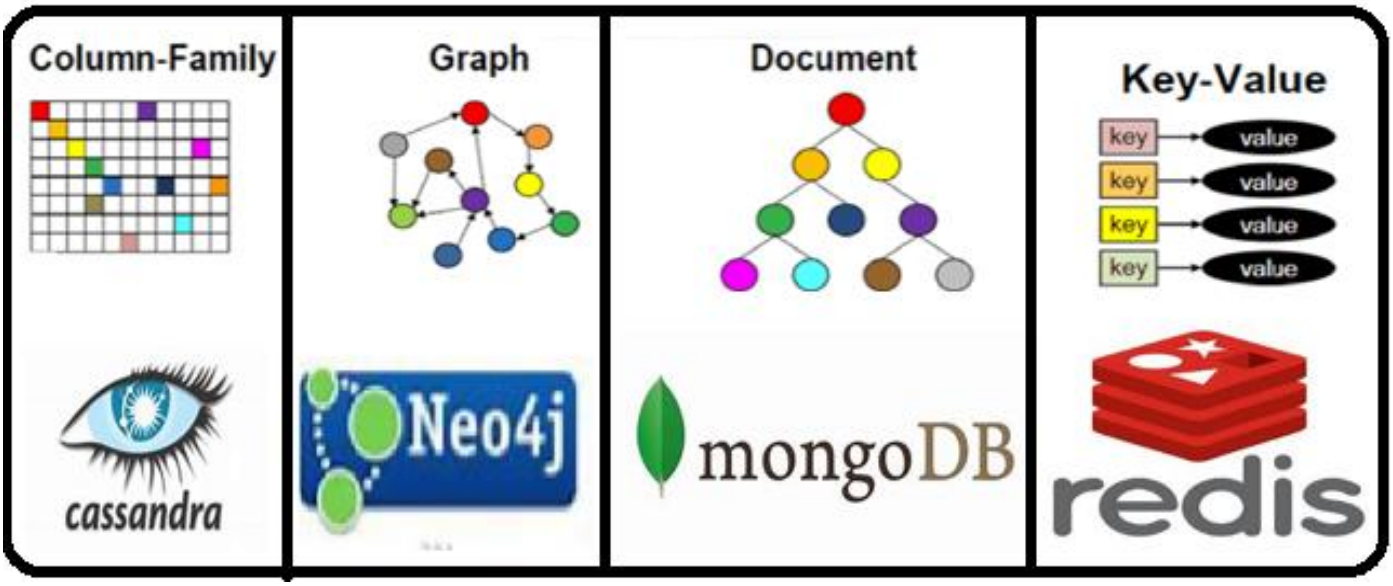

NoSQL
Not Only SQL


SQL
ou
NoSQL


mongoDB
HBASE
Tokyo Cabinet
Redis
riak
CouchDB
Scalaris
Neo4j
the graph database
membase
Cassandra


NoSQL
Gaming
Social
IoT
Web
Mobile
Enterprise
Key/value store
Document database
Column family store
SQL
Web
Mobile
Enterprise
Data mart
Relational table storage
Relationships use joins





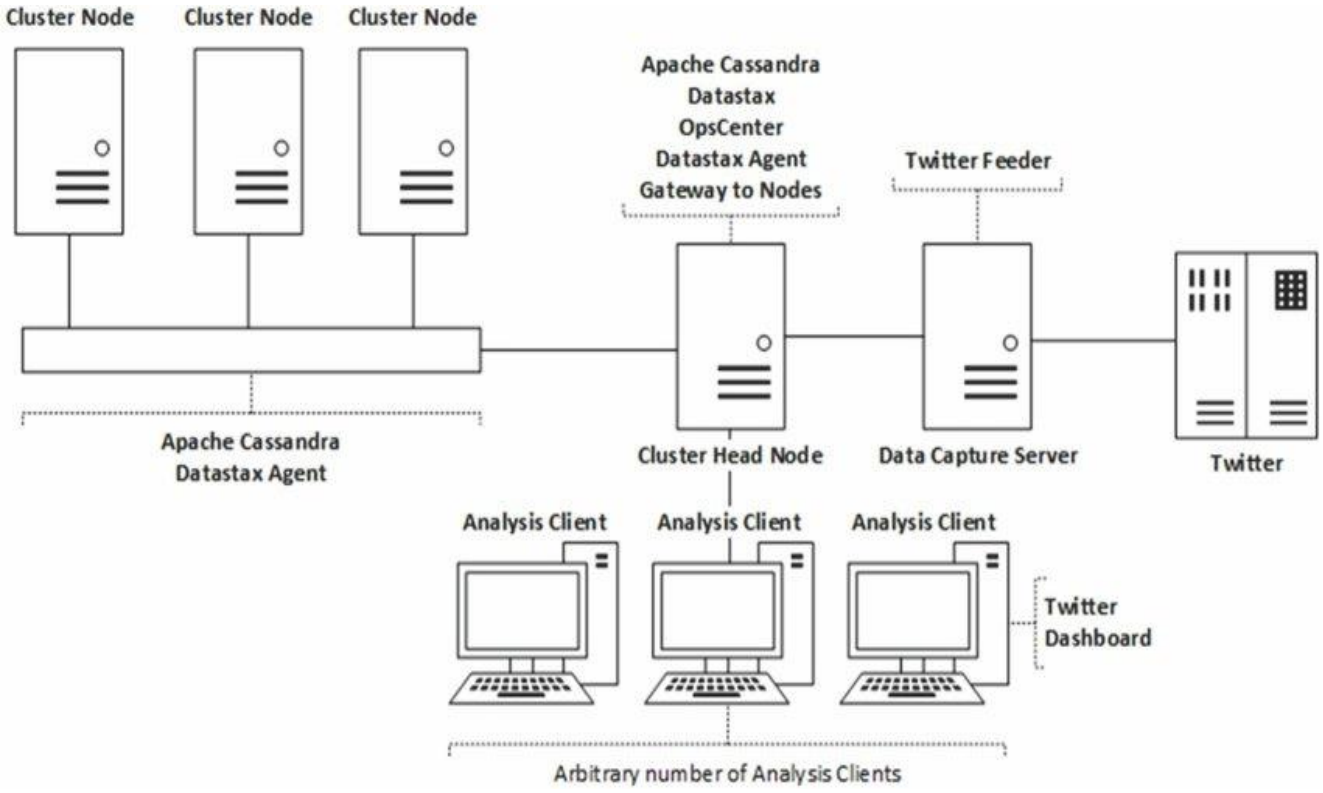
Não possuem squema forte, possuem alta performance e escalabilidade.

Existem mais de 300 tipos de banco de dados NoSql.

Banco relacional tradicional exige uma normalização com uma definição prévia de um esquema que contém os atributos, seus tipos de dados e seus relacionamentos proporcionando integridade dos dados, diminuindo a performance do banco.

O NoSQL utiliza ums forma diferente de acesso aos dados permitindo clusterização em muitos nós, mantendo uma alta performance, latência e escalabilidade.

Banco de dados orientado a coluna criado pelo facebook que utiliza a linguagem Python.



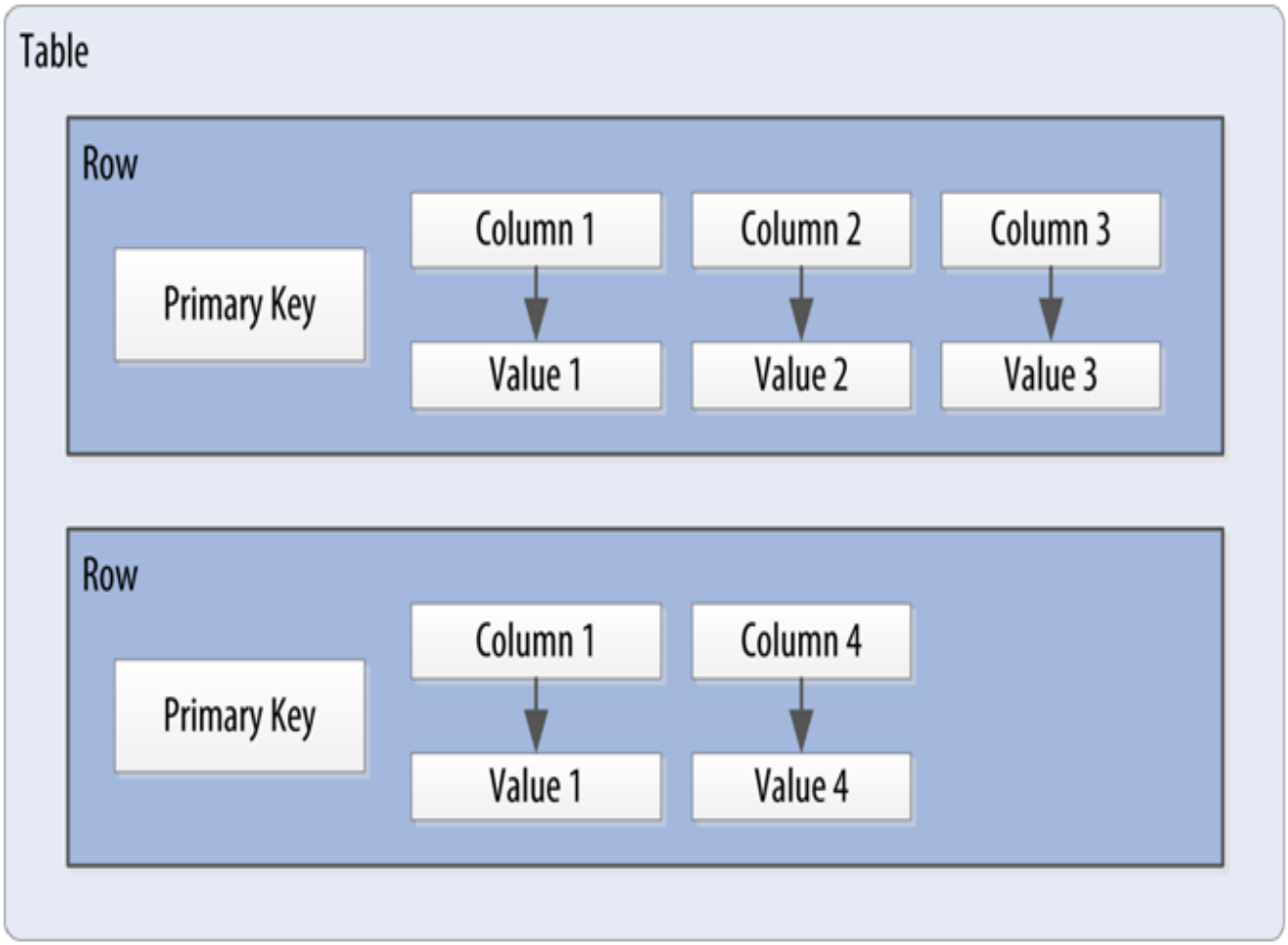


Table
Row
Primary Key
Column 1
Column 2
Column 3
Value 1
Value 2
Value 3
Row
Primary Key
Column 1
Column 4
Value 1
Value 4

Banco de Dados Relacional:

sku_produto	nome_produto	preco_produto
1	Tênis	100
2	Bola	50
3	Camisa	200
4	Bike	900

Banco de Dados Colunar:

sku_produto

id	value
0	1
1	2
2	3
3	4

nome_produto

id	value
0	Tênis
1	Bola
2	Camisa
3	Bike

preco_produto

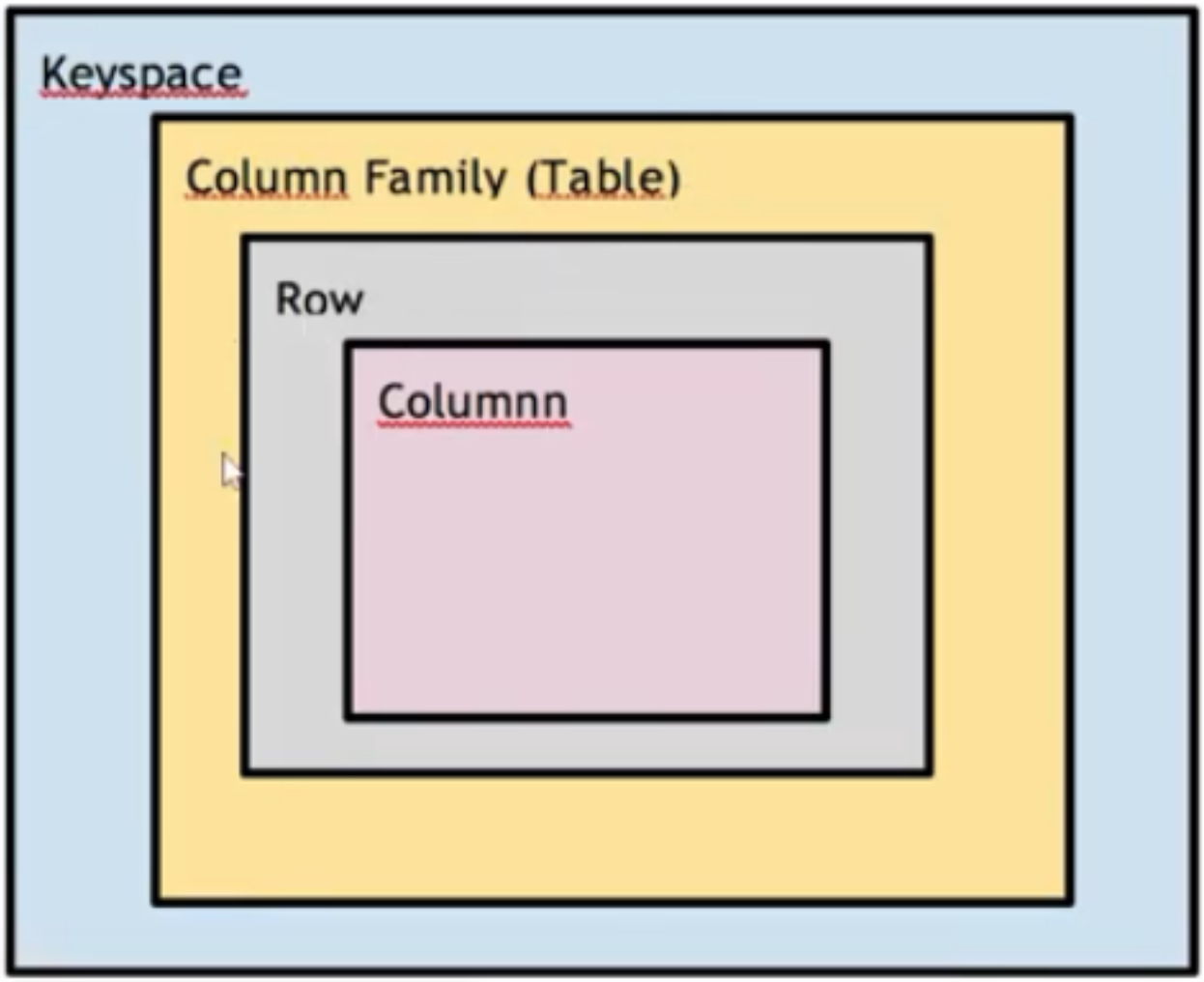
id	value
0	100
1	50
2	200
3	900

Modelo de Dados

- Alusão ao modelo relacional:

Cassandra	Modelo Relacional
Keyspace	Database
Família de colunas	Tabela
Coluna	Coluna
Valor	Valor

Keyspace, corresponde ao banco de dados, onde definimos as tabelas(column family) e definimos também replication factor.



Criando um Keyspace

```
CREATE KEYSPACE blog

WITH

REPLICATION = {

  'class': 'SimpleStrategy',

  'replication_factor' : 1

};
```

Note que temos como replication_factor = 1, ou seja, como temos somente 1 nó no cluster, o fator de replicação é 1

Entrando no Keyspace e criando uma table

Antes de criarmos a primeira tabela, é necessário entrar neste keyspace criado, e para isso basta executar um simples comando:

```
cqlsh> use blog;
```

Após feito, você criará sua primeira tabela dentro deste keyspace:

```
CREATE TABLE posts (
   tag varchar,
   name varchar,
   author varchar,
   description text,
   likes int,
   PRIMARY KEY (tag, name)
 );
```

Inserindo Valores na Table

Agora vamos inserir os primeiros valores dentro da tabela criada:

```
insert into blog.posts (tag, name, author, description, likes)
values
('apache-cassandra','Cassandra post','Jose','post do cassandra',0);
```

Feito isso, para visualizarmos se ocorreu tudo como previsto e de uma forma mais didática, executamos o seguinte comando:

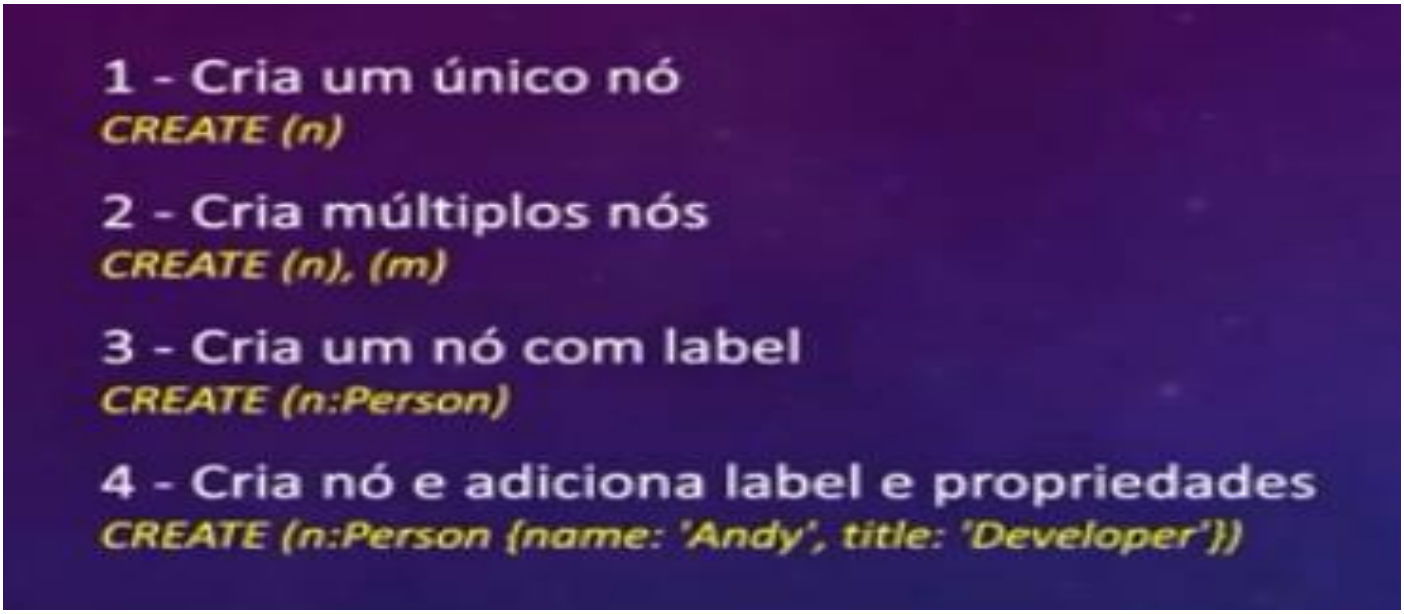
```
cqlsh:blog> select * from posts;
```

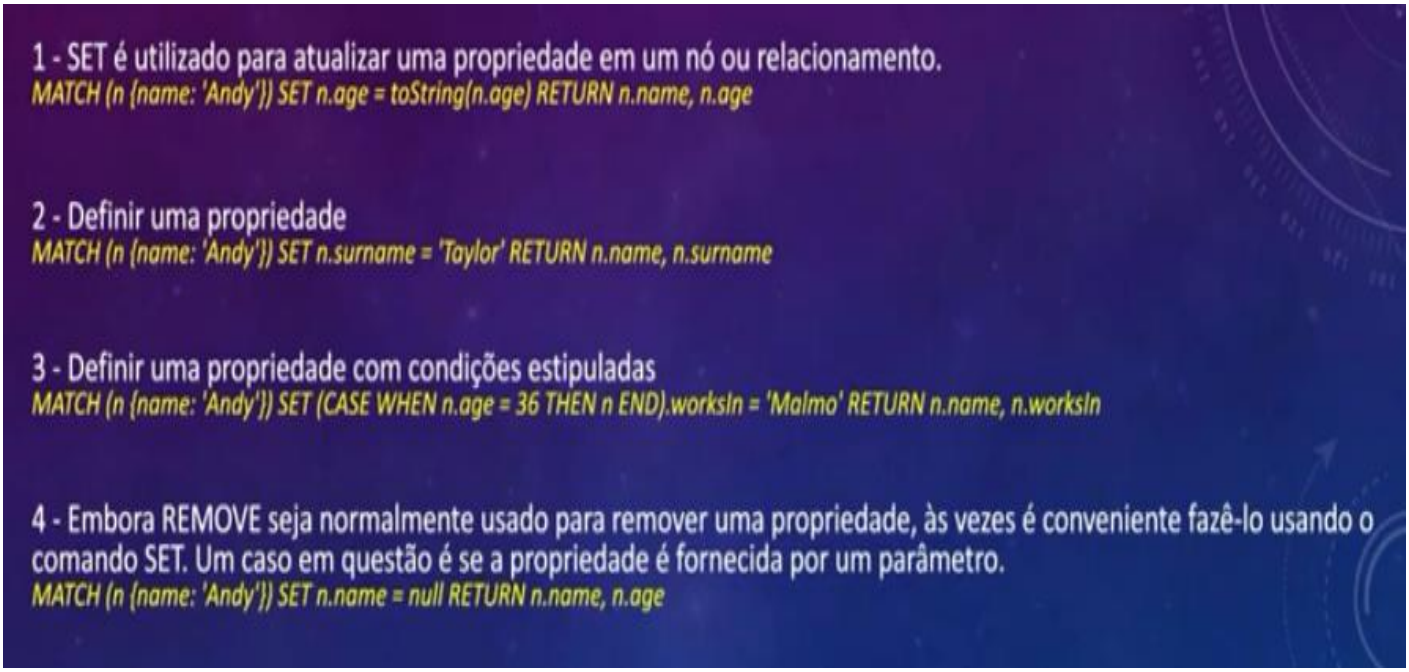
Este comando irá selecionar todos os dados provenientes da tabela "posts", e ele nos retornará:

Caso executado o insert mais de uma vez o cassandra fará um Update no resgistro.

Primeiro e amis popular Banco de dados orientado a grafos.

Neo4j é um banco de dados NoSql orientado a grafos que trabalha com nós, cada nós corresponde a um registro que se relacionam.


1 - Cria um único nó
CREATE (n)
2 - Cria múltiplos nós
CREATE (n), (m)
3 - Cria um nó com label
CREATE (n:Person)
4 - Cria nó e adiciona label e propriedades
CREATE (n:Person {name: 'Andy', title: 'Developer'})


1 - SET é utilizado para atualizar uma propriedade em um nó ou relacionamento.
MATCH (n {name: 'Andy'}) SET n.age = toString(n.age) RETURN n.name, n.age
2 - Definir uma propriedade
MATCH (n {name: 'Andy'}) SET n.surname = 'Taylor' RETURN n.name, n.surname
3 - Definir uma propriedade com condições estipuladas
MATCH (n {name: 'Andy'}) SET (CASE WHEN n.age = 36 THEN n END).worksIn = 'Malmo' RETURN n.name, n.worksIn
4 - Embora REMOVE seja normalmente usado para remover uma propriedade, às vezes é conveniente fazê-lo usando o comando SET. Um caso em questão é se a propriedade é fornecida por um parâmetro.
MATCH (n {name: 'Andy'}) SET n.name = null RETURN n.name, n.age

1 - Para excluir um nó, use a cláusula DELETE

```
MATCH (n:Person {name: 'UNKNOWN'}) DELETE n
```

2- Esta consulta não é para excluir grandes quantidades de dados, mas é útil ao fazer experiências com pequenos conjuntos de dados de exemplo

```
MATCH (n) DETACH DELETE n
```

3 - Quando você deseja excluir um nó e qualquer relacionamento que vai para ou a partir dele, use DETACH DELETE

```
MATCH (n {name: 'Andy'}) DETACH DELETE n
```

4 - Também é possível excluir apenas relacionamentos, deixando o (s) nó (s) não afetado (s)

```
MATCH (n {name: 'Andy'})-[r:KNOWS]->() DELETE r
```

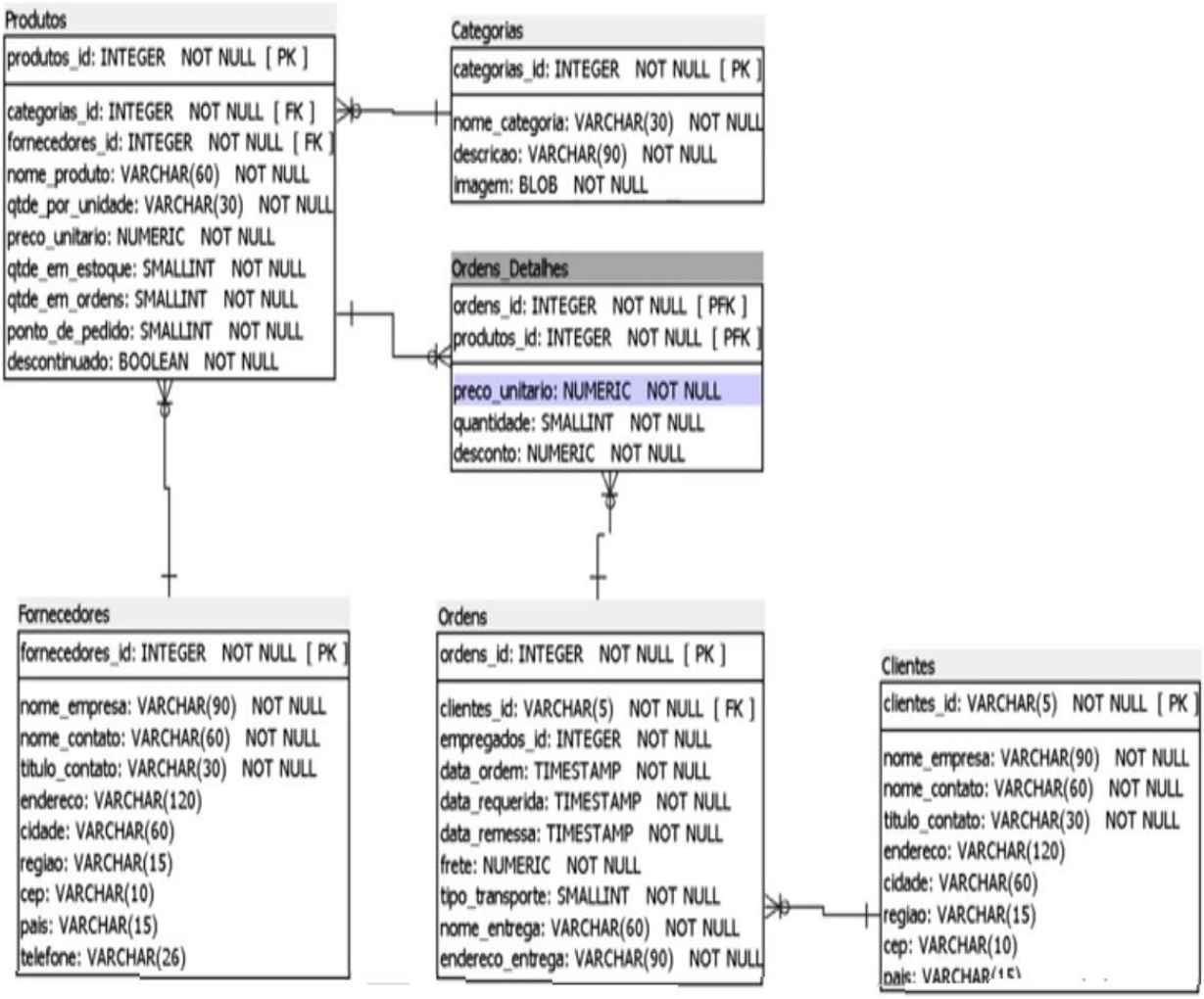
1 - Casos por município:

```
MATCH (m:Municipio) WITH m
MATCH (m) <-[:Municipio]-(p:Pessoa)
return m.municipio AS Municipio, COUNT(p) AS Quantidade ORDER
BY COUNT(p) DESC
```

Intro to Cypher Query

- MATCH (p:pessoa{idade: 25}) RETURN p
  - MATCH é o nosso comando de seleção
  - p:pessoa é o nosso conjunto de dados
  - {idade:25} é o nosso where
  - return p é quais registros eu quero que retorne

Recentemente foi inserido o comando Where.



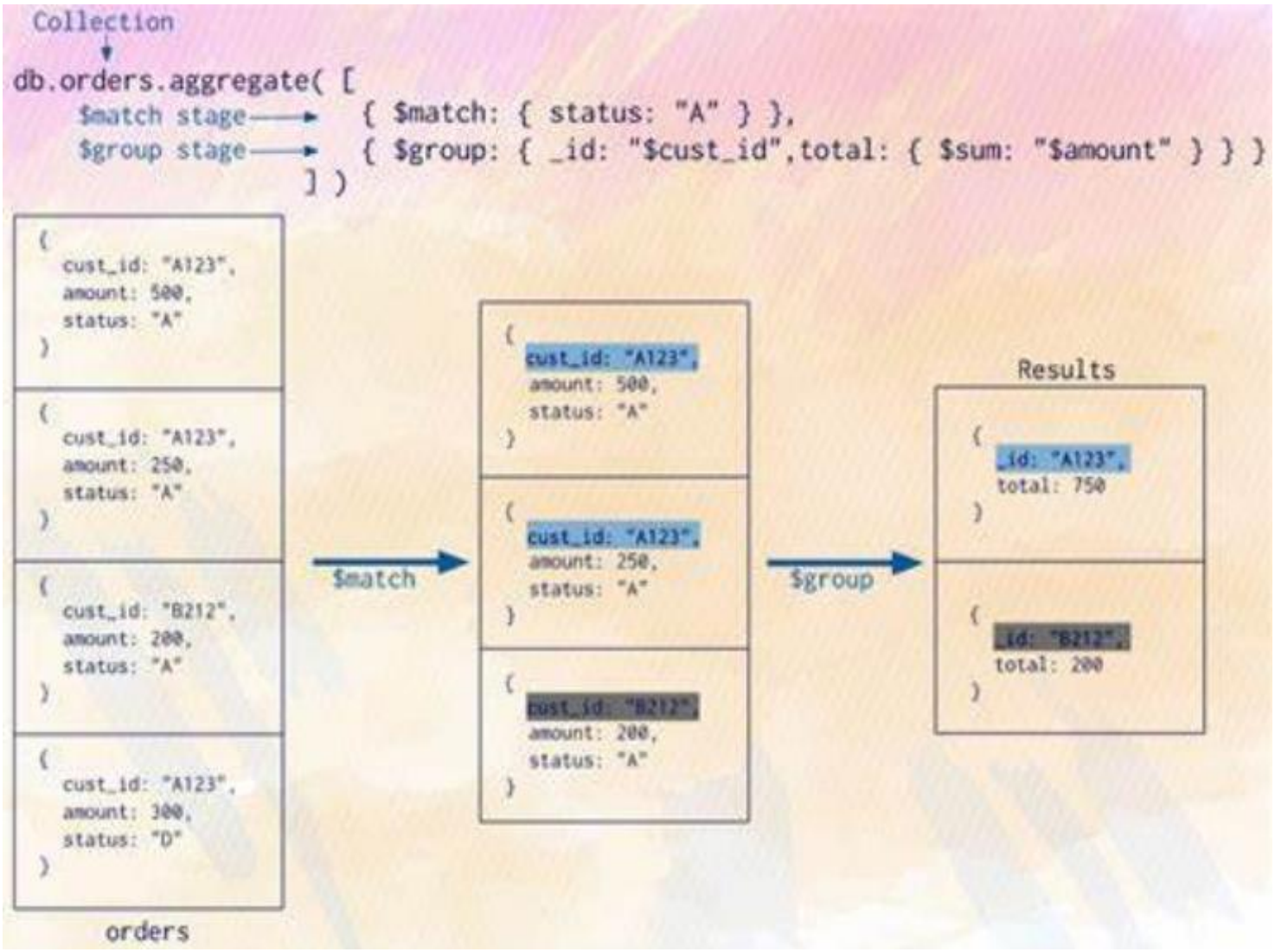
Banco de dados orientado a documentos desenvolvido em C++ e suportado pela maioria das linguagens atuais.

Banco Relacional	Mongo DB
Banco de Dados	Banco de Dados
Tabela, Visão	Coleção
Linha	Documento (JSON, BSON)
Coluna (Esquema Rígido)	Campo(Esquema Flexível)
Índice	Índice
Junção	Documento Embutido
Chave Estrangeira	Referência
Partição	Sharding

Importante utilizar os recursos da linguagem criar esquemas.

Ex. Não criar campos data como string, utilizar indices de pesquisa.

BSON - Binary JSON.



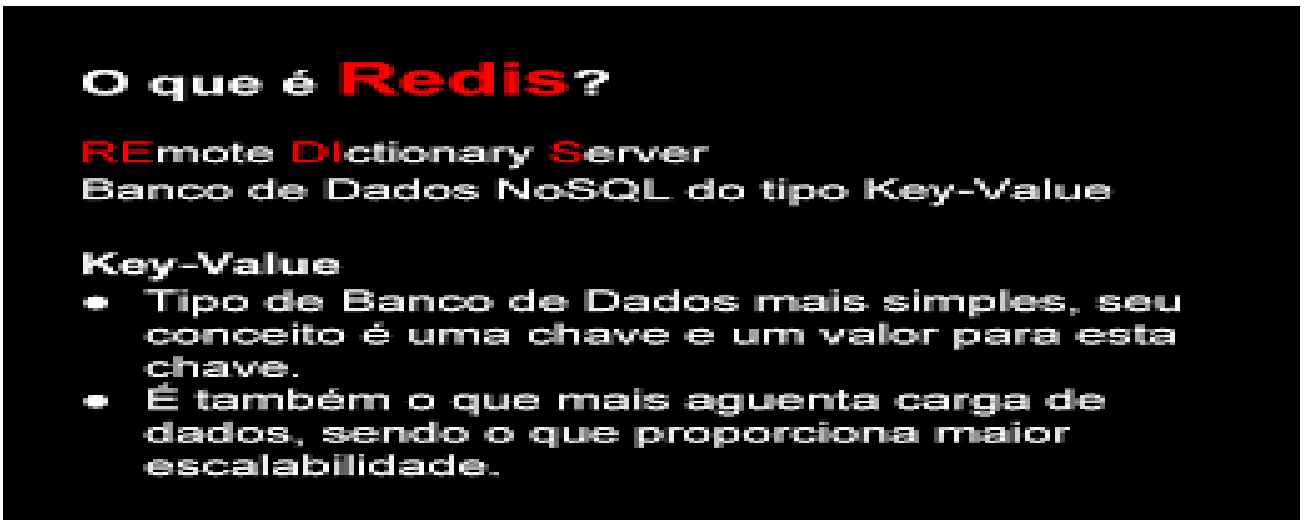
SQL	MongoDB
INSERT INTO USERS VALUES(1,1)	db.users.insert({a:1, b:1})
SELECT a,b FROM users	db.users.find({}, {a: 1, b: 1})
SELECT * FROM users	db.users.find()
SELECT * FROM users WHERE age=33	db.users.find({age: 33})
SELECT * FROm users WHERE name = "pedro"	db.users.find({name:"pedro"})

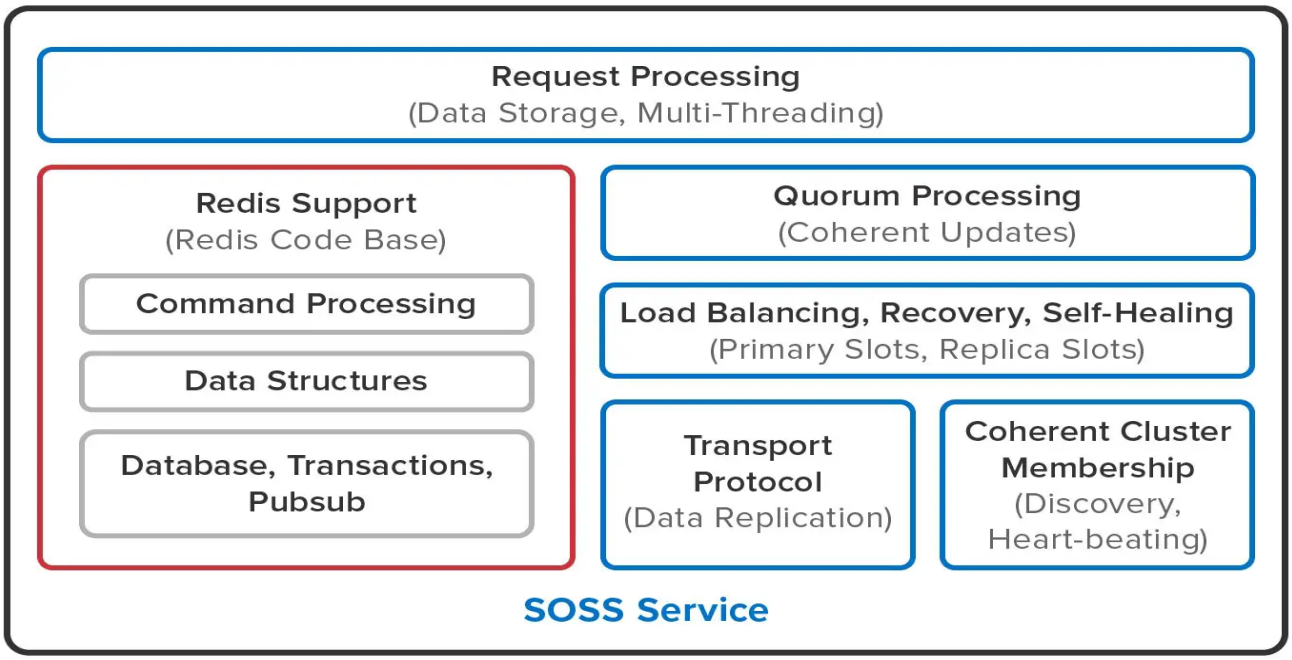
Operador	Função
$sum	Utilizado para a soma de valores.
$avg	Utilizado para encontrar a média entre os valores.
$gte	Utilizado como critério de "maior ou igual".
$lte	Utilizado como critério de "menor ou igual".
$min	Utilizado para recuperar o "menor valor".
$max	Utilizado para recuperar o "maior valor".
$in	Utilizado para localizar dentro de um array qualquer um dos parâmetros.
$all	Utilizado para localizar dentro de um array todos os elementos do parâmetro.
$regex	Utilizado para o uso de expressões regulares.
$and	Idêntico ao operador "and" do SQL.
$or	Idêntico ao operador "or" do SQL.

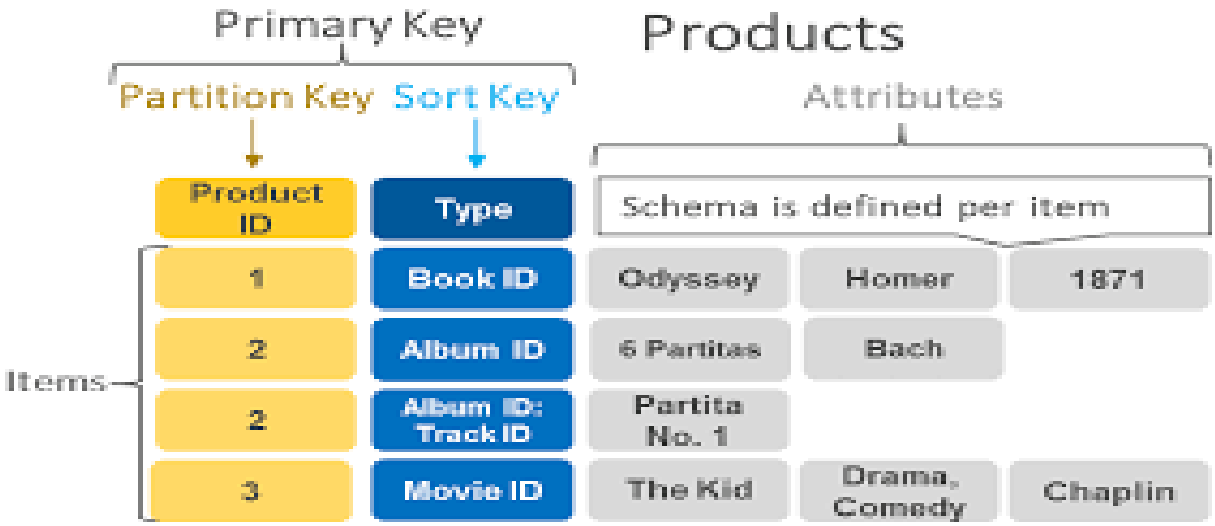
amazon
DynamoDB

Ele é um banco de dados totalmente gerenciavel que suporta tanto modelos de documentos e valores-chaves (key-value).

Banco de dados orientado por chave e valor.





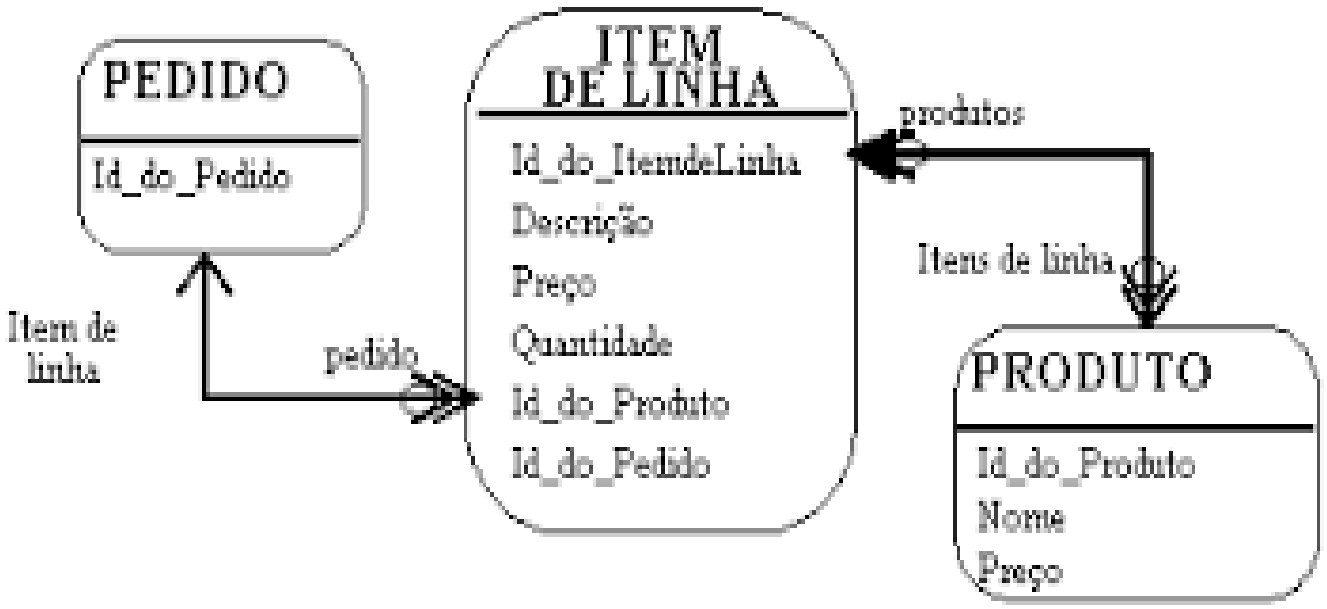


Pessoa

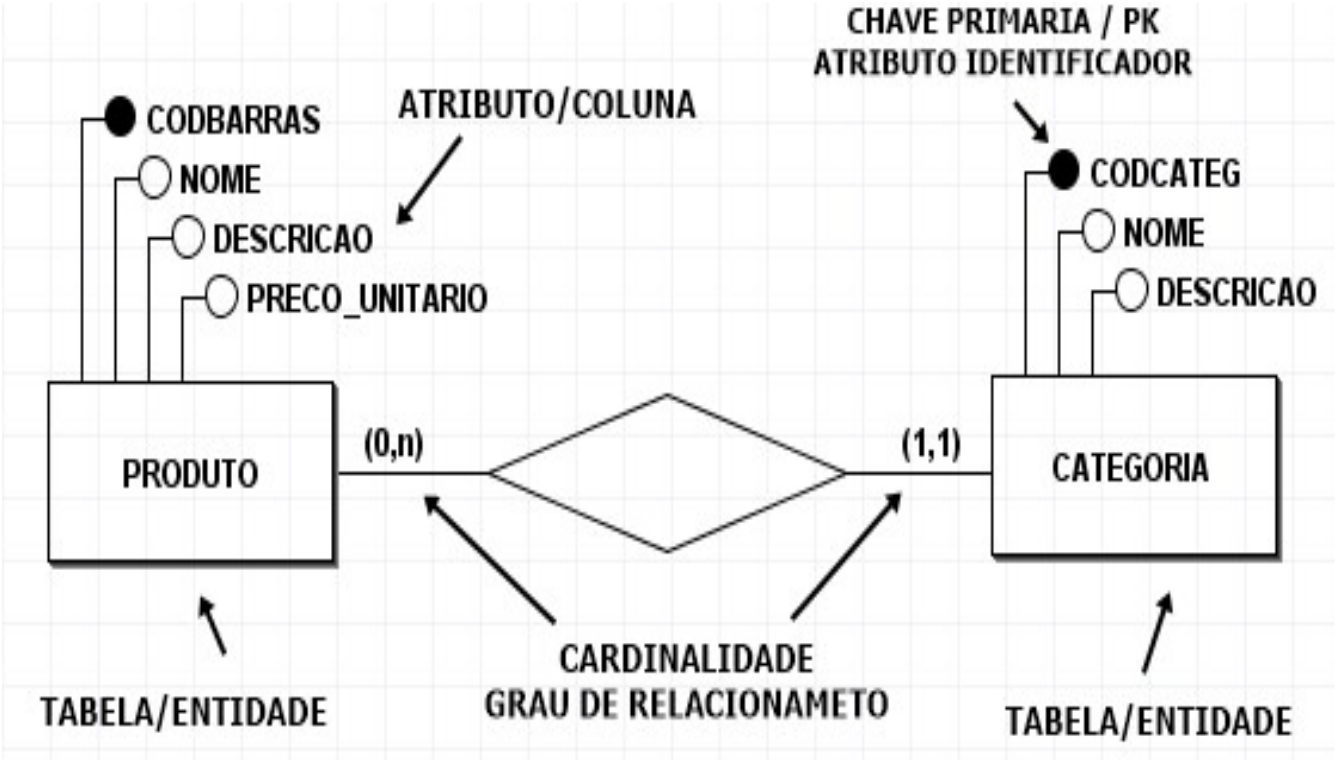
```
{
"id" : 987,
"nome" :"Ed".
"rua" :"Jordao",
"numero": 91  }
```

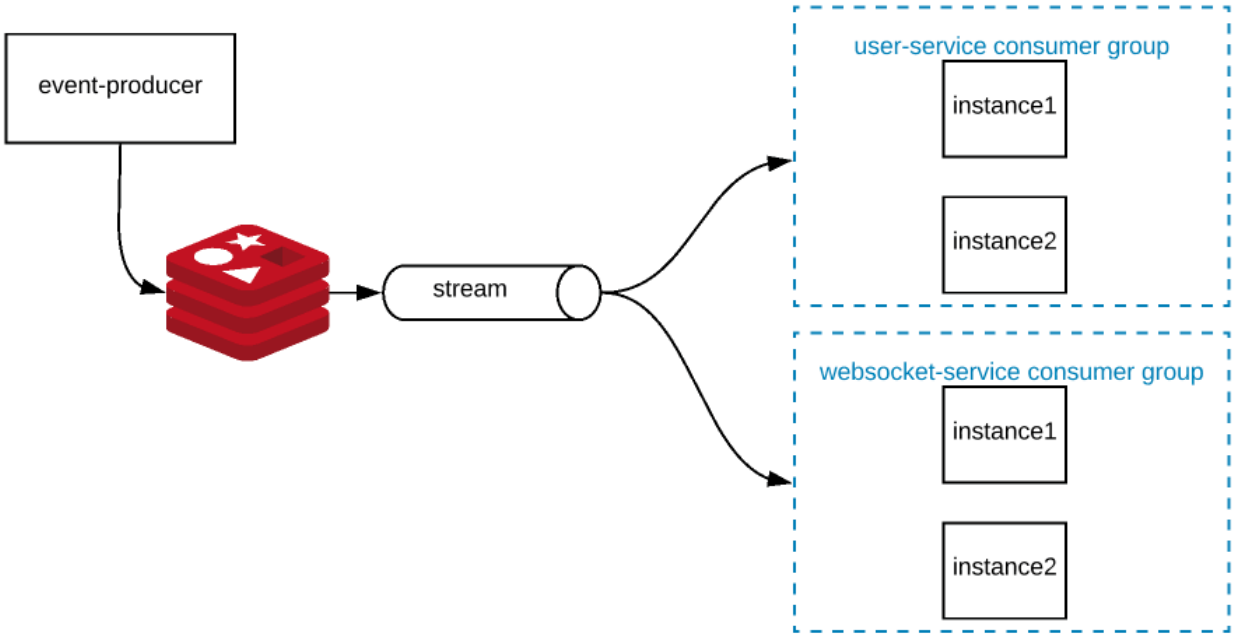
Pessoa

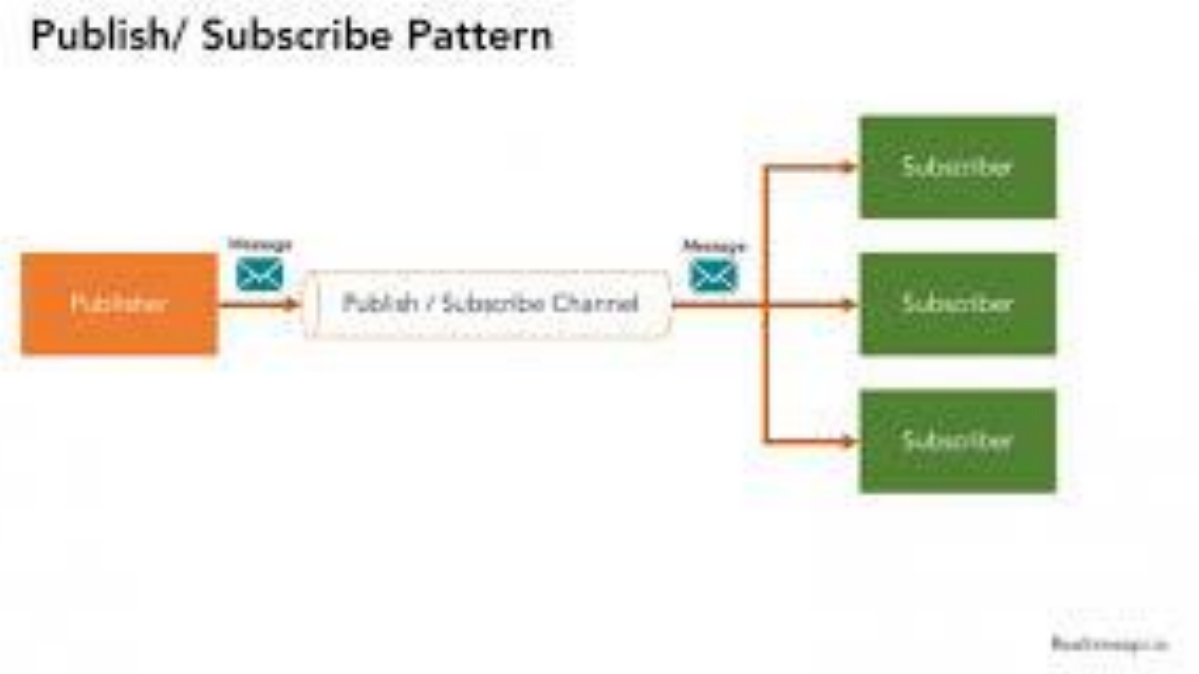
```
{
 "id" : 988,
 "nome": "Olivia",
  "sexo": "feminino" }
```



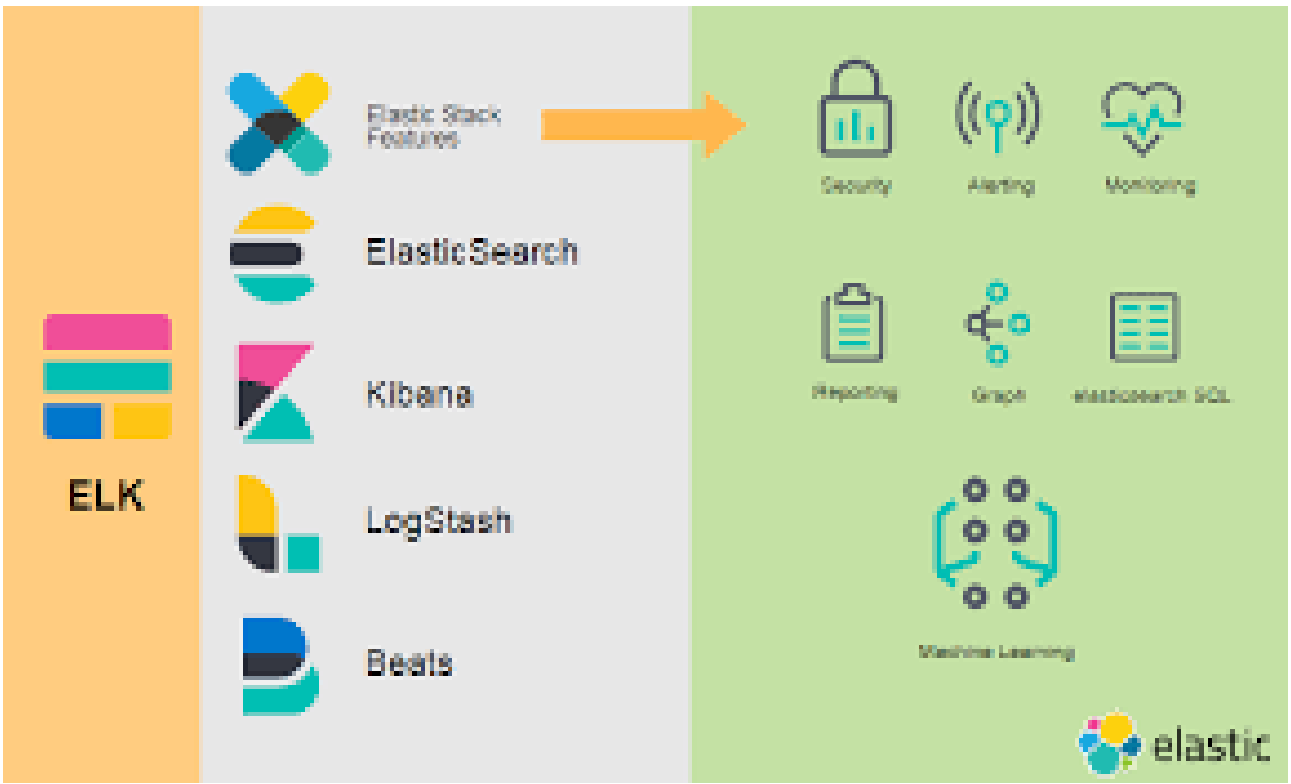

```
{
  "ClienteId" : 12,
  "DataPedido": "01/04/2018",
  "PedidoID": 123,
  "ItensPedido"
  [
    {
      "ItensPedidoID": 234,
      "Produto"
      {
        "ProdutoID": 33,
        "DescricaoProduto": "teste",
        "PrecoProduto": 2.00
      },
      "Quantidade": 50,
      "ValorTotal": 100.00
    }
  ]
}
```


CHAVE PRIMARIA / PK
ATRIBUTO IDENTIFICADOR
CODBARRAS
ATRIBUTO/COLUNA
NOME
DESCRICAO
PRECO_UNITARIO
CODCATEG
NOME
DESCRICAO
PRODUTO
(0,n)
(1,1)
CATEGORIA
TABELA/ENTIDADE
CARDINALIDADE
GRAU DE RELACIONAMETO
TABELA/ENTIDADE


event-producer
stream
user-service consumer group
instance1
instance2
websocket-service consumer group
instance1
instance2


Publish/ Subscribe Pattern
Publisher
Message
Publish / Subscribe Channel
Message
Subscriber
Subscriber
Subscriber

elasticsearch



config/elasticsearch.yml

elasticsearch.yml

```
# ======================== Elasticsearch Configuration =========================
#
# NOTE: Elasticsearch comes with reasonable defaults for most settings.
#       Before you set out to tweak and tune the configuration, make sure you
#       understand what are you trying to accomplish and the consequences.
#
# The primary way of configuring a node is via this file. This template lists
# the most important settings you may want to configure for a production cluster.
#
# Please consult the documentation for further information on configuration options:
# https://www.elastic.co/guide/en/elasticsearch/reference/index.html
#
# ---------------------------------- Cluster -----------------------------------
#
# Use a descriptive name for your cluster:
#
#cluster.name: my-application
#
# ------------------------------------ Node ------------------------------------
#
# Use a descriptive name for the node:
#
#node.name: node-1
#
```

config/jvm.options

jvm.options

```
## JVM configuration

################################################################
## IMPORTANT: JVM heap size
################################################################
##
## You should always set the min and max JVM heap
## size to the same value. For example, to set
## the heap to 4 GB, set:
##
## -Xms4g
## -Xmx4g
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/current/heap-size.html
## for more information
##
################################################################

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms1g
-Xmx1g
```