Data Persistence

By
Raúl Mazo
Paola Vallejo

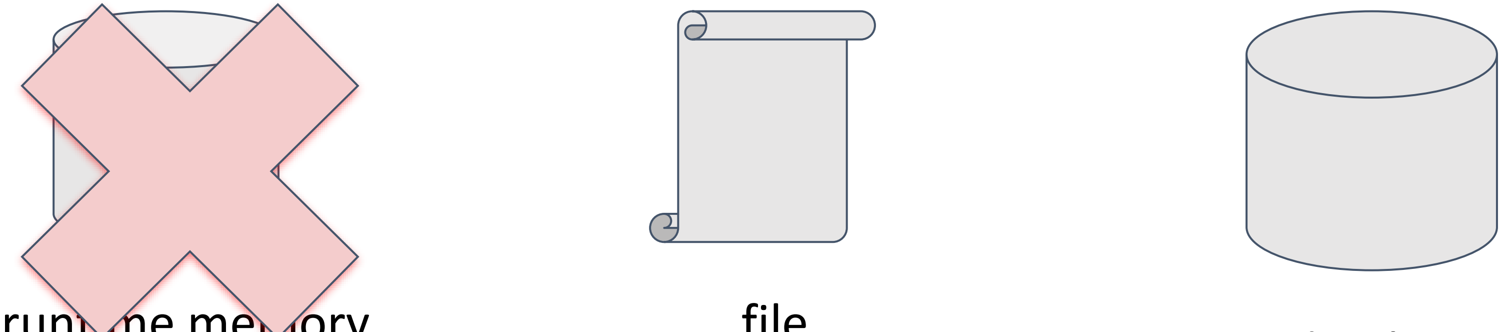
Data persistence

- Persistence is the ability to **save the information** of a program for **reuse** at a **later** time. Persistence is the p**ermanence** of information
- User:
  - Save, Save as, Recover …
- Programmer:
  - mechanism responsible for the **backup** and **recovery** of data
  - a program can be terminated **without losing its data** and execution status
  - for a data store to be considered persistent, it must write to **non-volatile storage**
  - **serialization** of the data and the reverse process of retrieving the data from the serialized information

Why data persistence?

- For later use
- To maintain the status
- For logging purposes
- To further process and derive knowledge
- Data can be stored, read, updated/modified, and deleted

Ways of data persistent



runtime memory
cache memory

file

database

Data storage formats

Plain-text, XML, JSON, tables, text files, images

Ways of data persistent

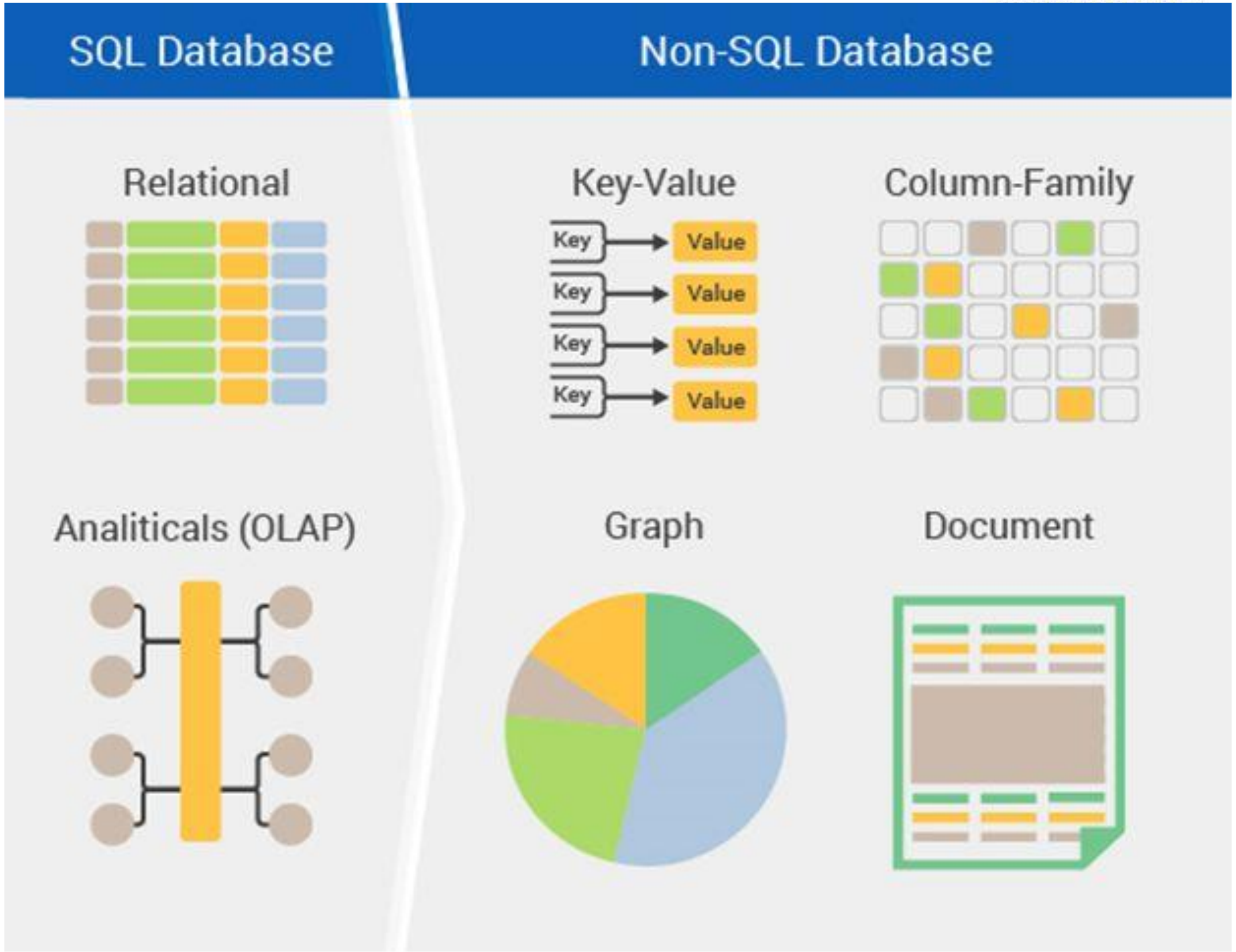
- **Pure in-memory**, such as cache memory
    - High speed and availability
    - Few data
    - No persistence at all
- **In-memory with periodic snapshots**, such as Redis
    - Periodic snapshots to disk at a configurable interval
- **Disk-based** with update-in-place writes, such as MySQL or MongoDB
- **Commitlog-based**, such as all traditional OLTP databases (Oracle, SQL Server ...)

Database type

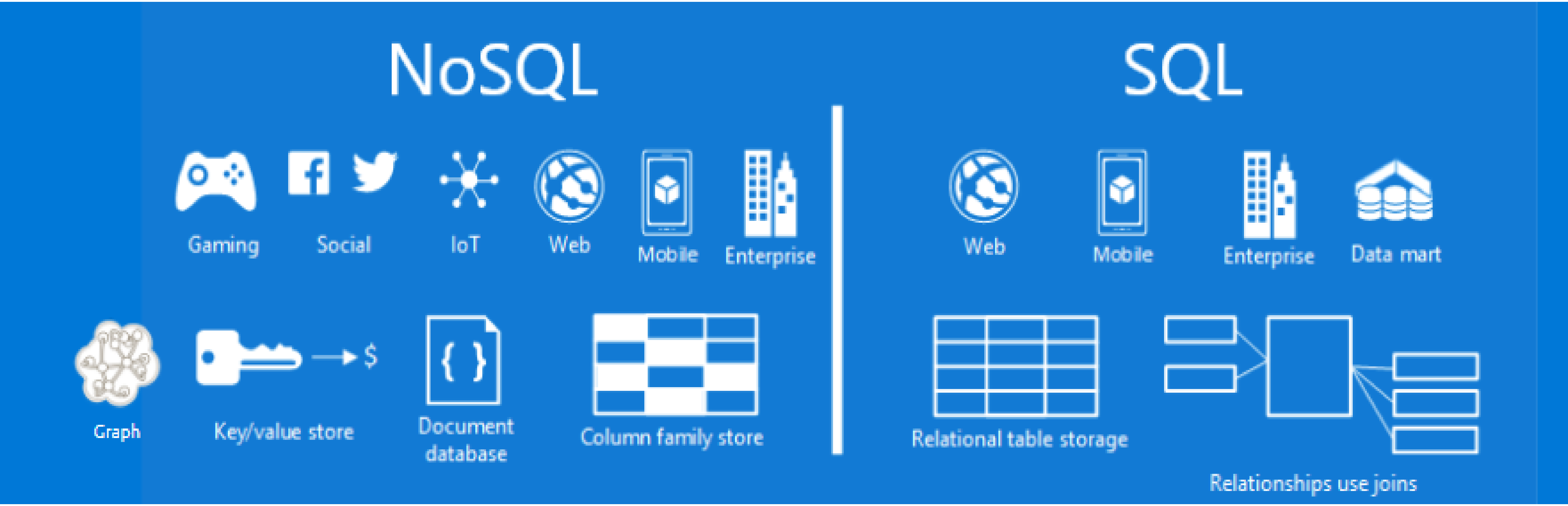
- Object-oriented
- Network
- Hierarchical

SQL

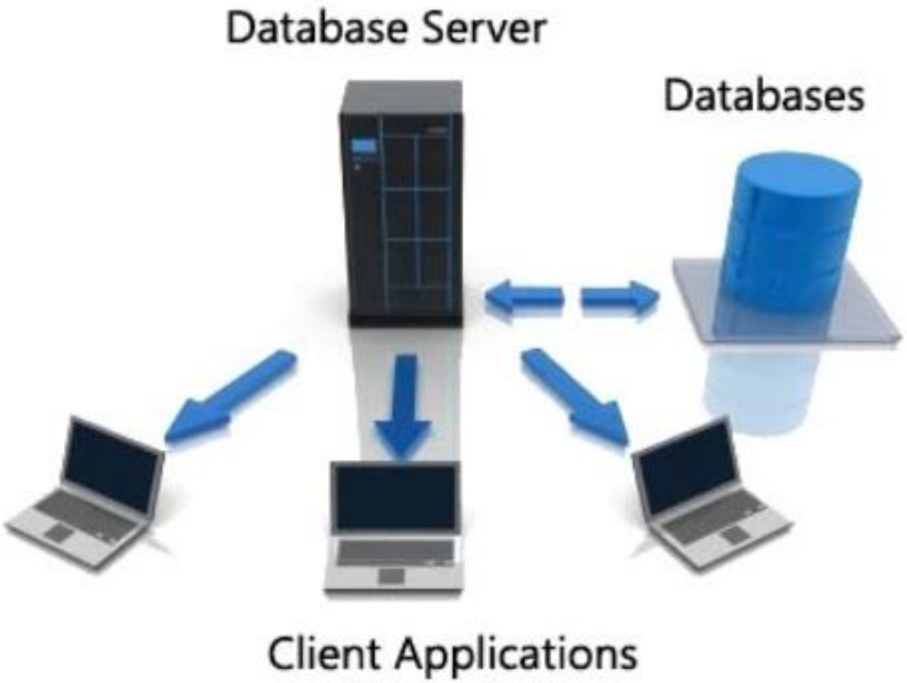
Non-SQL



Contexts


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

Database Server



- **Back-end system** of a database application using client/server architecture.

- Performs tasks such as data analysis, storage, data manipulation, and other non-user specific tasks.

Database Management System

- Software for creating and managing databases.

- Interface between the database and end users or application programs. Ex. MySQL, PostgreSQL, Microsoft Access, SQL Server, Oracle.

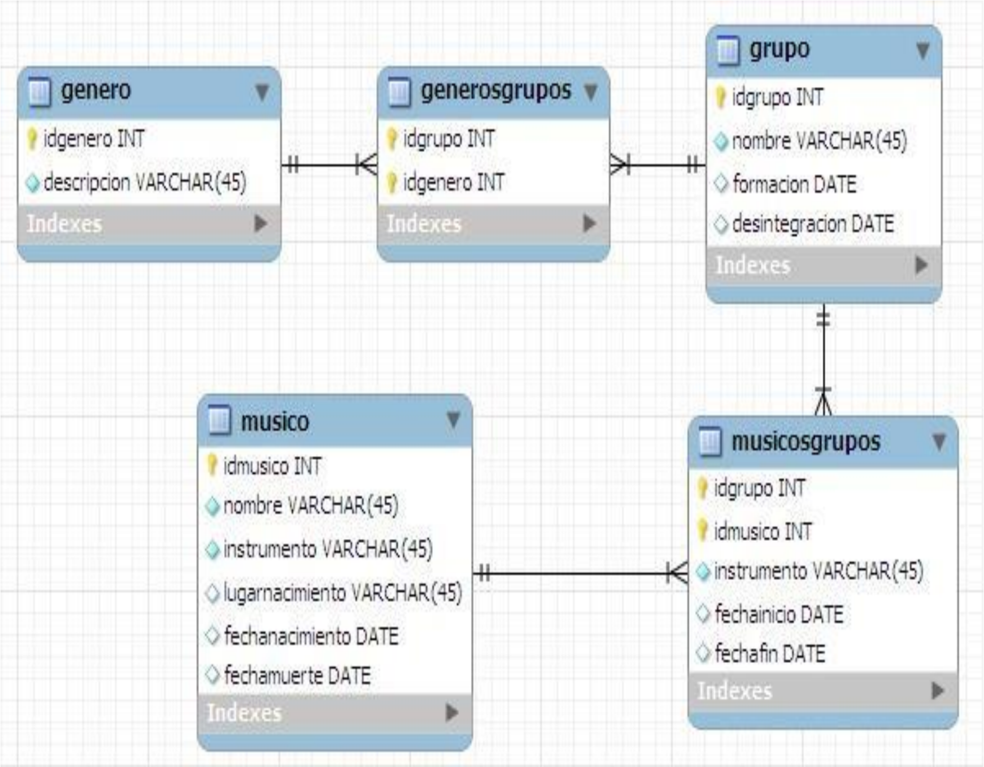
Relational models

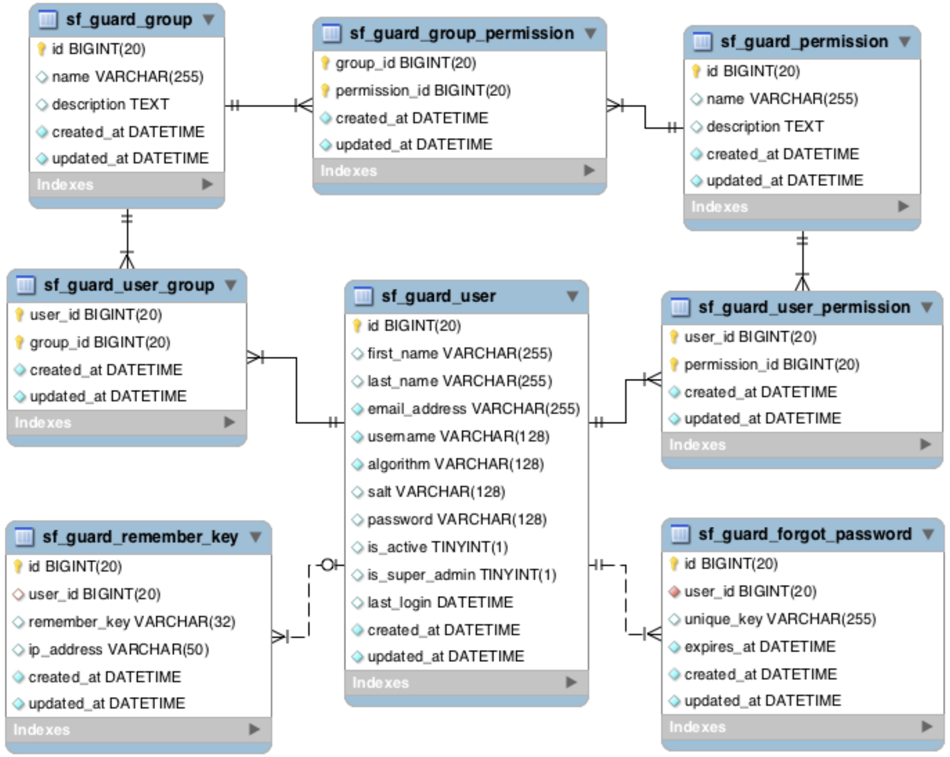
General concept

The relational data model organizes and represents data in the form of tables and relationships:

Logical representation	Physical representation	Relational model
Table	Sequential File	Relation
Row	Register	Tuple
Column	Field	Attribute

Example - Relational model

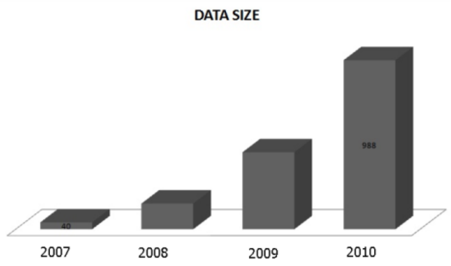




Non-SQL models

SQL database issues

Major problems that make it ineffective



Huge growth of information on the Internet. The growth of data from 2007 to 2010 is almost 25 times.

Connectivity. Over time, information is becoming more and more connected.

Semi-structure. Information that has some mandatory attributes, but many optional attributes. As the information grew, there was a need to increase the columns of the table, which would lead to sparse tables.

Response to relational inefficiency

□ NoSQL databases that were initially created in response to the needs for better scalability, lower latency and greater flexibility in the era of bigdata and cloud computing.

These non-functional aspects are the main reason for the use of NoSQL databases.

NoSQL databases are structures that allow storing information in situations where relational databases generate scalability and performance problems when thousands of concurrent users and millions of daily queries occur.

NoSQL advantages

Run on machines with few resources:	•Require little computation, so they can be mounted on lower cost machines.
Horizontal scalability:	•To improve performance, more nodes are added, with the only operation being to indicate to the system which nodes are available.
Handle large amounts of data:	•Distributed structure, in many cases by means of Hash tables.
Does not generate bottlenecks:	•SQL systems need to transcribe each statement in order to be executed, which is a common entry point, which can slow down the system when faced with many requests.

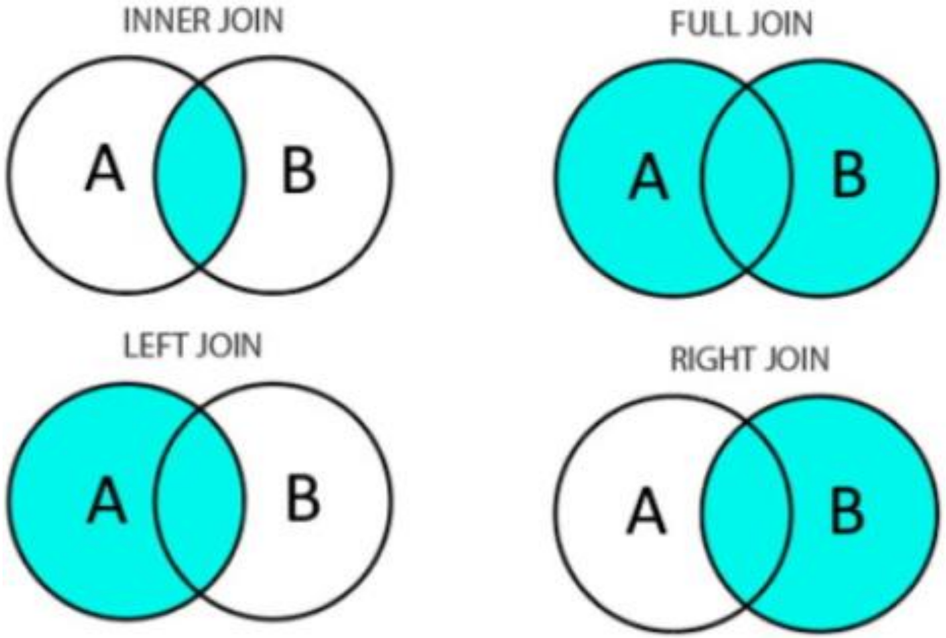
Non-SQL

Do not use SQL as a query language

- Most NoSQL databases avoid using this type of language or use it as a support language. To give some examples, Cassandra uses the CQL language, MongoDB uses JSON or BigTable uses GQL.

Do not use fixed structures such as tables for data storage.

- They allow the use of other types of information storage models such as key-value systems, objects or graphs.



JOIN operations are not usually allowed

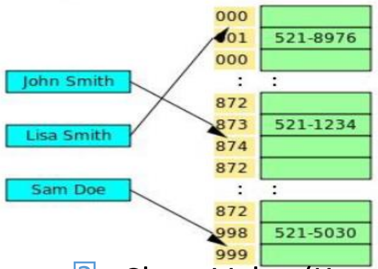
- When having such an extremely large volume of data it is often desirable to avoid JOINs.
- This is because, when the operation is not a key lookup, the overhead can become very costly.
- The most straightforward solutions are to denormalize the data, or to perform the JOIN in software, at the application layer.

Distributed architecture

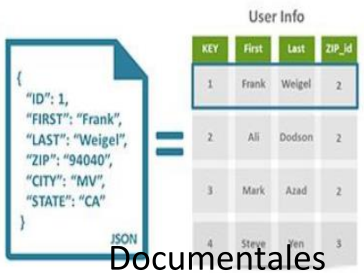
- Relational databases are usually centralized in a single machine or in a master-slave structure, however in NoSQL cases the information may be shared in several machines by means of distributed Hash table mechanisms.

NoSQL Overview

NoSQL database types



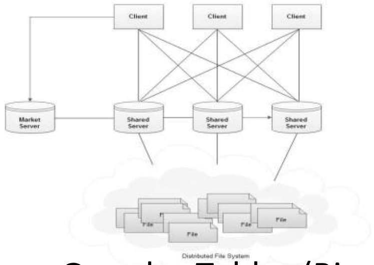
Clave Valor (Key Value Stores)


```
{
  "ID": 1,
  "FIRST": "Frank",
  "LAST": "Weigel",
  "ZIP": "94040",
  "CITY": "MV",
  "STATE": "CA"
}
```

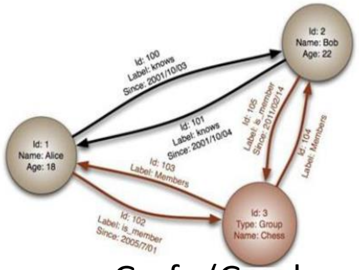
User Info

KEY	First	Last	ZIP_id
1	Frank	Weigel	2
2	Ali	Dodson	2
3	Mark	Azad	2
4	Steve	Yen	3

Documentales (Document Database)



Grandes Tablas (Big Table)



Grafo (Graph Databases)

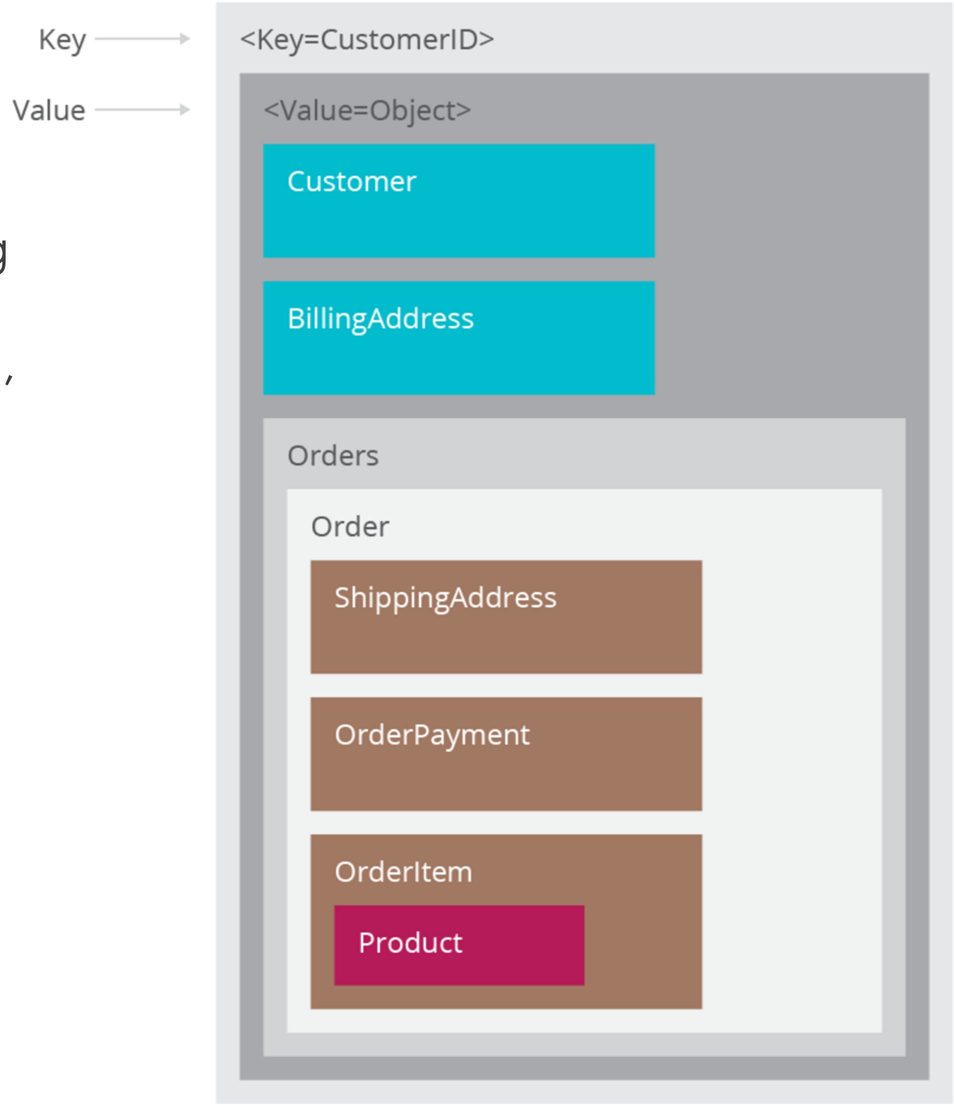
NoSQL database examples

Key Value	• Berkeley DB, Tokyo Tyrant, Voldemart, Crassandra.
Big Table	• Google BigTable, HBAse
Document	• Mongo DB, Couch DB.
Graph	• Neo4j, InfoGrid o Virtuoso

Key - Value

- They store tuples containing a key and its value. When you want to retrieve a data, you simply search for its key and retrieve the value.
  - Hash tables

- DynamoDB
- Redis



Key

Value

<Key=CustomerID>

<Value=Object>

Customer

BillingAddress

Orders

Order

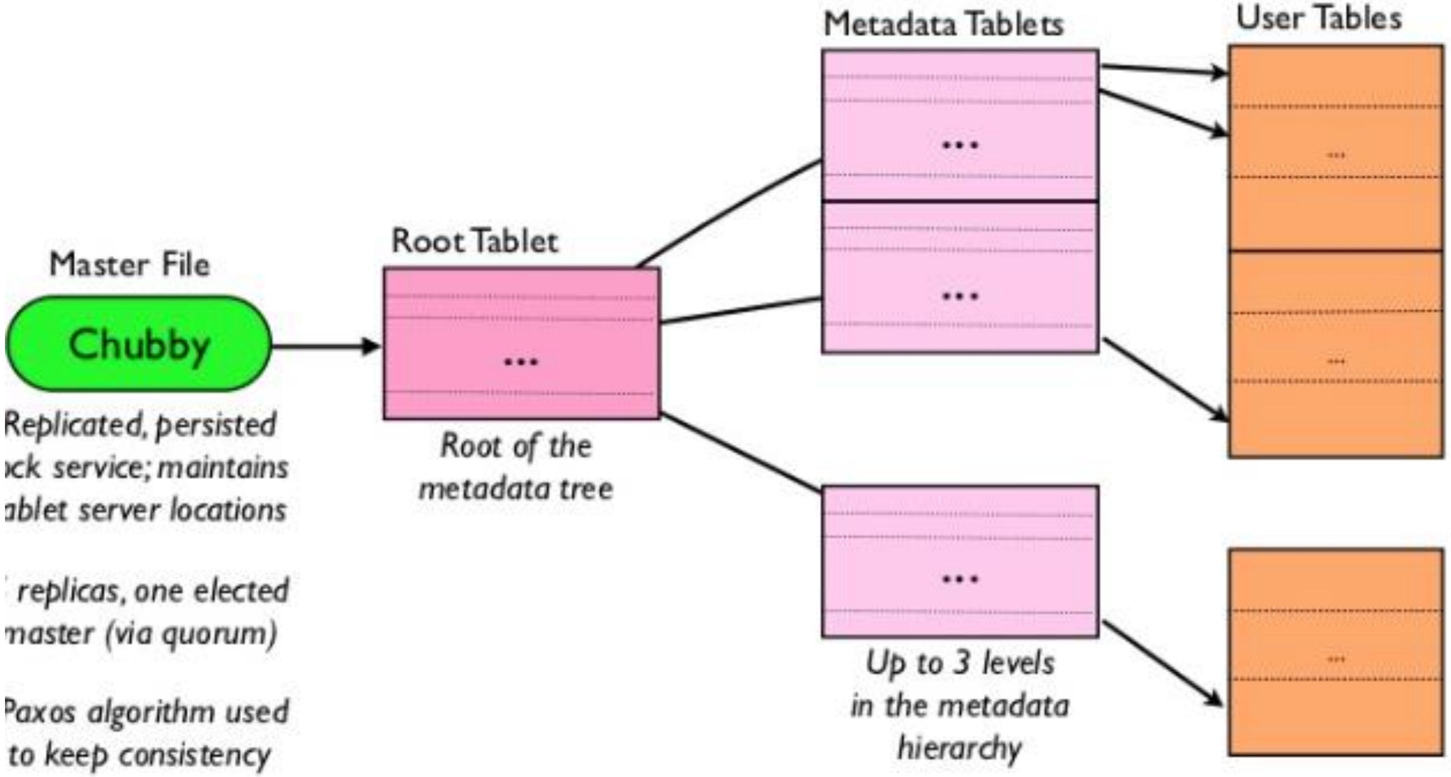
ShippingAddress

OrderPayment

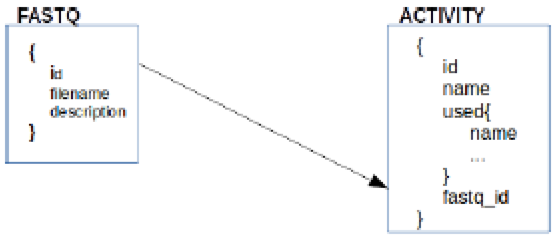
OrderItem

Product

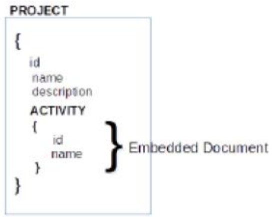
Bigtable


Master File
Chubby
Replicated, persisted
ɔck service; maintains
ablet server locations
' replicas, one elected
naster (via quorum)
Paxos algorithm used
to keep consistency
Root Tablet
...
Root of the
metadata tree
Metadata Tablets
...
...
...
Up to 3 levels
in the metadata
hierarchy
User Tables
...
...
...

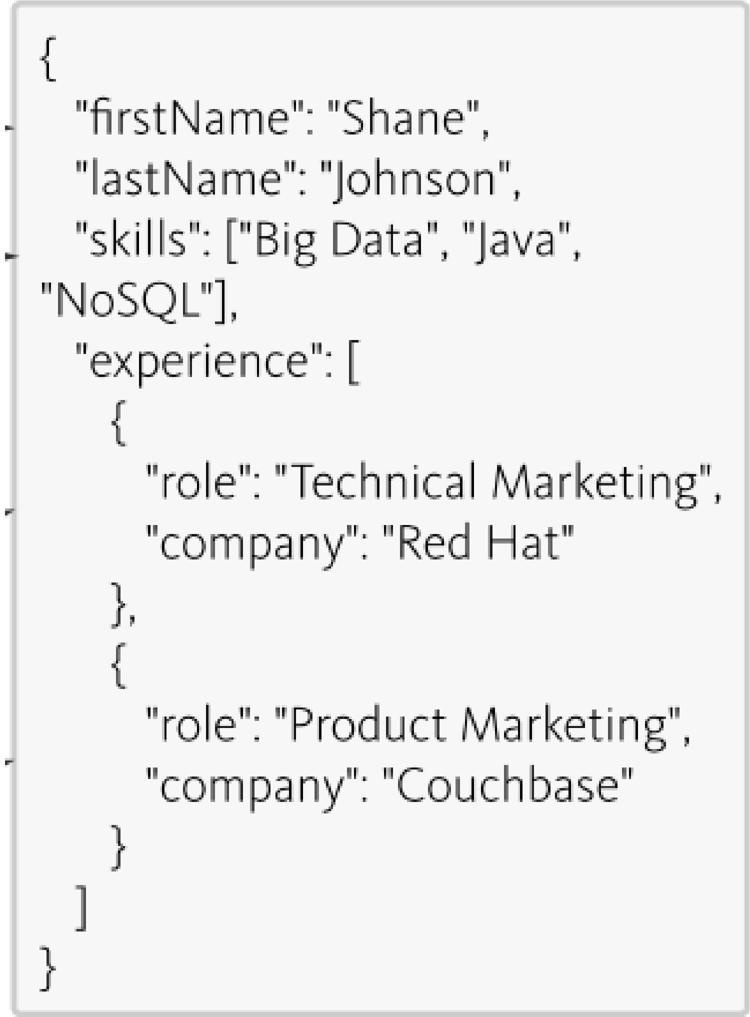
Documents



reference relations



embedded documents



```
{
   "firstName": "Shane",
   "lastName": "Johnson",
   "skills": ["Big Data", "Java",
"NoSQL"],
   "experience": [
      {
        "role": "Technical Marketing",
        "company": "Red Hat"
      },
      {
        "role": "Product Marketing",
        "company": "Couchbase"
      }
   ]
}
```

- Semi-structured data, i.e. documents. These data are stored in some standard format such as XML, JSON or BSON.

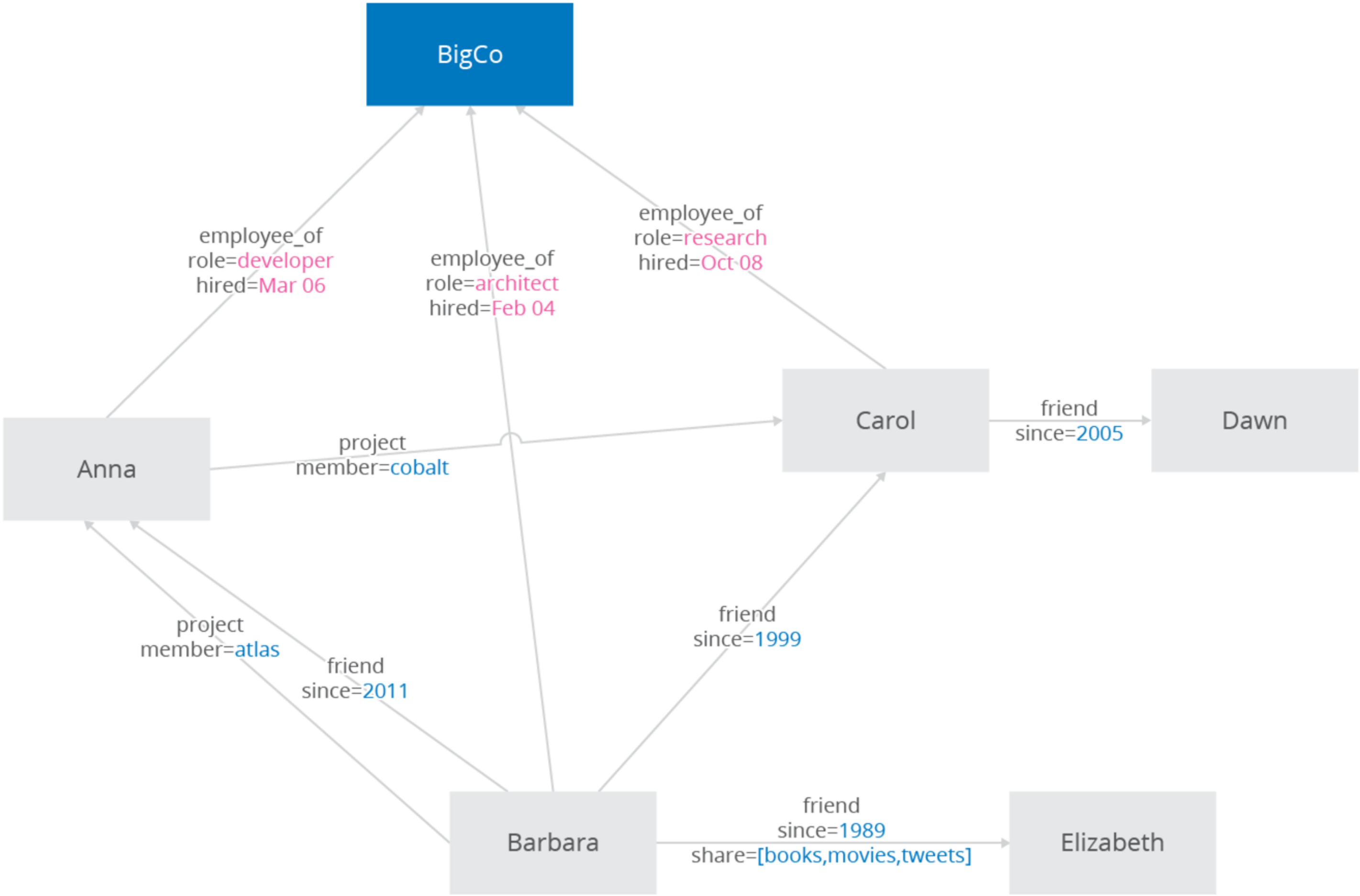
- MongoDB
- CouchDB

Graph

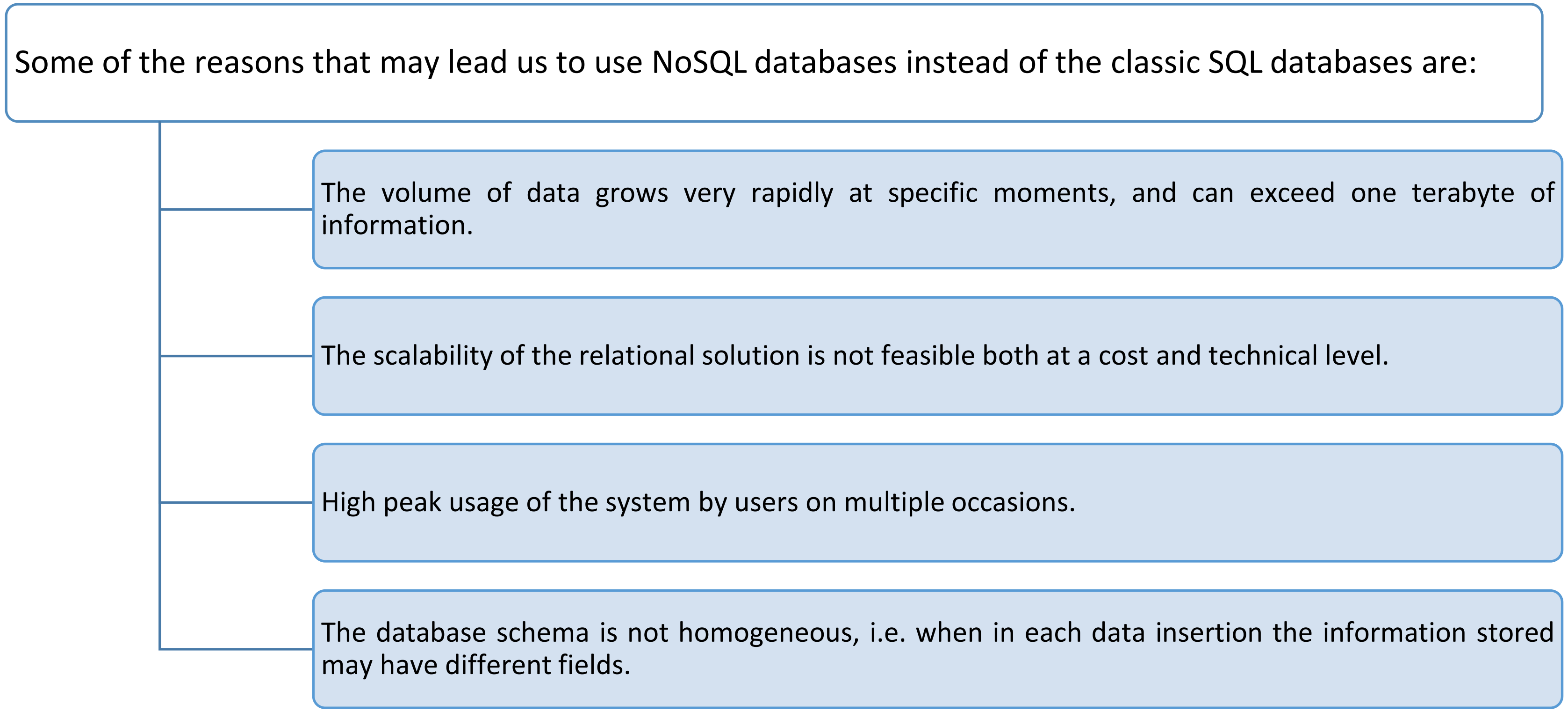
- Based on graph theory, they use nodes and edges to represent the stored data. They are very useful for storing information in models with many relationships, such as networks and social connections.

- Queries using indexes.

  - Infinite Graph
  - Neo4j



When to use NoSQL DB

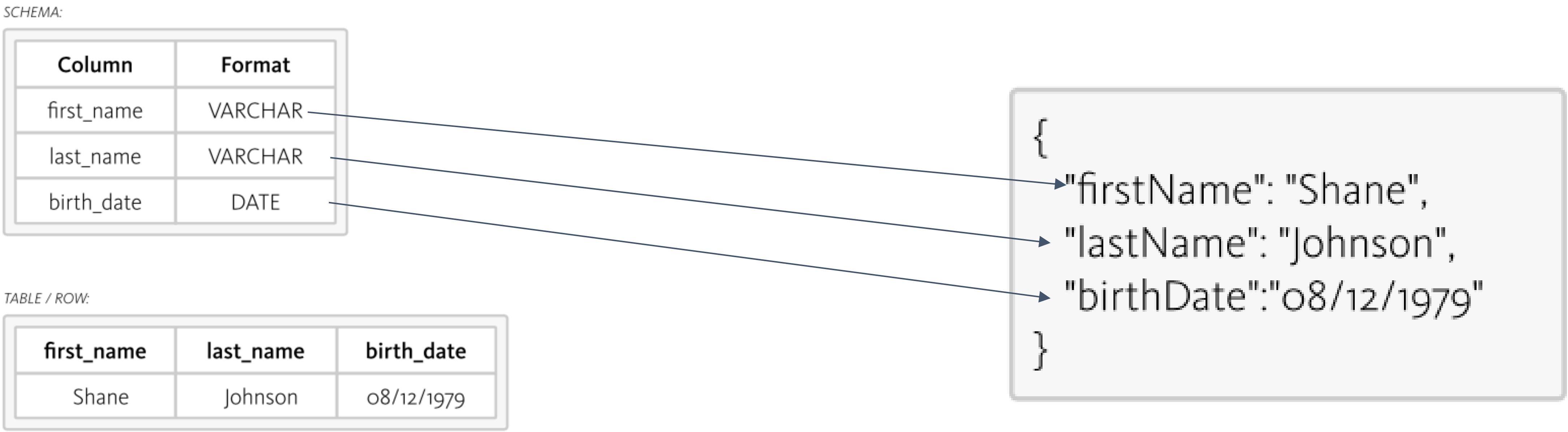


Some of the reasons that may lead us to use NoSQL databases instead of the classic SQL databases are:

- The volume of data grows very rapidly at specific moments, and can exceed one terabyte of information.
- The scalability of the relational solution is not feasible both at a cost and technical level.
- High peak usage of the system by users on multiple occasions.
- The database schema is not homogeneous, i.e. when in each data insertion the information stored may have different fields.

Example 1

What would be the structure of a JSON document?



SCHEMA:

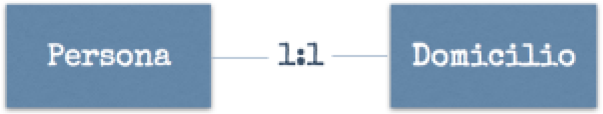
Column	Format
first_name	VARCHAR
last_name	VARCHAR
birth_date	DATE

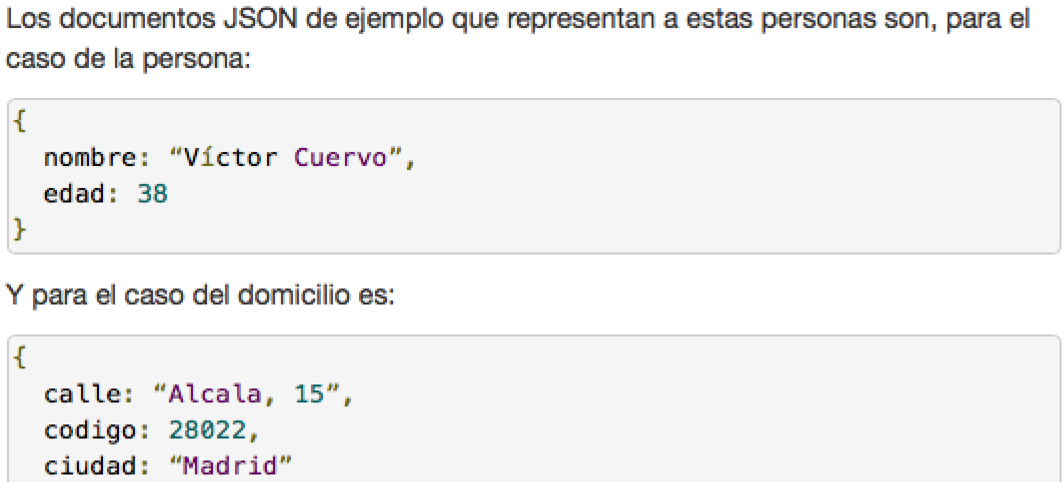
TABLE / ROW:

first_name	last_name	birth_date
Shane	Johnson	08/12/1979

```
{
 "firstName": "Shane",
 "lastName": "Johnson",
 "birthDate":"08/12/1979"
}
```

Example 2

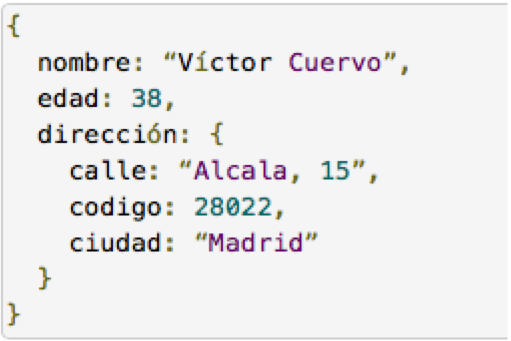



Los documentos JSON de ejemplo que representan a estas personas son, para el caso de la persona:

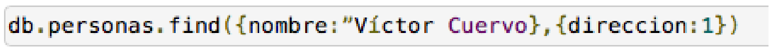
```
{
  nombre: “Víctor Cuervo”,
  edad: 38
}
```

Y para el caso del domicilio es:

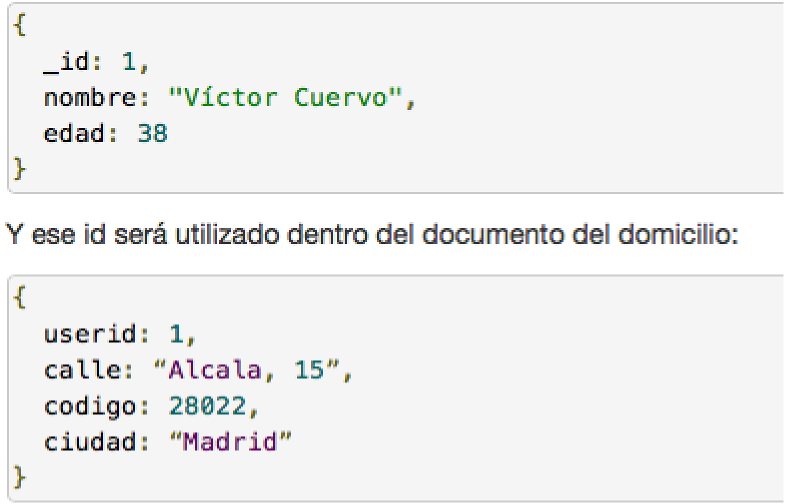
```
{
  calle: “Alcala, 15”,
  codigo: 28022,
  ciudad: “Madrid”
}
```



```
{
  nombre: “Víctor Cuervo”,
  edad: 38,
  dirección: {
    calle: “Alcala, 15”,
    codigo: 28022,
    ciudad: “Madrid”
  }
}
```



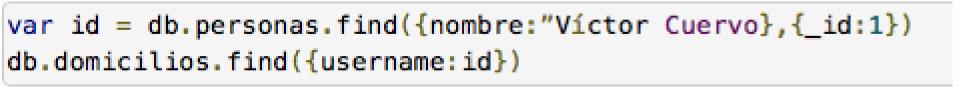
```
db.personas.find({nombre:”Víctor Cuervo},{direccion:1})
```



```
{
  _id: 1,
  nombre: "Víctor Cuervo",
  edad: 38
}
```

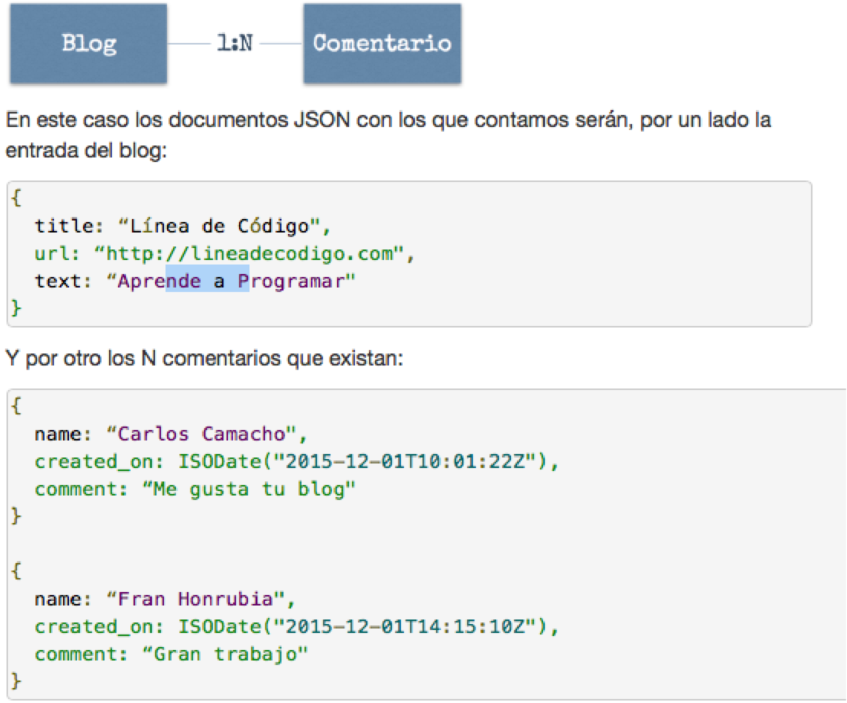
Y ese id será utilizado dentro del documento del domicilio:

```
{
  userid: 1,
  calle: “Alcala, 15”,
  codigo: 28022,
  ciudad: “Madrid”
}
```



```
var id = db.personas.find({nombre:”Víctor Cuervo},{_id:1})
db.domicilios.find({username:id})
```

Exercise

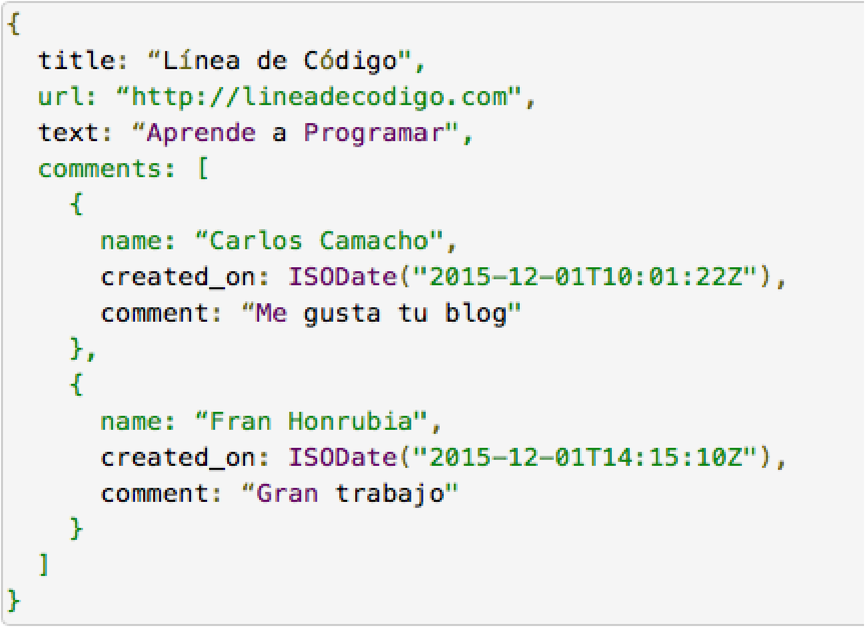


En este caso los documentos JSON con los que contamos serán, por un lado la entrada del blog:

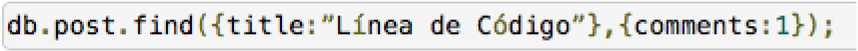
```
{
  title: “Línea de Código",
  url: “http://lineadecodigo.com",
  text: “Aprende a Programar"
}
```

Y por otro los N comentarios que existan:

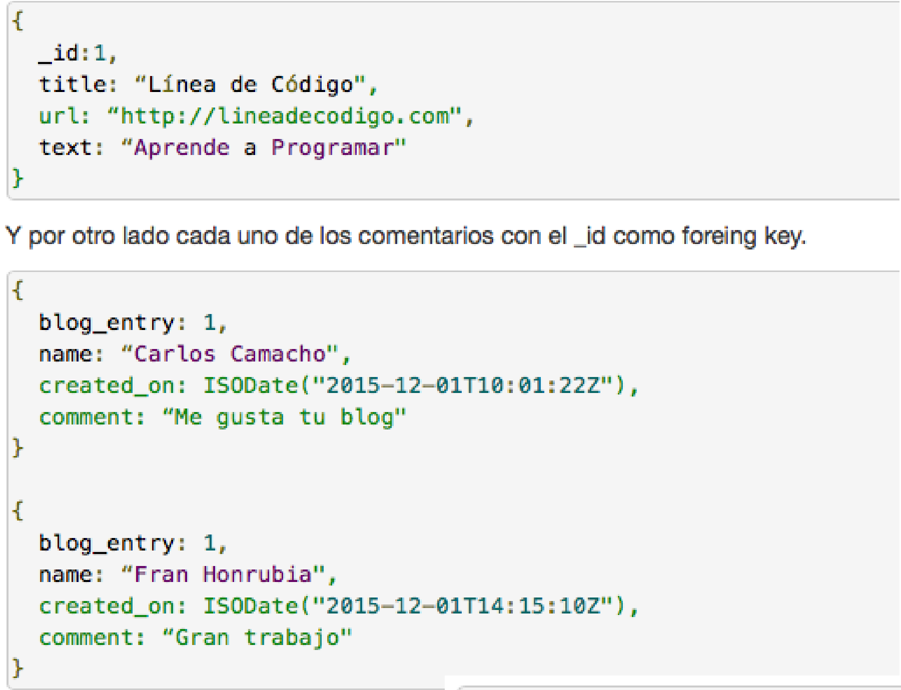
```
{
  name: “Carlos Camacho",
  created_on: ISODate("2015-12-01T10:01:22Z"),
  comment: “Me gusta tu blog"
}

{
  name: “Fran Honrubia",
  created_on: ISODate("2015-12-01T14:15:10Z"),
  comment: “Gran trabajo"
}
```



```
{
  title: “Línea de Código",
  url: “http://lineadecodigo.com",
  text: “Aprende a Programar",
  comments: [
    {
      name: “Carlos Camacho",
      created_on: ISODate("2015-12-01T10:01:22Z"),
      comment: “Me gusta tu blog"
    },
    {
      name: “Fran Honrubia",
      created_on: ISODate("2015-12-01T14:15:10Z"),
      comment: “Gran trabajo"
    }
  ]
}
```



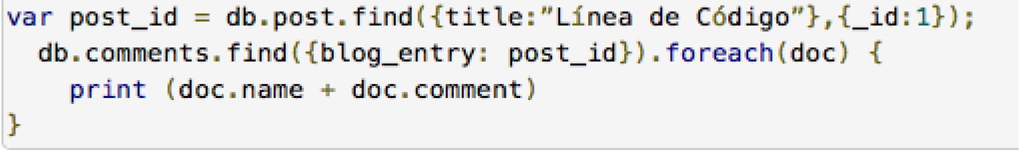
```
db.post.find({title:”Línea de Código”},{comments:1});
```



```
{
  _id:1,
  title: “Línea de Código",
  url: “http://lineadecodigo.com",
  text: “Aprende a Programar"
}
```

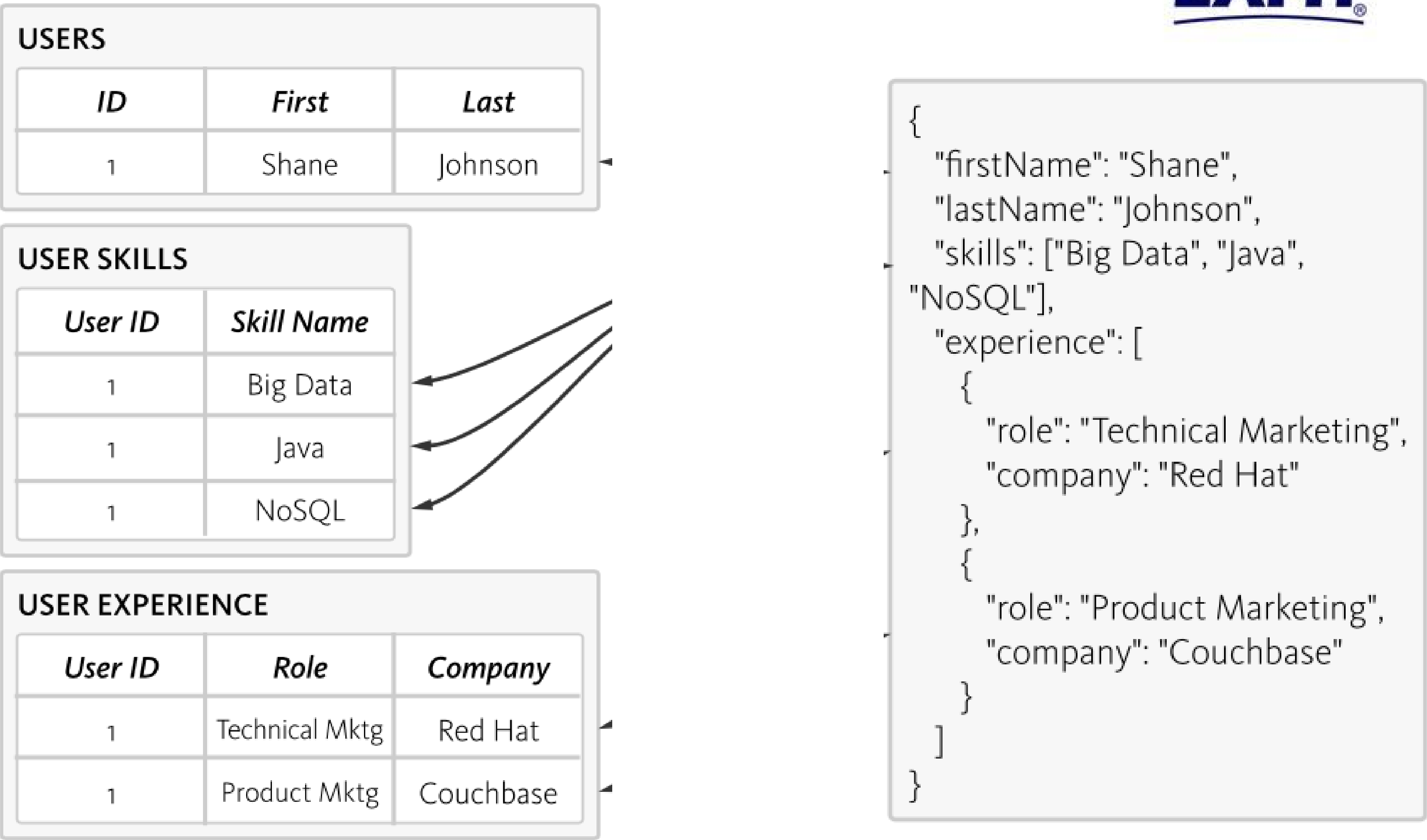
Y por otro lado cada uno de los comentarios con el _id como foreing key.

```
{
  blog_entry: 1,
  name: “Carlos Camacho",
  created_on: ISODate("2015-12-01T10:01:22Z"),
  comment: “Me gusta tu blog"
}

{
  blog_entry: 1,
  name: “Fran Honrubia",
  created_on: ISODate("2015-12-01T14:15:10Z"),
  comment: “Gran trabajo"
}
```



```
var post_id = db.post.find({title:”Línea de Código”},{_id:1});
  db.comments.find({blog_entry: post_id}).foreach(doc) {
    print (doc.name + doc.comment)
}
```

Example 3



USERS

ID	First	Last
1	Shane	Johnson

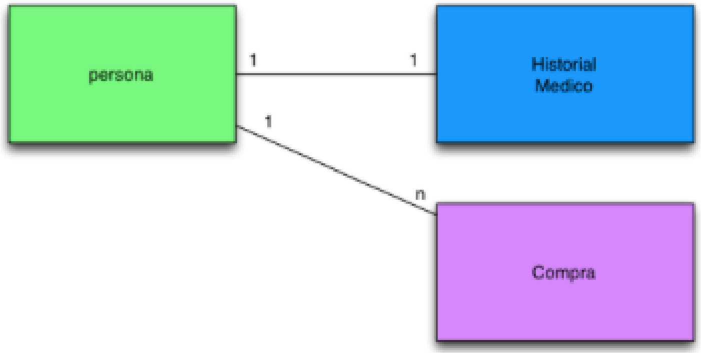
USER SKILLS

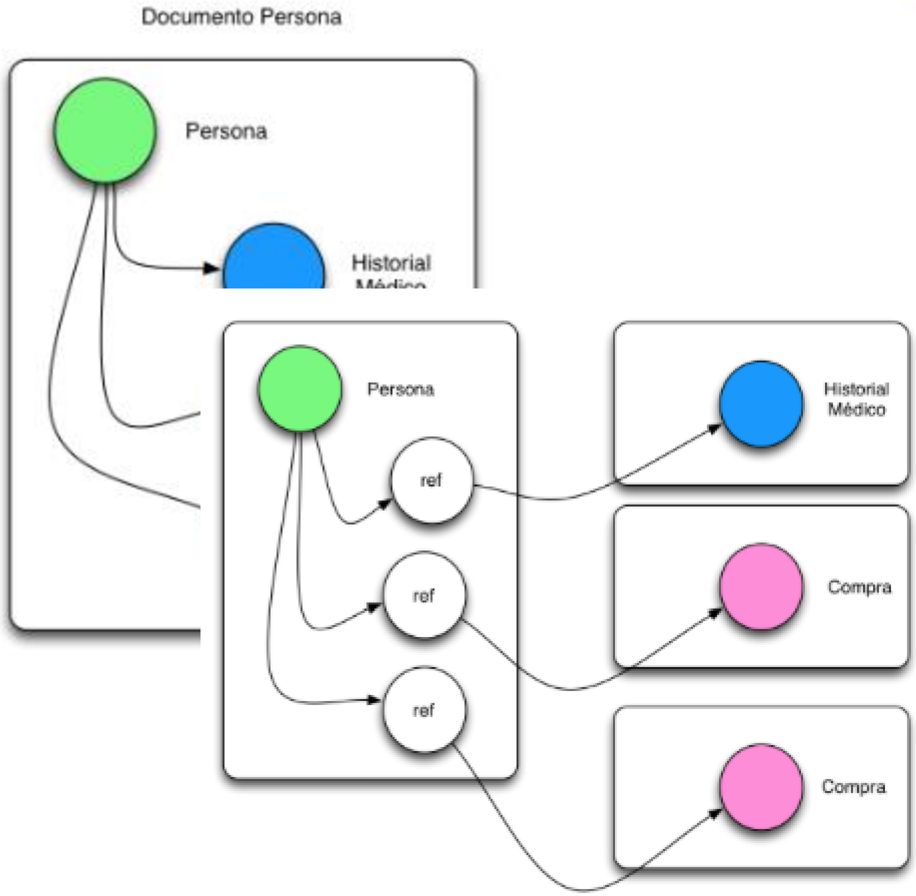
User ID	Skill Name
1	Big Data
1	Java
1	NoSQL

USER EXPERIENCE

User ID	Role	Company
1	Technical Mktg	Red Hat
1	Product Mktg	Couchbase

```
{
  "firstName": "Shane",
  "lastName": "Johnson",
  "skills": ["Big Data", "Java",
"NoSQL"],
  "experience": [
    {
      "role": "Technical Marketing",
      "company": "Red Hat"
    },
    {
      "role": "Product Marketing",
      "company": "Couchbase"
    }
  ]
}
```

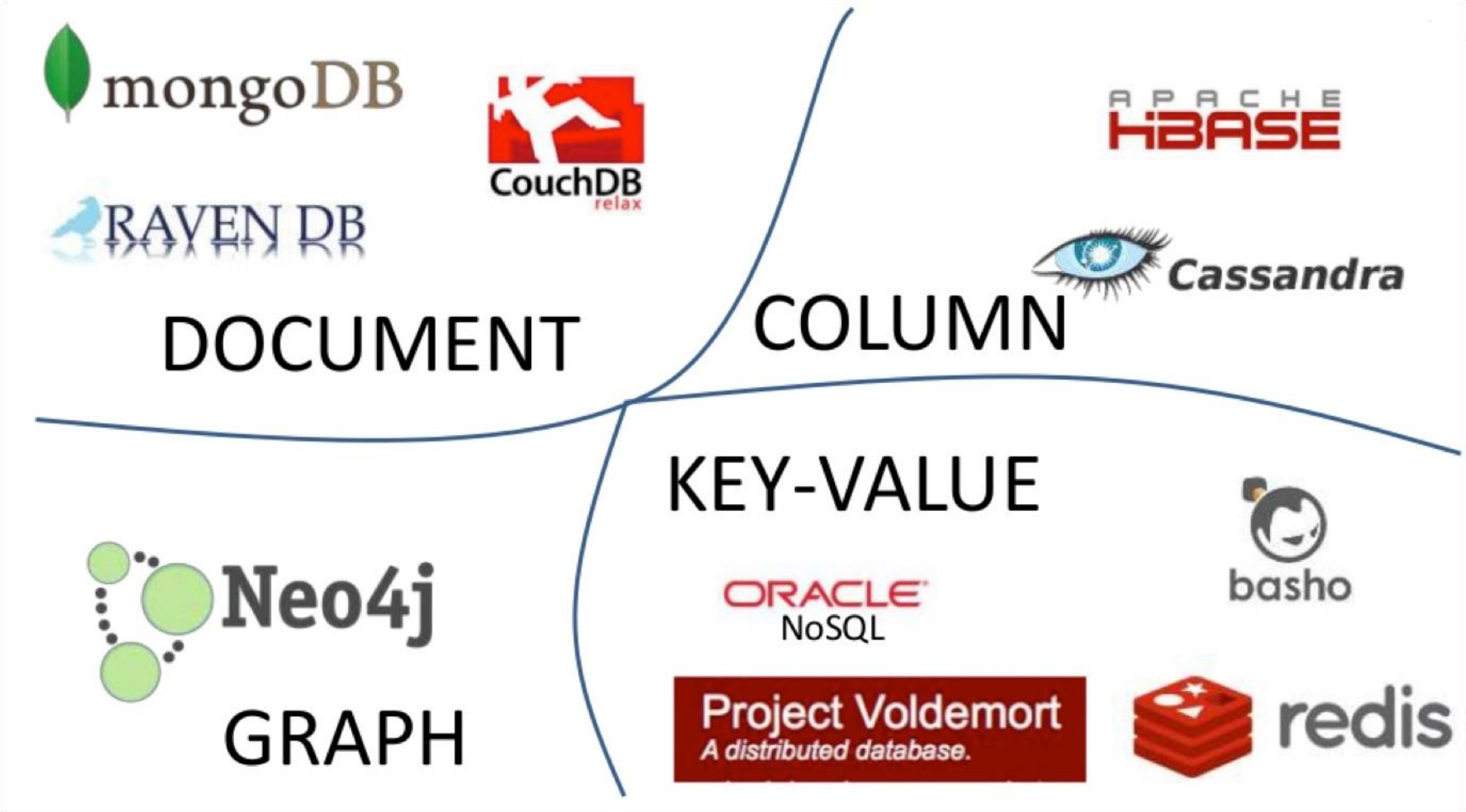




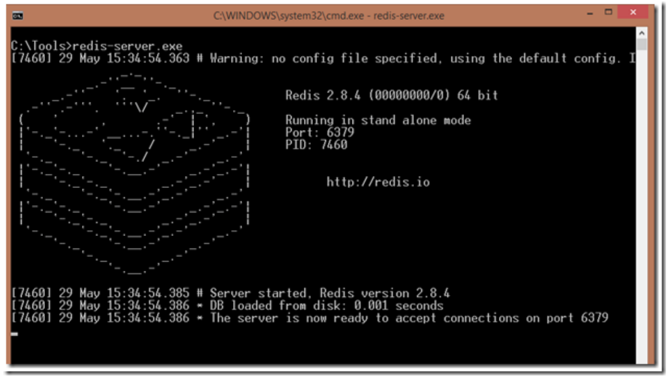
Cita Médica   Dirección del Paciente

Technologies

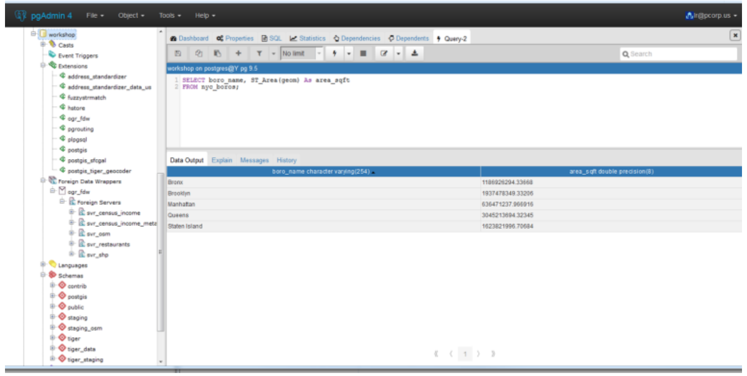
ORACLE®
Microsoft®
SQL Server®
PostgreSQL
MySQL®
SYBASE®
IBM
DB2®


mongoDB
CouchDB relax
RAVEN DB
DOCUMENT
APACHE HBASE
Cassandra
COLUMN
KEY-VALUE
basho
ORACLE NoSQL
Project Voldemort A distributed database.
redis
Neo4j
GRAPH

CLI o Graphical client



```
C:\Tools>redis-server.exe
[7460] 29 May 15:34:54.363 # Warning: no config file specified, using the default config. I

                                    Redis 2.8.4 (00000000/0) 64 bit

                                    Running in stand alone mode
                                    Port: 6379
                                    PID: 7460

                                         http://redis.io

[7460] 29 May 15:34:54.385 # Server started, Redis version 2.8.4
[7460] 29 May 15:34:54.386 * DB loaded from disk: 0.001 seconds
[7460] 29 May 15:34:54.386 * The server is now ready to accept connections on port 6379
```

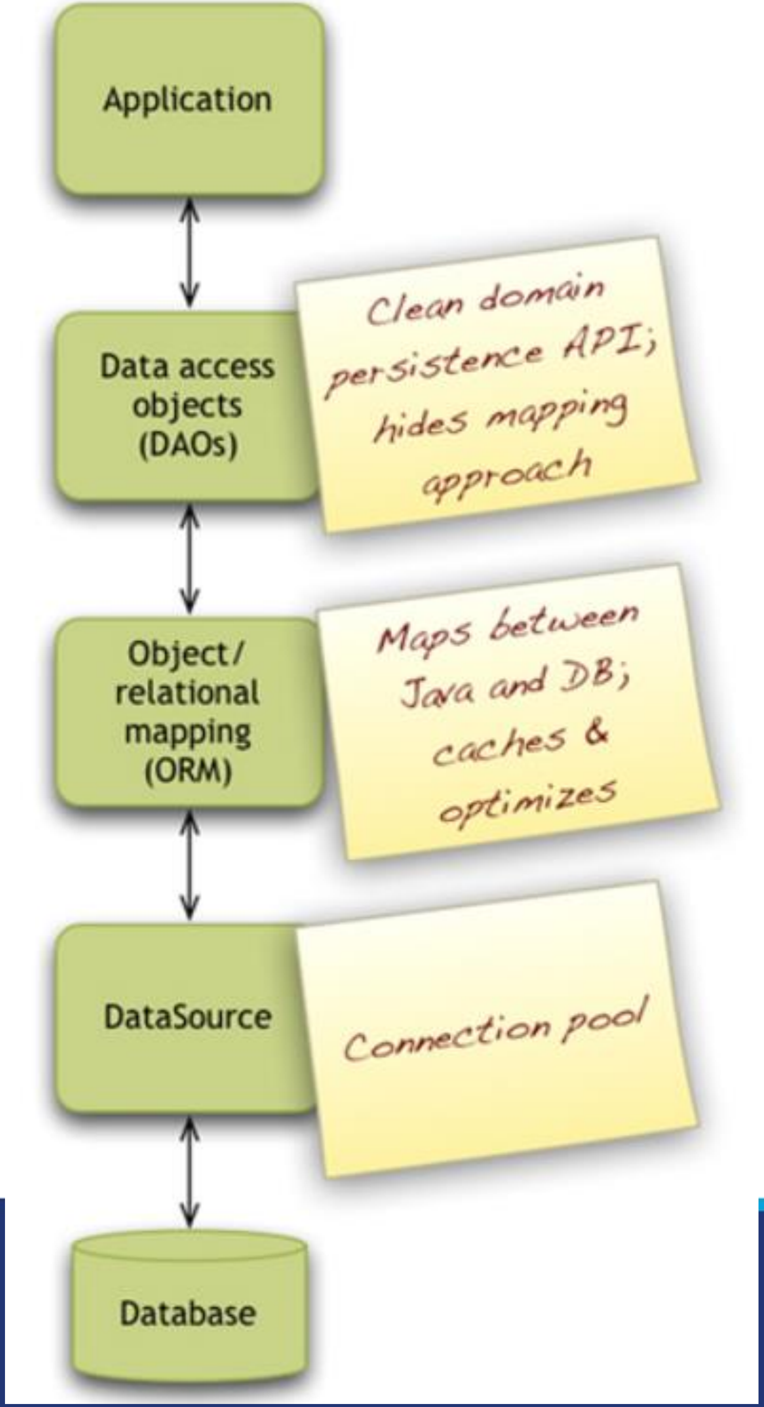


workshop on postgres@Y pg 9.5

```
SELECT boro_name, ST_Area(geom) As area_sqft
FROM nyc_boros;
```

boro_name character varying(254)	area_sqft double precision(8)
Bronx	1186926294.33668
Brooklyn	1937478349.33206
Manhattan	636471237.966916
Queens	3045213694.32345
Staten Island	1623821996.70684

Database Connectivity

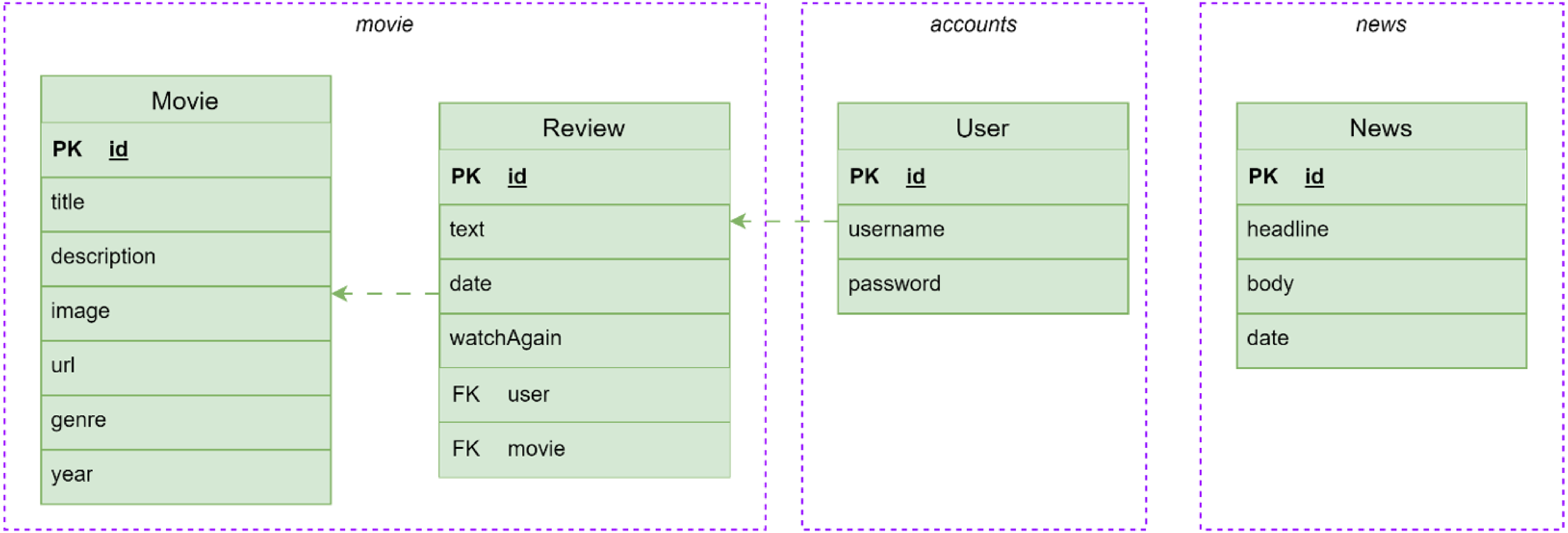

Application
Data access objects (DAOs)
Clean domain persistence API; hides mapping approach
Object/ relational mapping (ORM)
Maps between Java and DB; caches & optimizes
DataSource
Connection pool
Database

ORM (Object-relational mapping)

- System to map any data to the object structure.
- Frameworks mostly use ORM to interface to user code and covers the problematic to make the storage of objects.
- The mapping can be defined using XML files or metadata annotations.

- JPA (Java Persistence API)
- ORM with Spring

Example – MovieReviews:



movie

Movie
- **PK** **id**
- title
- description
- image
- url
- genre
- year

Review
- **PK** **id**
- text
- date
- watchAgain
- FK user
- FK movie

accounts

User
- **PK** **id**
- username
- password

news

News
- **PK** **id**
- headline
- body
- date

Activity (to report on the wiki):

Refine the domain model to design the Relational Database Model of your project.

Questions???

https://eafit-my.sharepoint.com/:v:/g/personal/pvallej3_eafit_edu_co/EUZnM6VwZclIpHpQSDaaS24BLAbrZ14-giwKLHSwxPYuyA?e=hCiJ6k&nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJTdHJlYW1XZWJBcHAiLCJyZWZlcnJhbFZpZXciOiJTaGFyZURpYWxvZy1MaW5rIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXcifX0%3D