
mongoDB


Apache
CASSANDRA™


APACHE
HBASE


amazon
DynamoDB


neo4j


RAVENDB


CouchDB
relax


redis


HYPERTABLE INC


OrientDB®

ACCIÓN FORMATIVA

DURACIÓN: 20 horas

MODALIDAD: Online

FECHAS y HORARIO: 8 de Julio a 12 de Julio de 9:30 a 13:30

CONTENIDOS

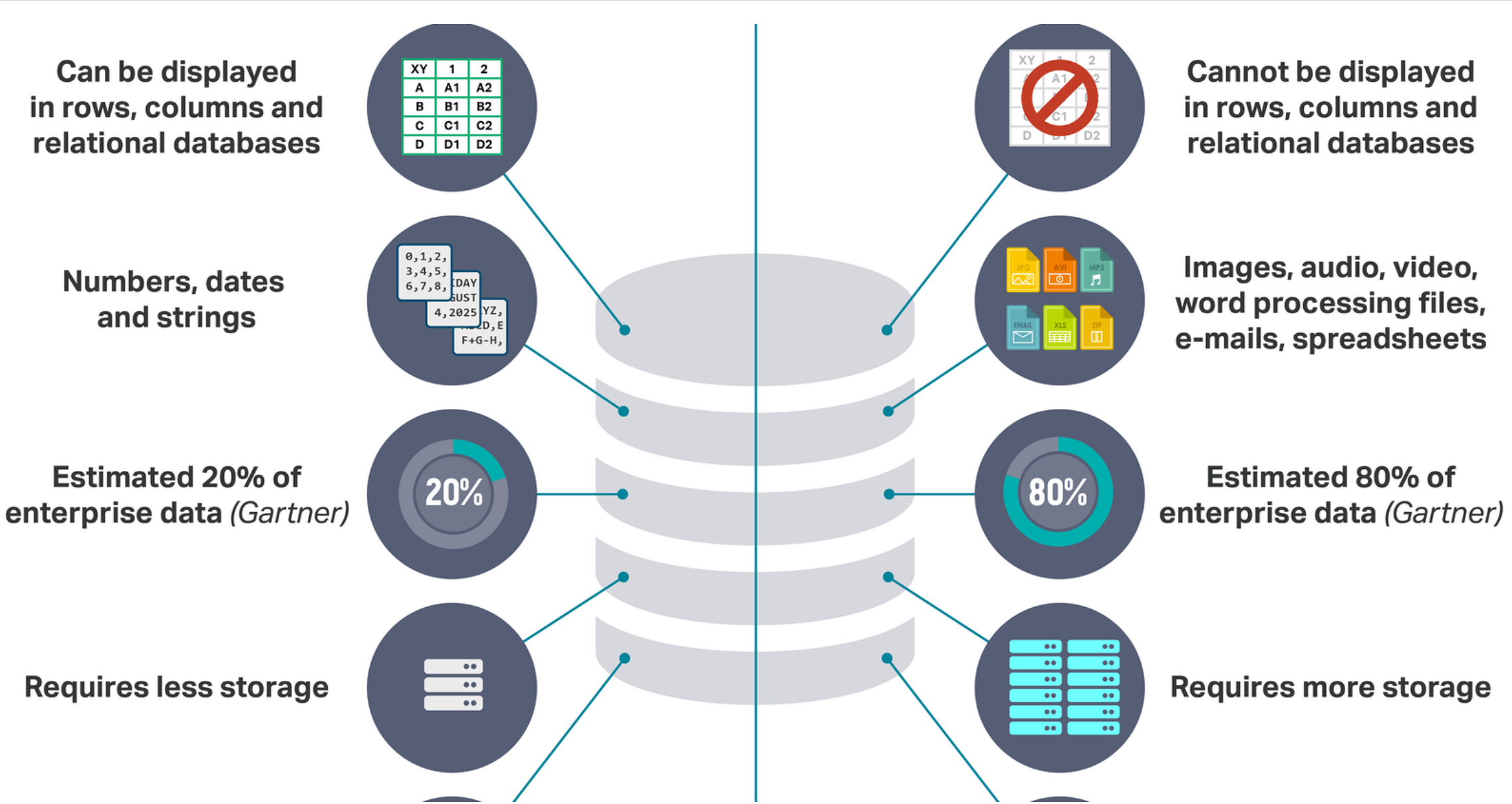
- Introducción.
- Bases de datos Relacionales.
- NoSQL.
- Redis.
- MongoDB.
- Apache Cassandra.
- Neo4J

PRESENTACIÓN

- NOMBRE Y UBICACIÓN
- EXPERIENCIA PREVIA CON BASES DE DATOS
- EXPERIENCIA PREVIA CON MONGODB
- EXPECTATIVAS DEL CURSO

STRUCTURED DATA

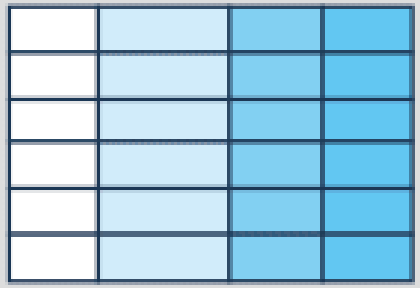
UNSTRUCTURED DATA


Can be displayed in rows, columns and relational databases
XY 1 2
A A1 A2
B B1 B2
C C1 C2
D D1 D2
Numbers, dates and strings
0,1,2,
3,4,5,
6,7,8,
4,2025
F+G-H,
Estimated 20% of enterprise data (Gartner)
20%
Requires less storage
Cannot be displayed in rows, columns and relational databases
Images, audio, video, word processing files, e-mails, spreadsheets
JPG
AVI
MP3
EMAIL
XLS
ZIP
Estimated 80% of enterprise data (Gartner)
80%
Requires more storage

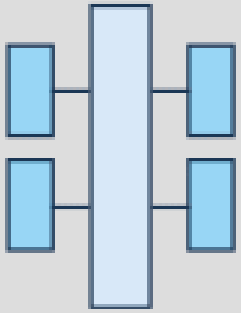

Cost reduction
BIG DATA ANALYTICS
Faster, better decision making
New products & services

SQL

Relational

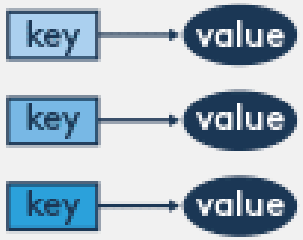


Analytical (OLAP)

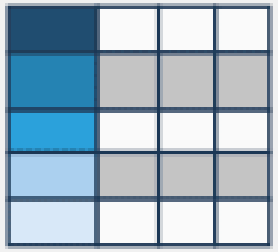


NoSQL

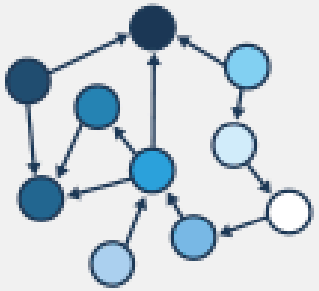
Key-Value



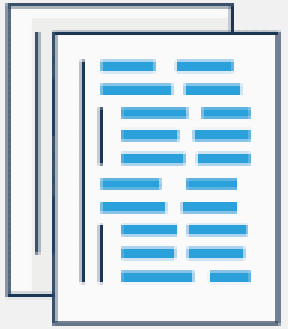
Column-Family

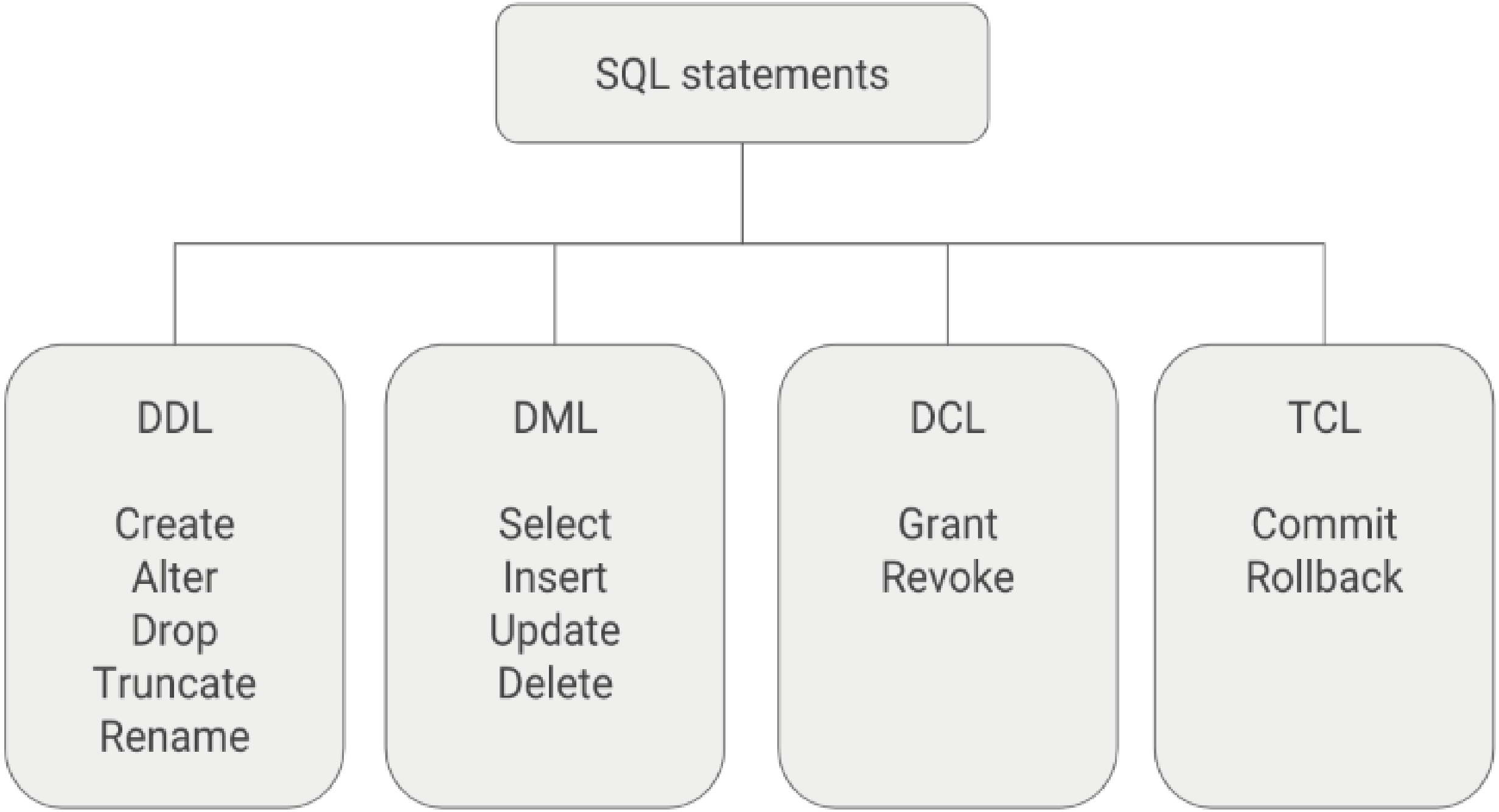


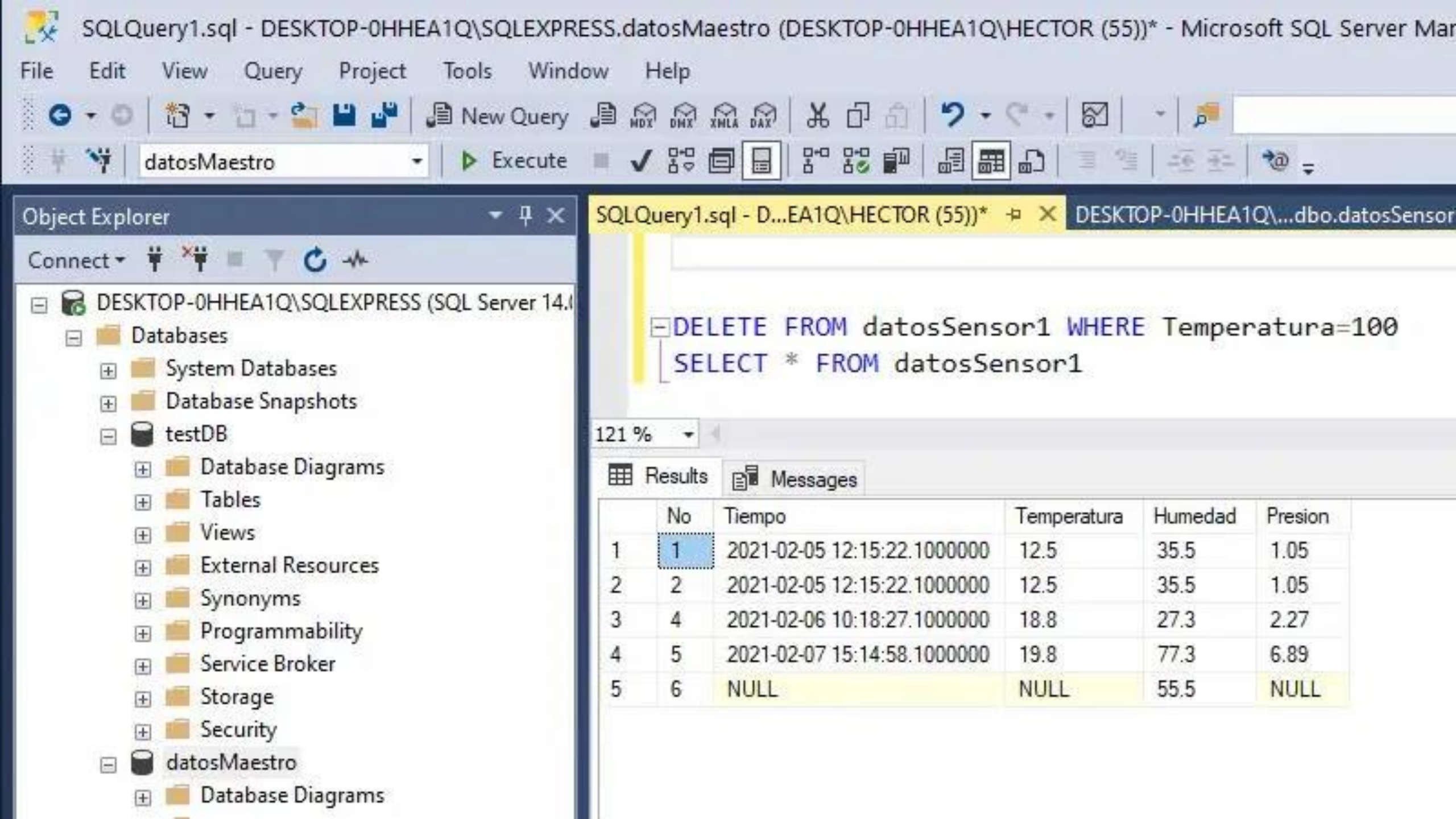
Graph



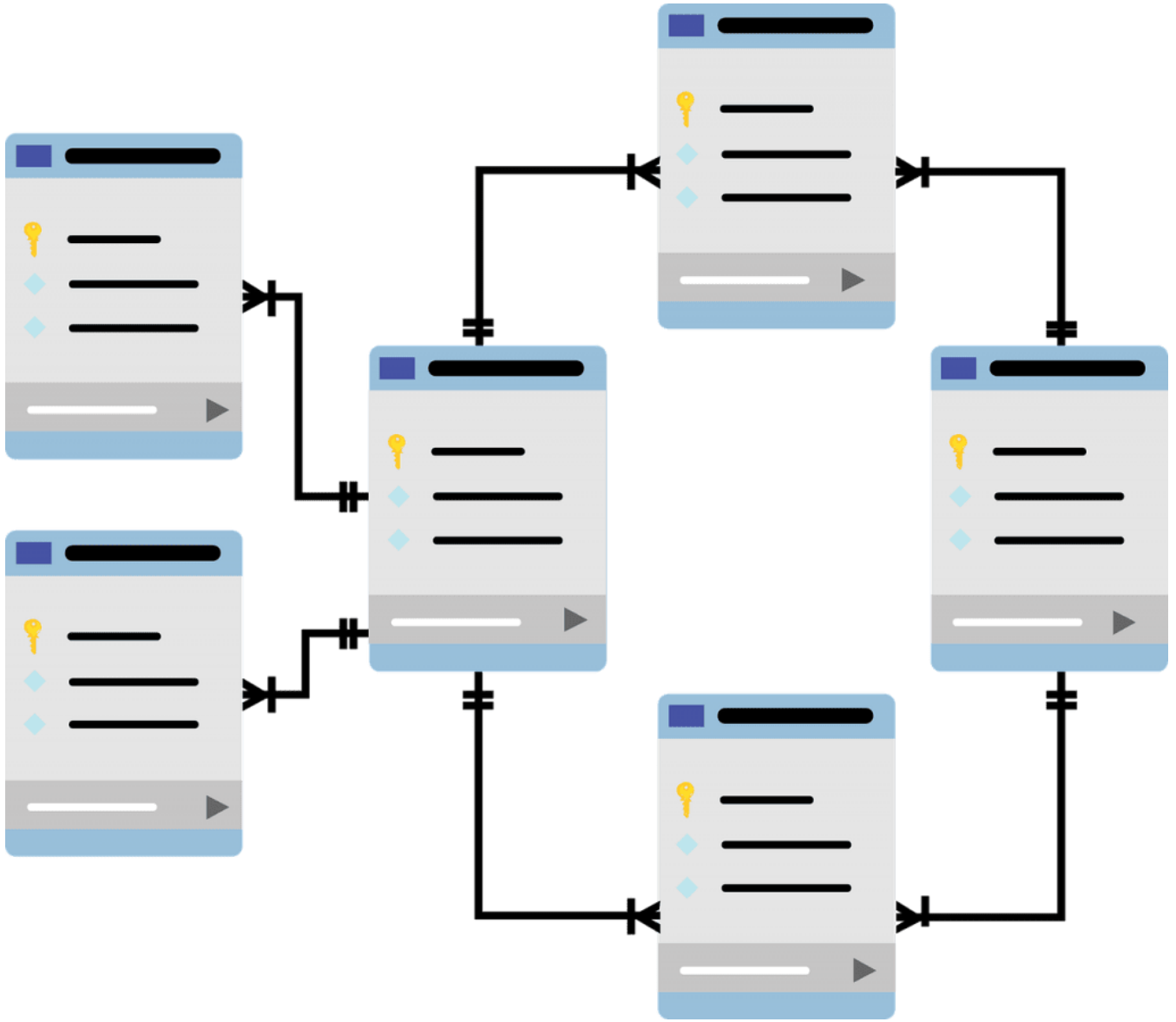
Document




SQL statements
DDL
Create
Alter
Drop
Truncate
Rename
DML
Select
Insert
Update
Delete
DCL
Grant
Revoke
TCL
Commit
Rollback


SQLQuery1.sql - DESKTOP-0HHEA1Q\SQLEXPRESS.datosMaestro (DESKTOP-0HHEA1Q\HECTOR (55))* - Microsoft SQL Server Mar
File Edit View Query Project Tools Window Help
New Query
datosMaestro
Execute
Object Explorer
Connect
DESKTOP-0HHEA1Q\SQLEXPRESS (SQL Server 14.
Databases
System Databases
Database Snapshots
testDB
Database Diagrams
Tables
Views
External Resources
Synonyms
Programmability
Service Broker
Storage
Security
datosMaestro
Database Diagrams
SQLQuery1.sql - D...EA1Q\HECTOR (55))*
DESKTOP-0HHEA1Q\...dbo.datosSensor
DELETE FROM datosSensor1 WHERE Temperatura=100
SELECT * FROM datosSensor1
121 %
Results
Messages
No Tiempo Temperatura Humedad Presion
1 1 2021-02-05 12:15:22.1000000 12.5 35.5 1.05
2 2 2021-02-05 12:15:22.1000000 12.5 35.5 1.05
3 4 2021-02-06 10:18:27.1000000 18.8 27.3 2.27
4 5 2021-02-07 15:14:58.1000000 19.8 77.3 6.89
5 6 NULL NULL 55.5 NULL

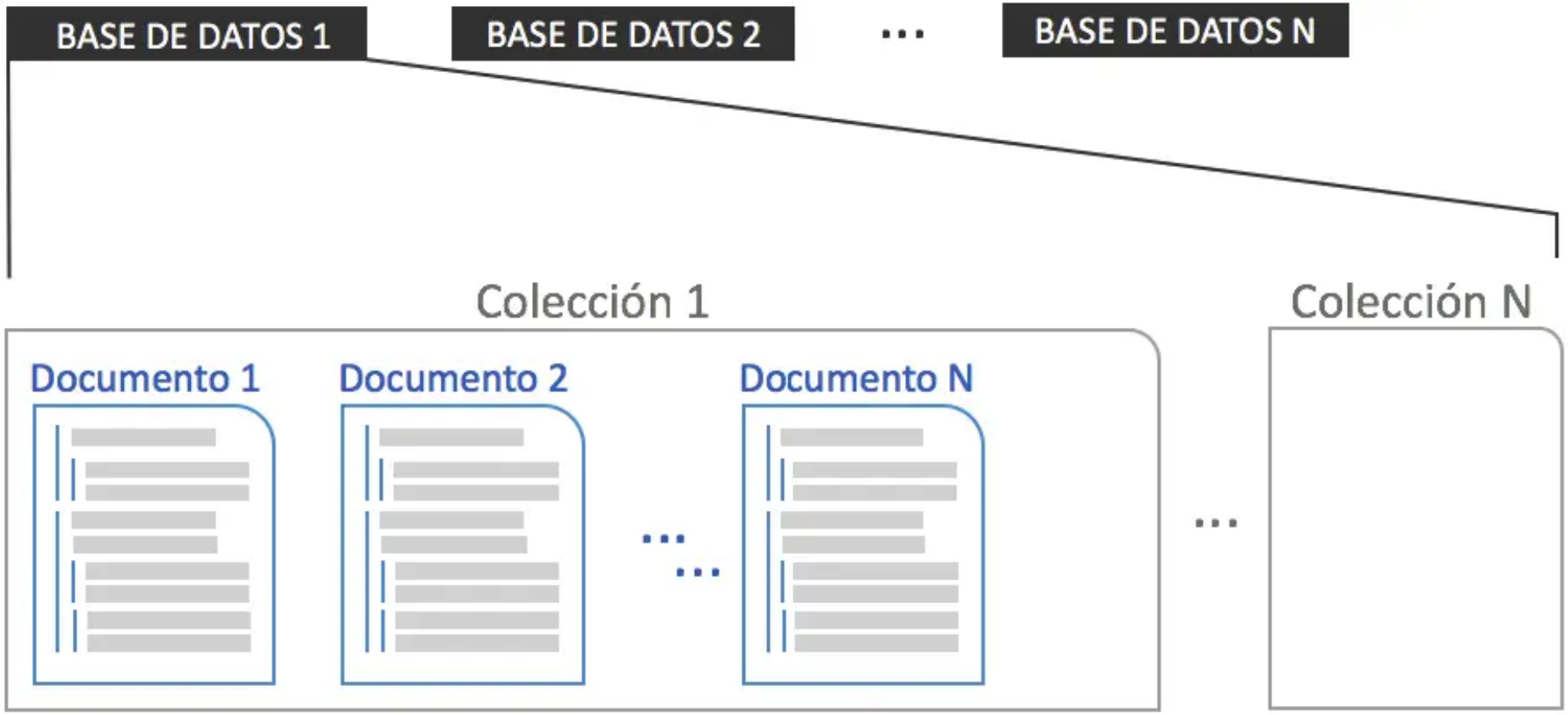
MODELO RELACIONAL

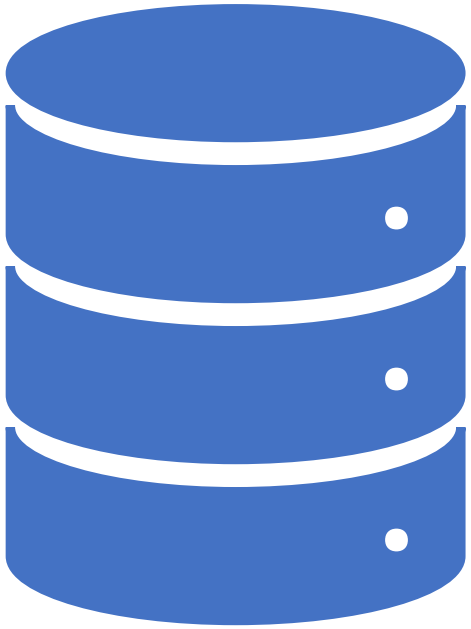



Microsoft Learn

MOTORES NOSQL

BASE DE DATOS DOCUMENTAL





MONGODB

- El gestor de base de datos MongoDB se lo puede asociar a un conjunto de gestores de bases de datos que no tienen como lenguaje principal el SQL para su manipulación.
- Los gestores de bases de datos NoSQL no requieren estructuras fijas como tablas, normalmente no soportan operaciones join y presentan como gran ventaja que pueden escalar en forma sencilla.

CARACTERÍSTICAS MONGODB

- Indexación
- Replicación
- Balanceo de carga
- Almacenamiento de archivos
- Agregación

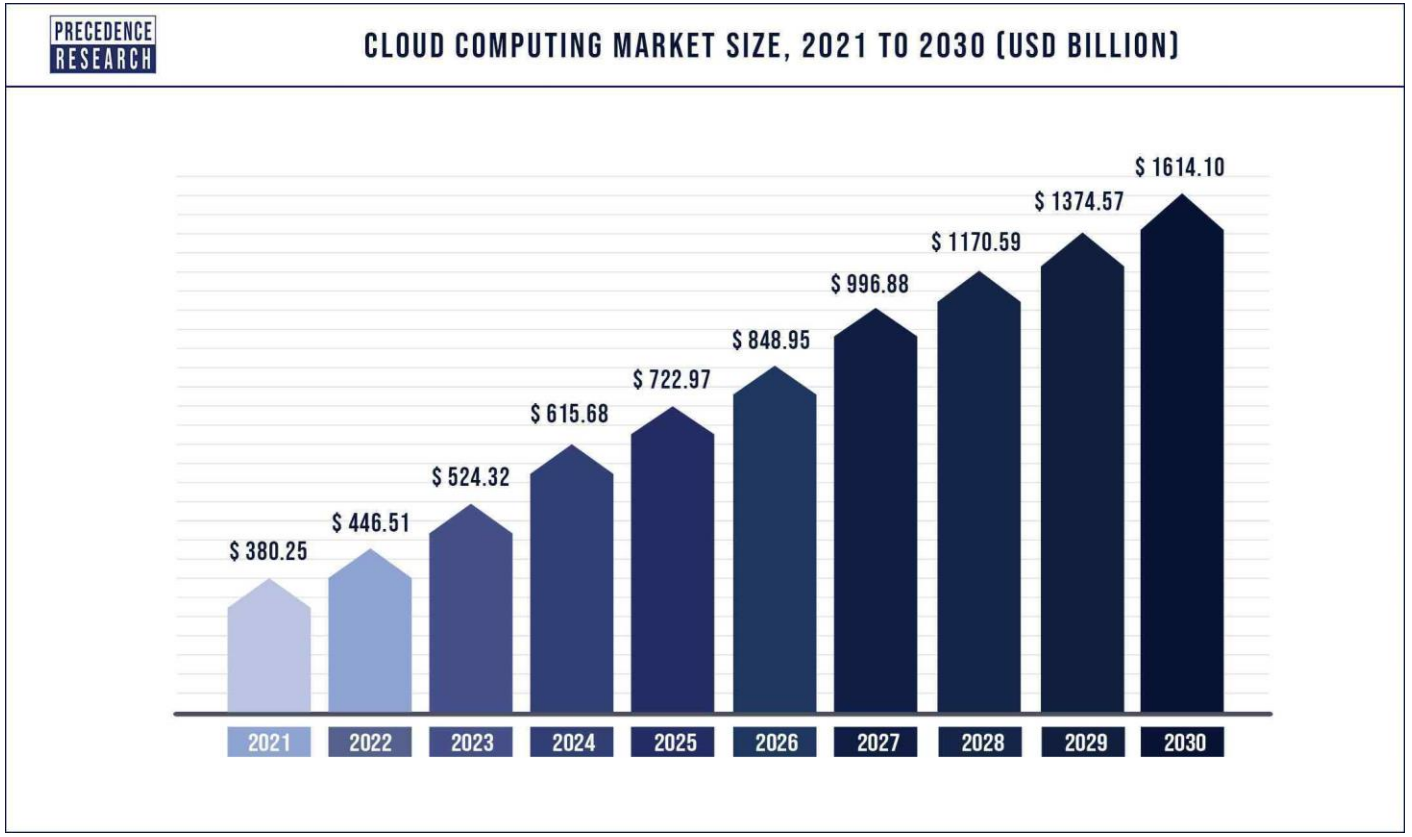
JSON

```
{
  codigo: 1,
  nombre: 'El aleph',
  autor: 'Borges',
  editoriales: ['Planeta','Siglo XXI']
}
```

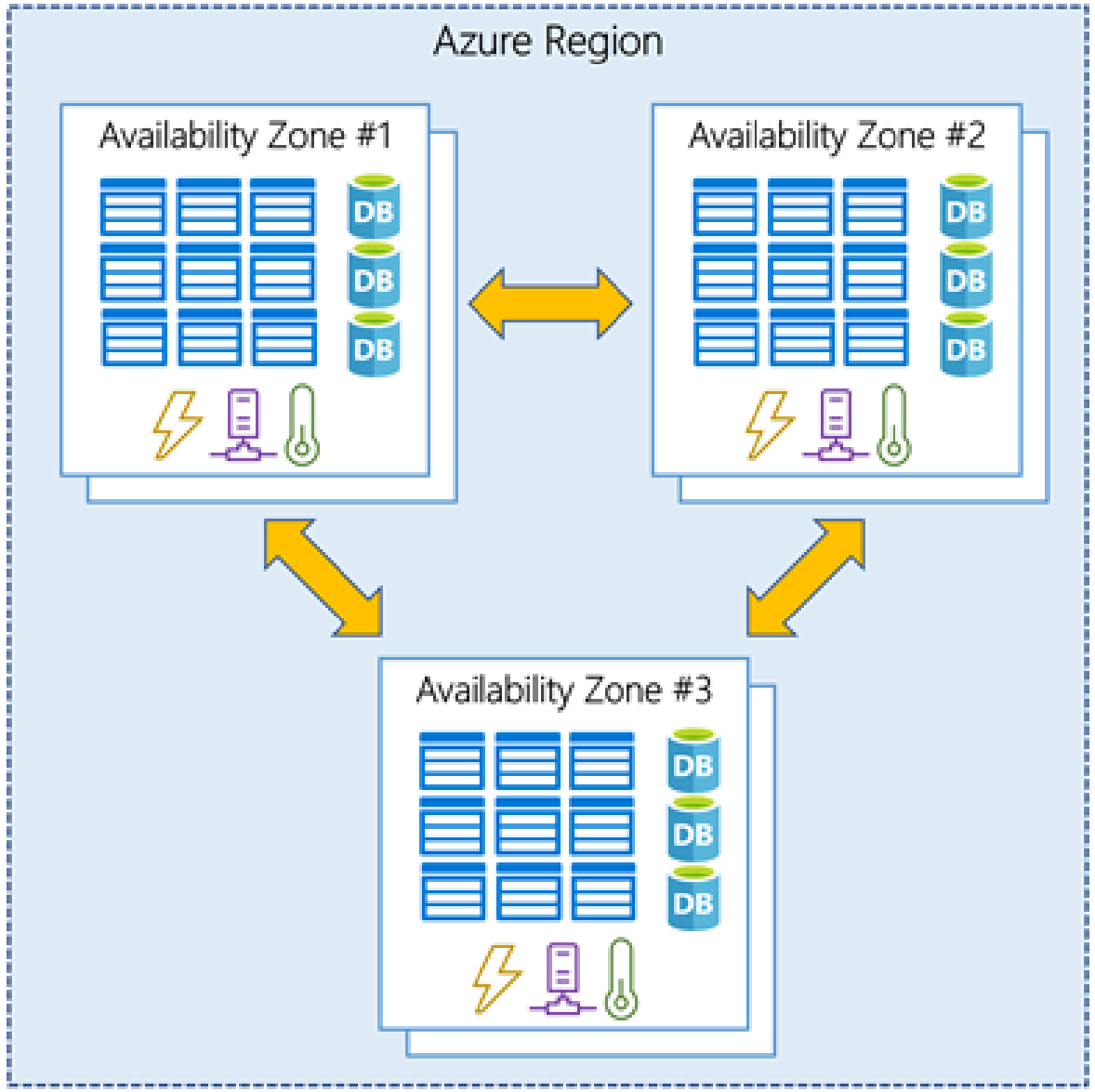
mongoDB® Atlas

Amazon DynamoDB

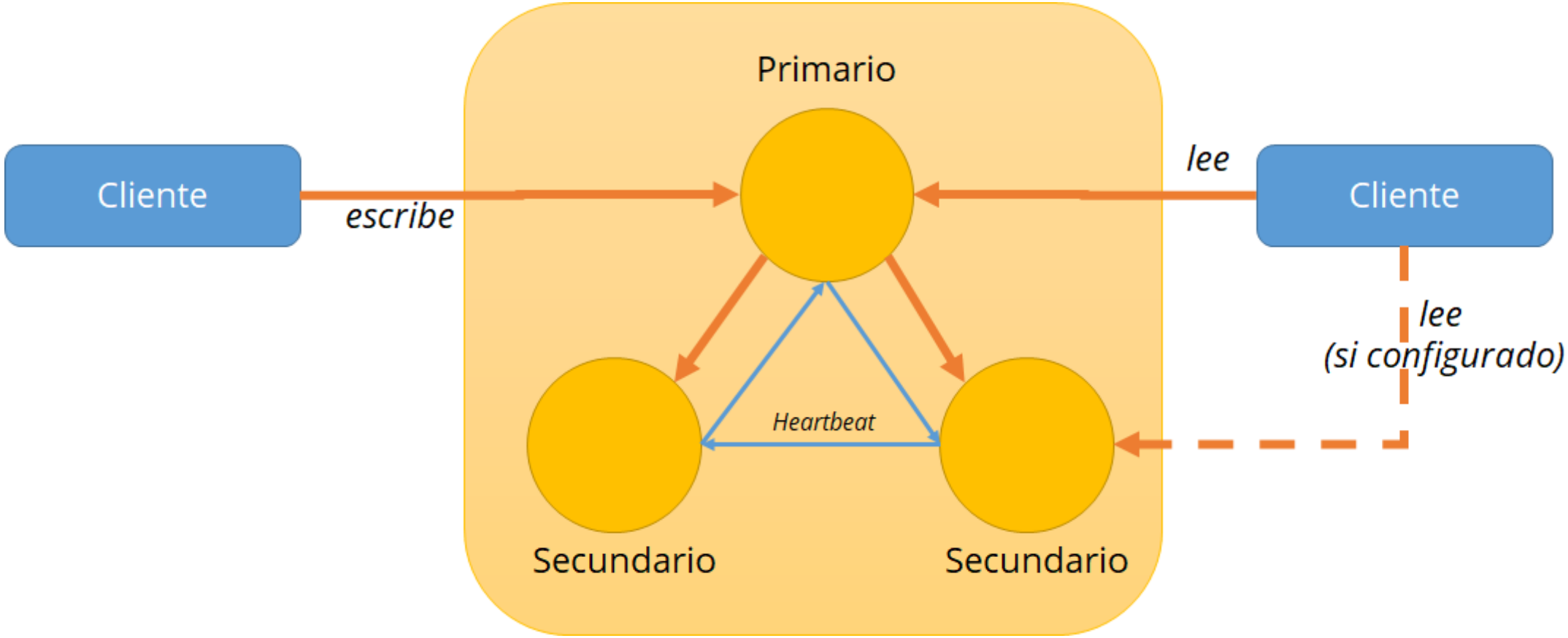
CLOUD COMPUTING

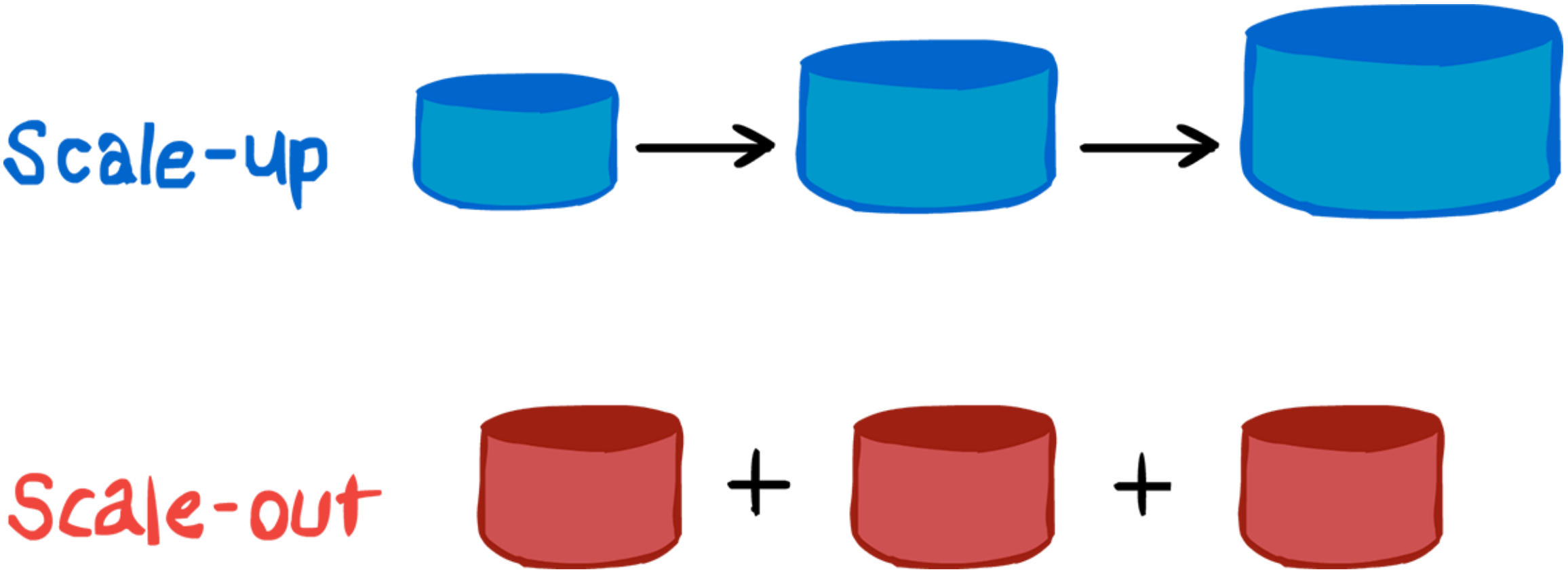


ALTA DISPONIBILIDAD



REPLICACIÓN

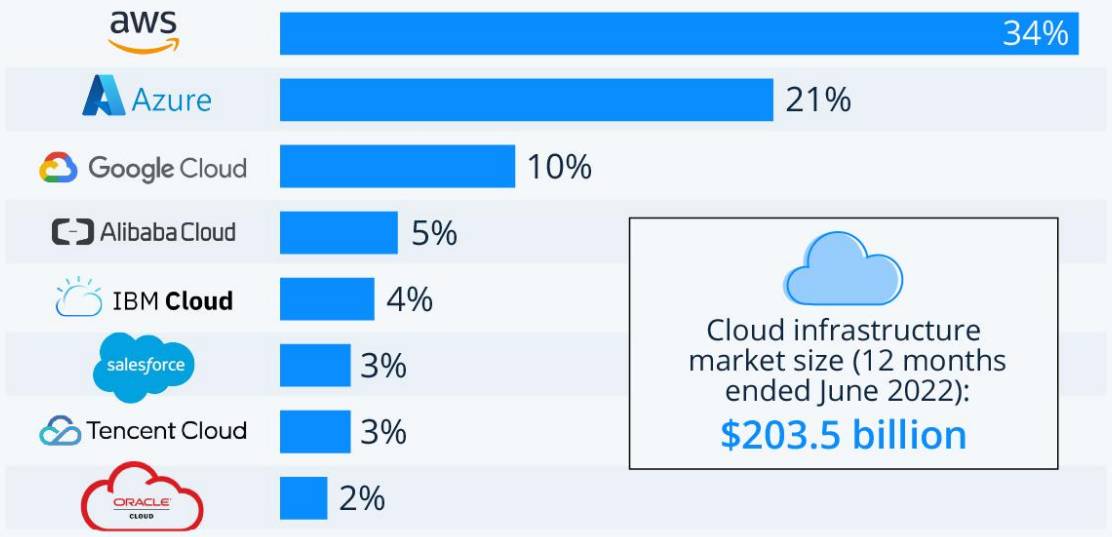




ESCALABILIDAD

Amazon Leads $200-Billion Cloud Market

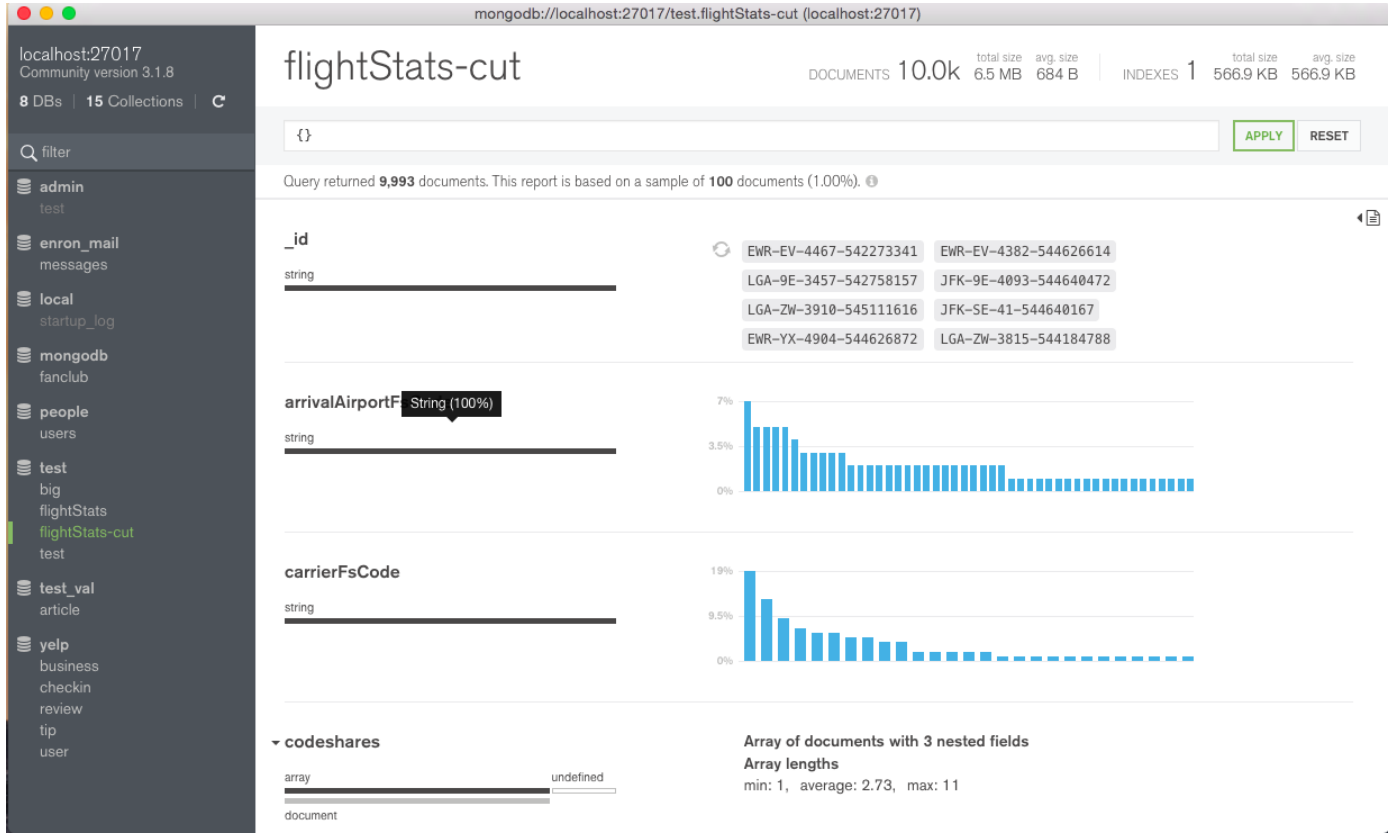
Worldwide market share of leading cloud infrastructure service providers in Q2 2022*



* includes platform as a service (PaaS) and infrastructure as a service (IaaS) as well as hosted private cloud services

Source: Synergy Research Group

MONGODB COMPASS




MongoDB
University
APK4Fun

myCompiler

Español | Reciente | Iniciar sesión | Inscríbase

Introduce un título...

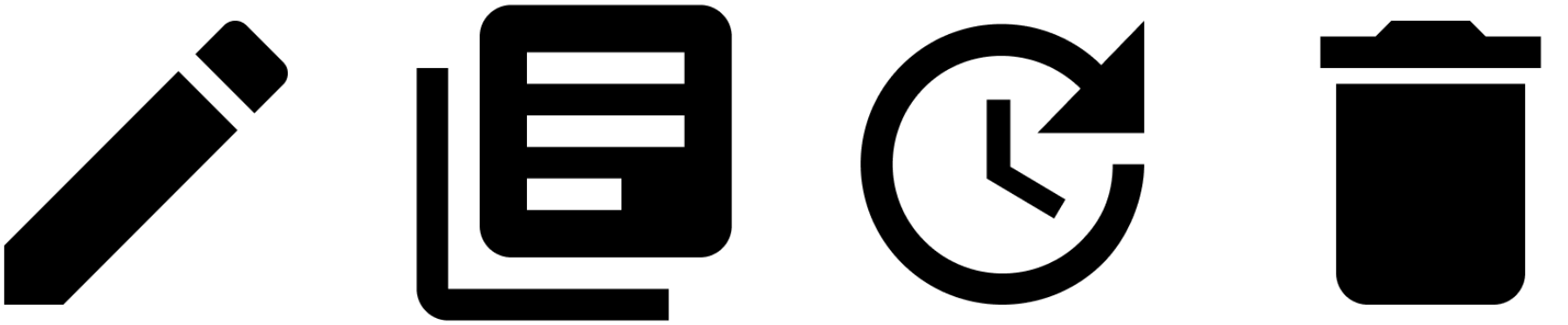
MongoDB

Ejecute | Guardar

```
1 db.students.insertMany([
2   { id: 1, name: 'Ryan', gender: 'M' },
3   { id: 2, name: 'Joanna', gender: 'F' }
4 ]);
5 db.students.find({ gender: 'F' });
```

Salida del programa

```
(Ejecute el programa para ver su salida)
```



Create Read Update Delete

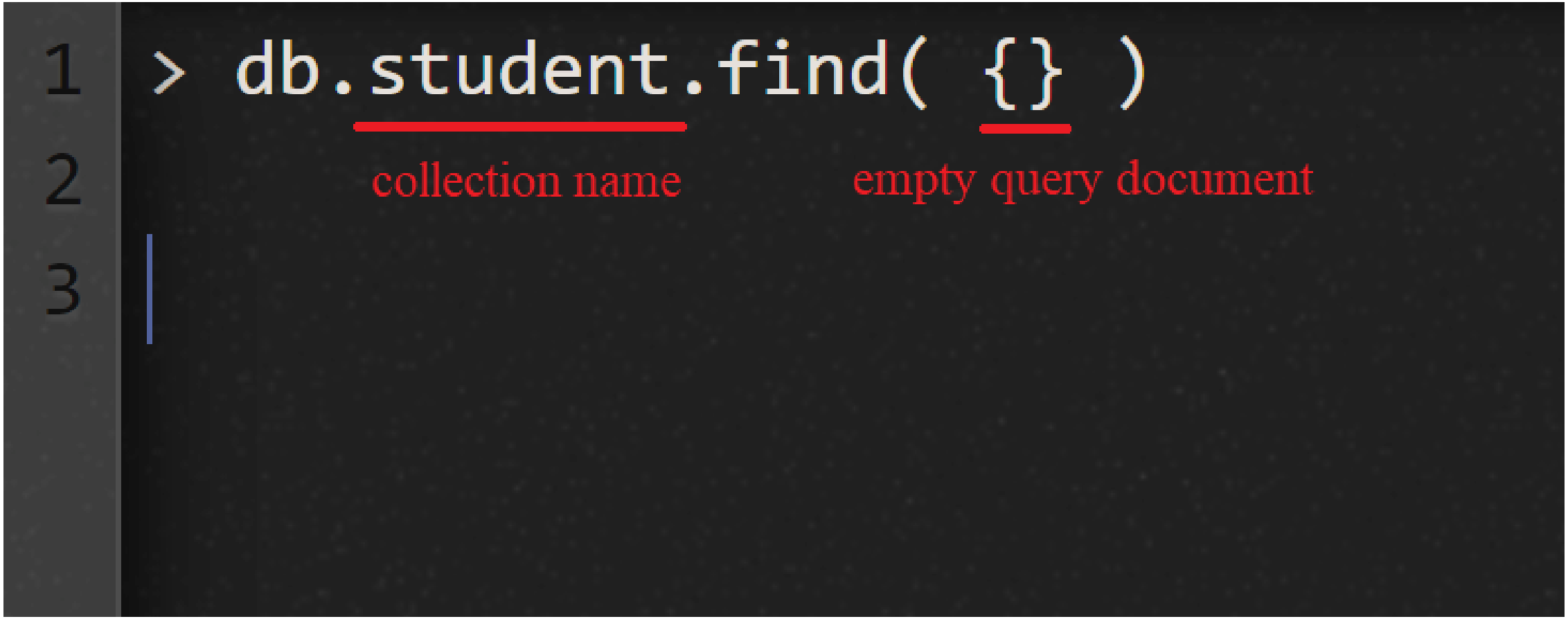
INSERTAR DOCUMENTOS

Para inserta un documento o un conjunto de documentos disponemos de los métodos:

- insertOne: Inserta un documento en una colección.
- insertMany: Inserta múltiples documentos en una colección.

CAMPO OBLIGATORIO _ID

- En MongoDB, cada documento almacenado en una colección requiere un único _id que actúa como clave principal . Si se inserta documento omite el _id, el controlador MongoDB automáticamente genera un ObjectId para el _id.



RECUPERAR DOCUMENTOS

OPERADORES COMPARACIÓN

- $eq - equal - igual
- $lt - low than - menor que
- $lte - low than equal - menor o igual que
- $gt - greater than - mayor que
- $gte - greater than equal - mayor o igual que
- $ne - not equal - distinto
- $in - in - dentro de
- $nin - not in - no dentro de

db.collection.deleteOne()

db.collection.deleteMany()

db.collection.remove()

ELIMINAR DOCUMENTOS Y BBDD

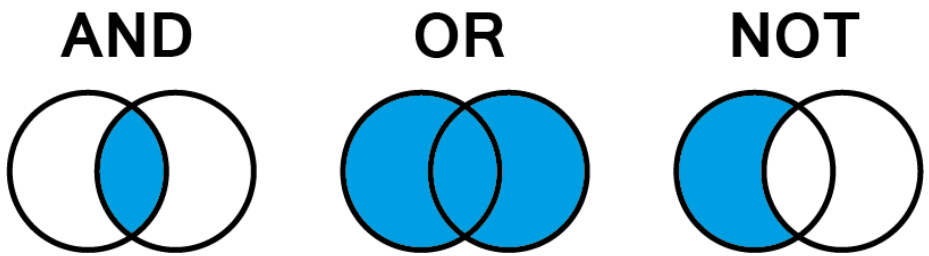
MODIFICAR UN ELEMENTO

```
> db.job.updateOne({salary:5000}, { $set: {firstName:'Morgan'}})
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
> db.job.find().pretty()
{
        "_id" : ObjectId("6286f50681063f5a55d4775b"),
        "firstName" : "Morgan",
        "lastName" : "Dew",
        "email" : "john.dew@abc.com",
        "salary" : 5000
}
```

MODIFICAR MÚLTIPLES ELEMENTOS

```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

OPERADORES LÓGICOS



CURSORES Y MÉTODOS

RECUPERAR SOLO ALGUNOS CAMPOS

- Hemos visto que el método 'find’:
  - Si no le pasamos parámetros nos retorna todos los documentos de la colección que hace referencia: db.libros.find({precio: 50},{titulo:1,cantidad:1,_id:0})
  - El primer parámetro en el caso que lo indiquemos filtra la colección y recupera los documentos que cumplen la condición
  - En el segundo parámetro del método 'find' debemos especificar cada campo y un valor 1 indicando que se lo quiere recuperar.

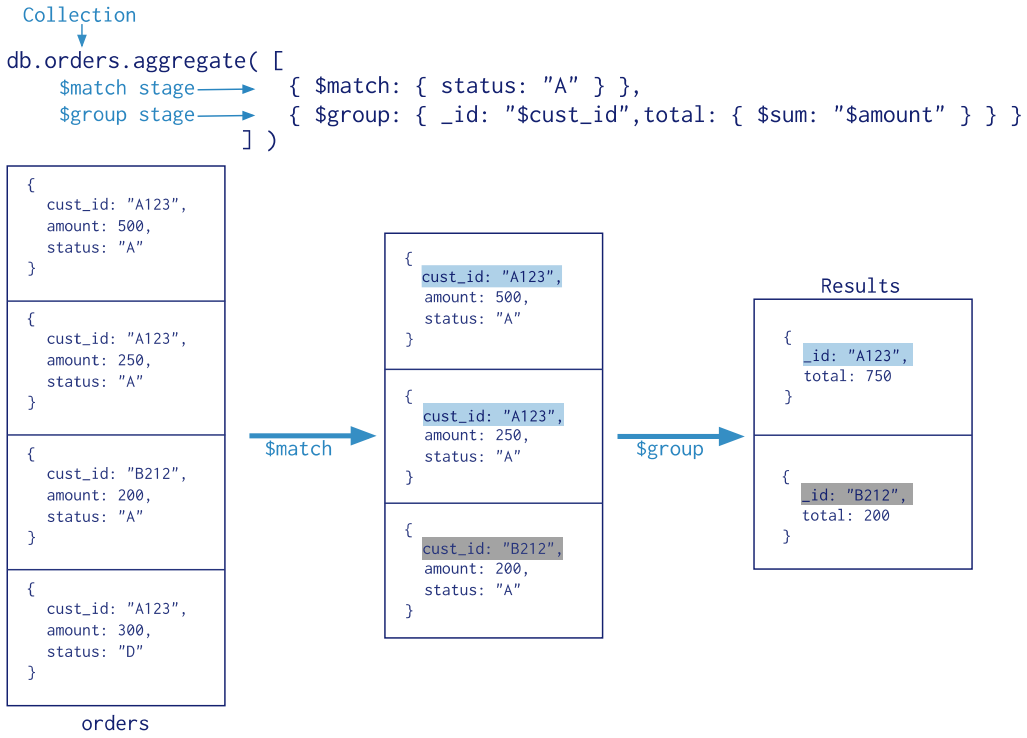
DOCUMENTOS EMBEBIDOS

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
             phone: "123-456-7890",
             email: "xyz@example.com"
           },
  access: {
             level: 5,
             group: "dev"
           }
}
```

Embedded sub-document

Embedded sub-document

AGGREGATION PIPELINE



$match

```
db.universities.aggregate([
  { $match : { country : 'Spain', city : 'Salamanca' } }
]).pretty()
```

$project

```
db.universities.aggregate([
  { $project : { _id : 0, country : 1, city : 1, name : 1 } }
]).pretty()
```

$group

```
db.universities.aggregate([
  { $group : { _id : '$name', totaldocs : { $sum : 1 } } }
]).pretty()
```

$out

```
db.universities.aggregate([
  { $group : { _id : '$name', totaldocs : { $sum : 1 } } },
  { $out : 'aggResults' }
])
```

$unwind

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $unwind : '$students' }
]).pretty()
```

$sort

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $unwind : '$students' },
  { $project : { _id : 0, 'students.year' : 1, 'students.number' : 1 } },
  { $sort : { 'students.number' : -1 } }
]).pretty()
```

$limit

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $unwind : '$students' },
  { $project : { _id : 0, 'students.year' : 1, 'students.number' : 1 } },
  { $sort : { 'students.number' : -1 } },
  { $limit : 2 }
]).pretty()
```

$addFields

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $addFields : { foundation_year : 1218 } }
]).pretty()
```

$count

```
db.universities.aggregate([
  { $unwind : '$students' },
  { $count : 'total_documents' }
]).pretty()
```

$lookup

```
db.universities.aggregate([
  { $match : { name : 'USAL' } },
  { $project : { _id : 0, name : 1 } },
  { $lookup : {
      from : 'courses',
      localField : 'name',
      foreignField : 'university',
      as : 'courses'
  } }
]).pretty()
```

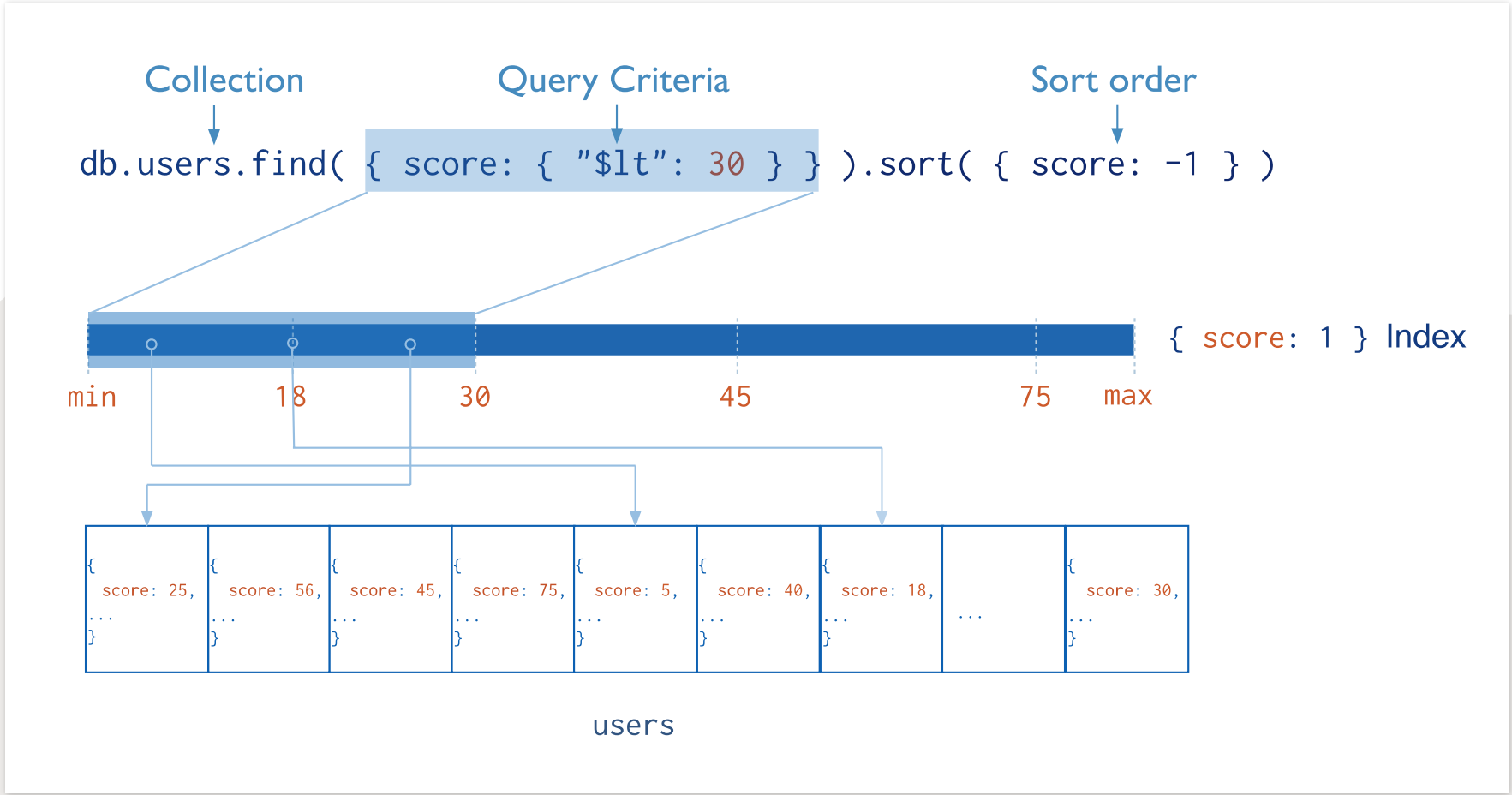
$sortByCount

```
db.courses.aggregate([
  { $sortByCount : '$level' }
]).pretty()
```

SQL VS MONGODB

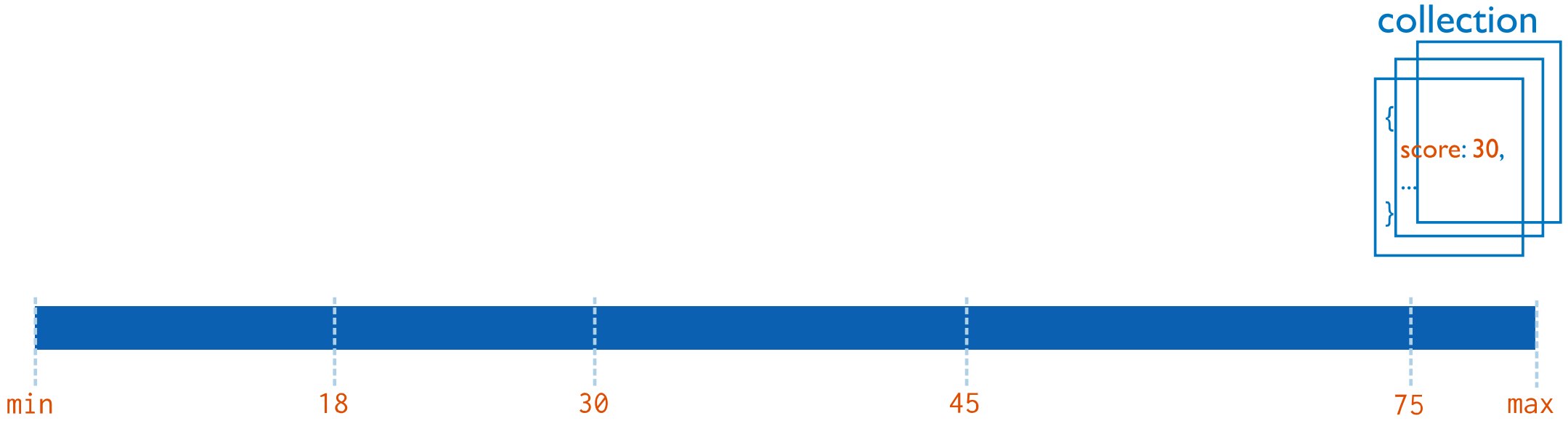
- https://www.mongodb.com/docs/manual/reference/sql-aggregation-comparison/

SQL Terms, Functions, and Concepts	MongoDB Aggregation Operators
WHERE	`$match`
GROUP BY	`$group`
HAVING	`$match`
SELECT	`$project`
ORDER BY	`$sort`
LIMIT	`$limit`
SUM()	`$sum`
COUNT()	`$sum` `$sortByCount`
join	`$lookup`
SELECT INTO NEW_TABLE	`$out`
MERGE INTO TABLE	`$merge` (Available starting in MongoDB 4.2)
UNION ALL	`$unionWith` (Available starting in MongoDB 4.4)

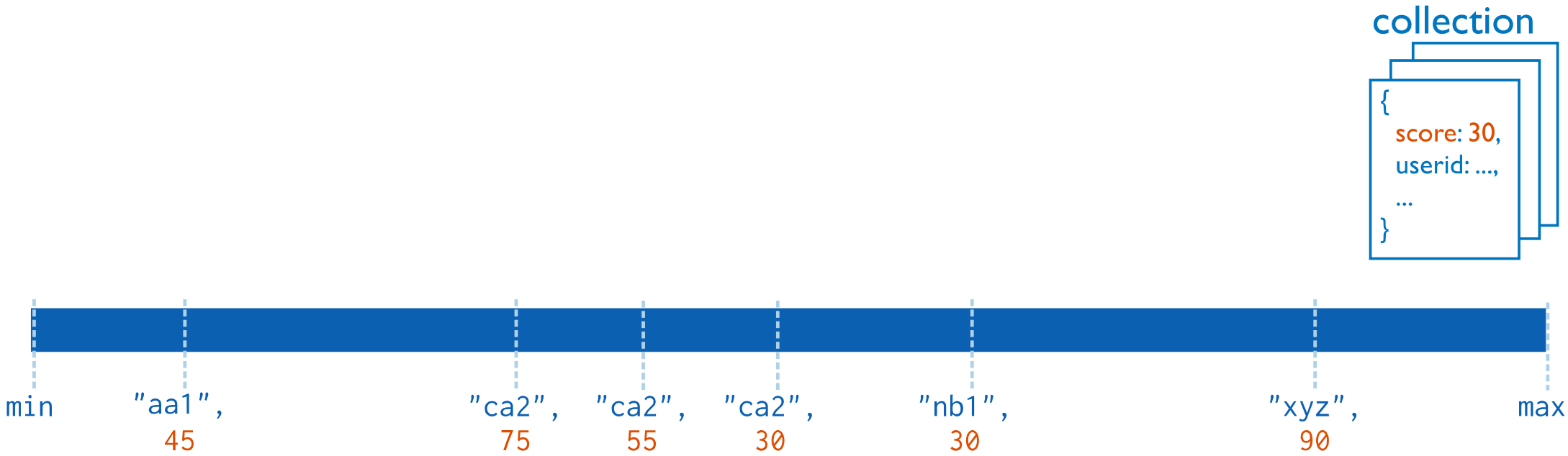

Collection
Query Criteria
Sort order
db.users.find({ score: { "$lt": 30 } }).sort({ score: -1 })
{ score: 1 } Index
min
18
30
45
75
max
{ score: 25, ... }
{ score: 56, ... }
{ score: 45, ... }
{ score: 75, ... }
{ score: 5, ... }
{ score: 40, ... }
{ score: 18, ... }
...
{ score: 30, ... }
users

ÍNDICES

ÍNDICE SIMPLE



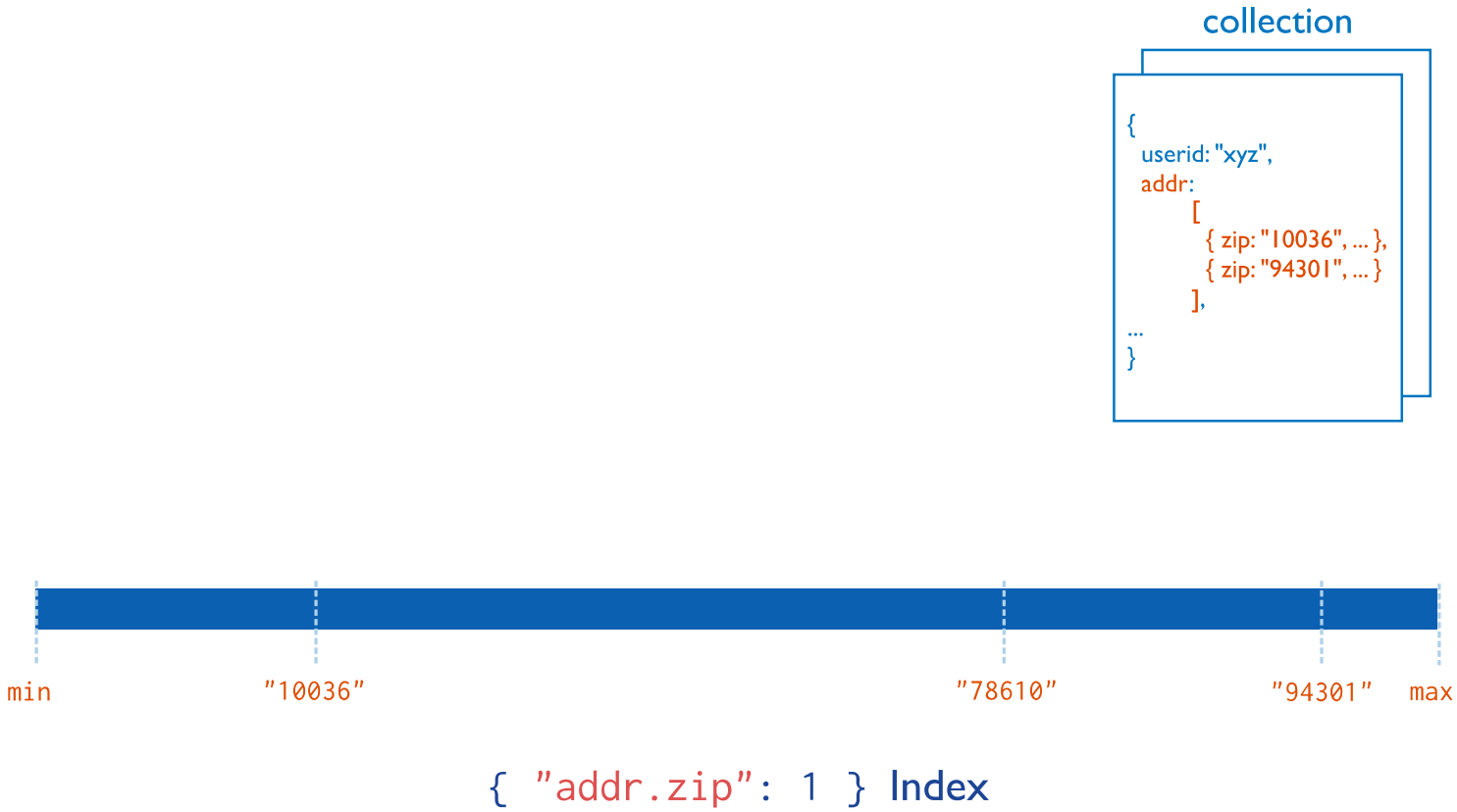
`{ score: 1 }` **Index**

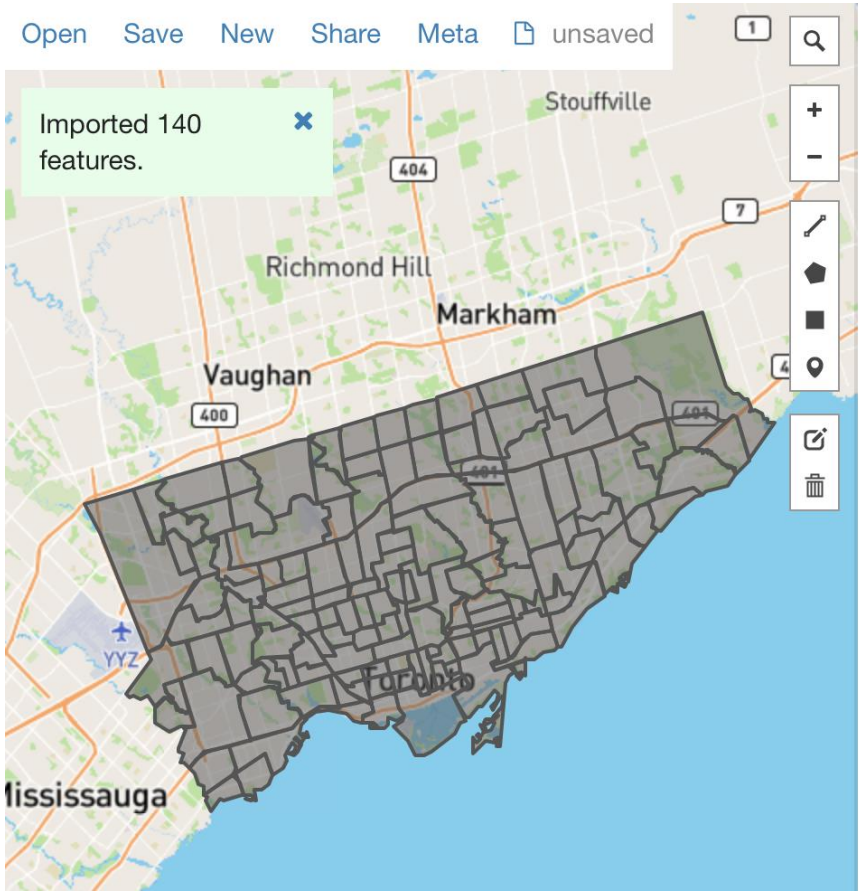


{ userid: 1, score: -1 } **Index**

ÍNDICE COMPUESTO

ÍNDICE MULTIKEY



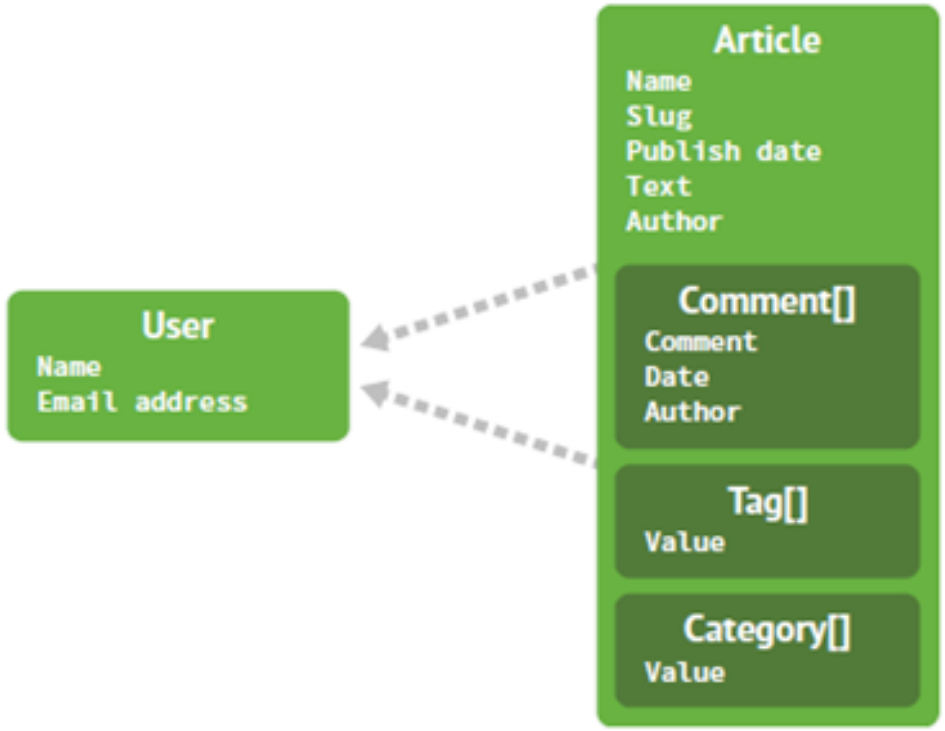



```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [
            [
              -79.4359157087306
              43.6801533947749
            ],
            [
              -79.4359116034227
              43.679547504996
            ],
            [
              -79.4337822305274
              43.6744077713687
```

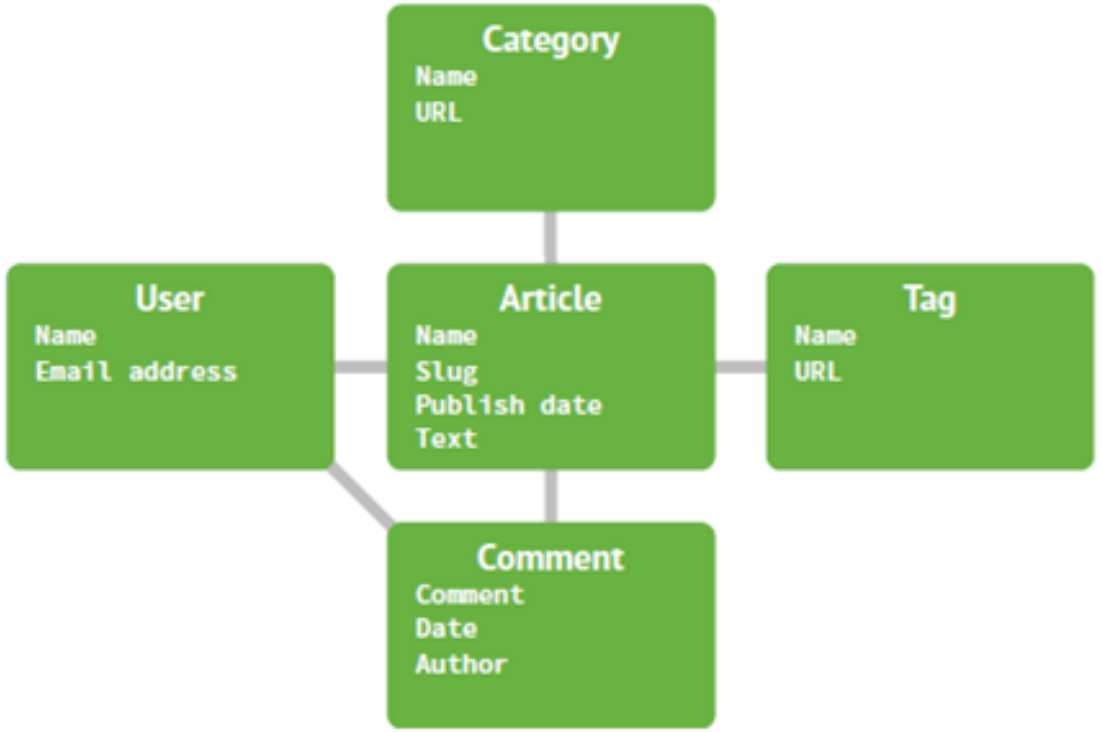

GEOJSON

ÍNDICE GEOESPACIAL

- db.collection.createIndex({ <location field> : "2dsphere" })
- db.collection.createIndex({ <location field> : "2d" })

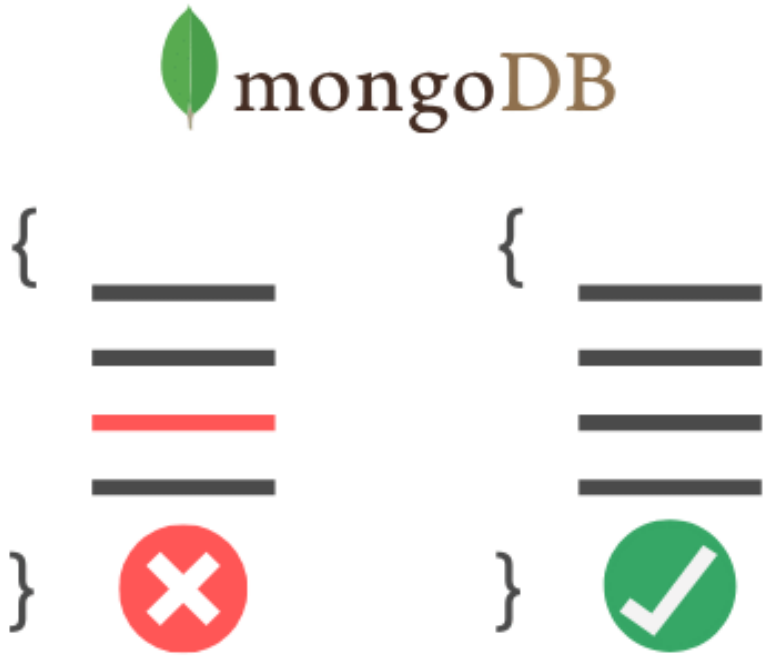


MongoDB Data Model



ScaleBase Data Model

DATA MODEL



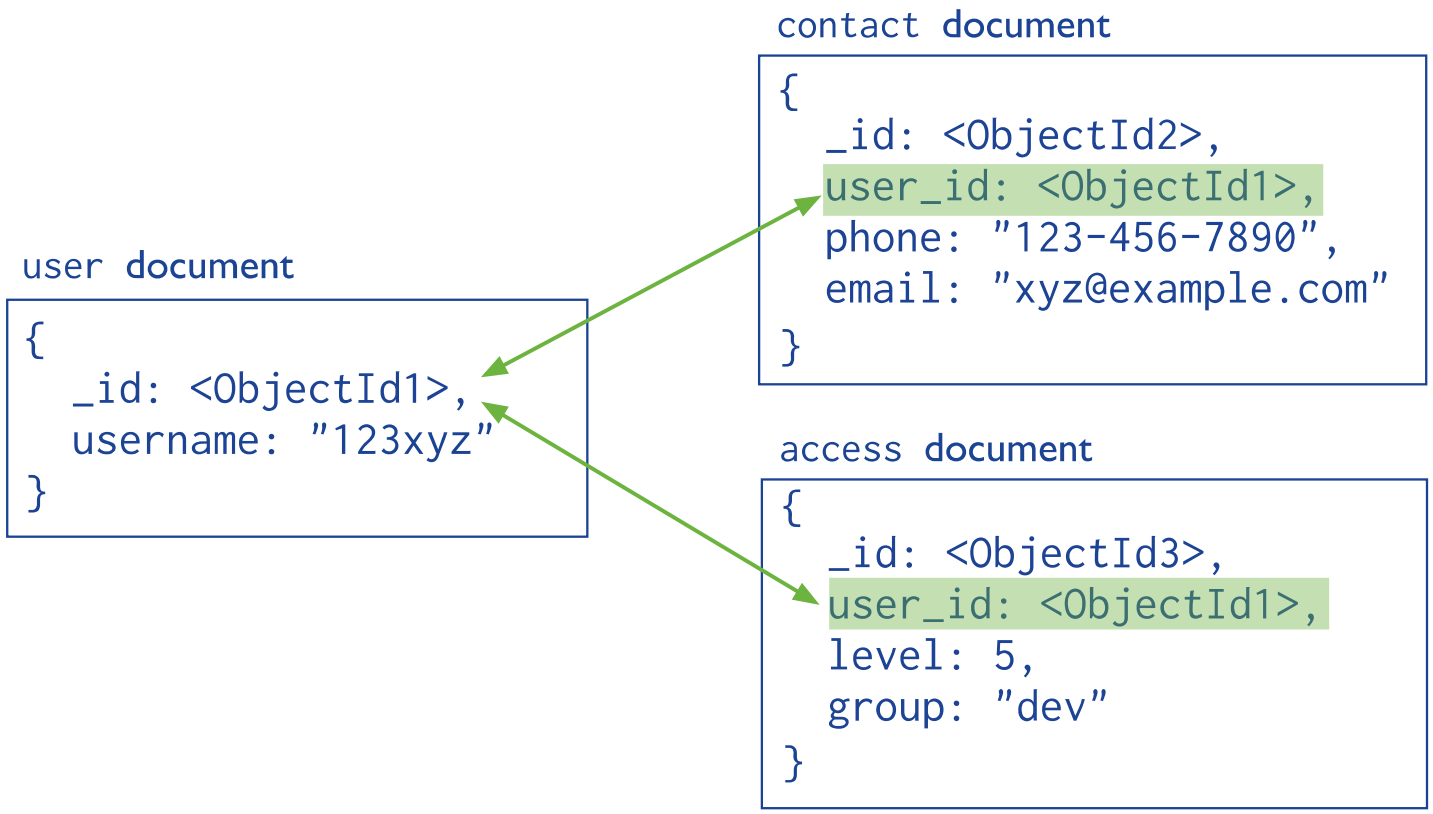
ESQUEMA DE VALIDACIÓN

```
{
  _id: <ObjectId1>,
  username: "123xyz",
  contact: {
              phone: "123-456-7890",
              email: "xyz@example.com"
            },
  access: {
              level: 5,
              group: "dev"
            }
}
```

Embedded sub-document

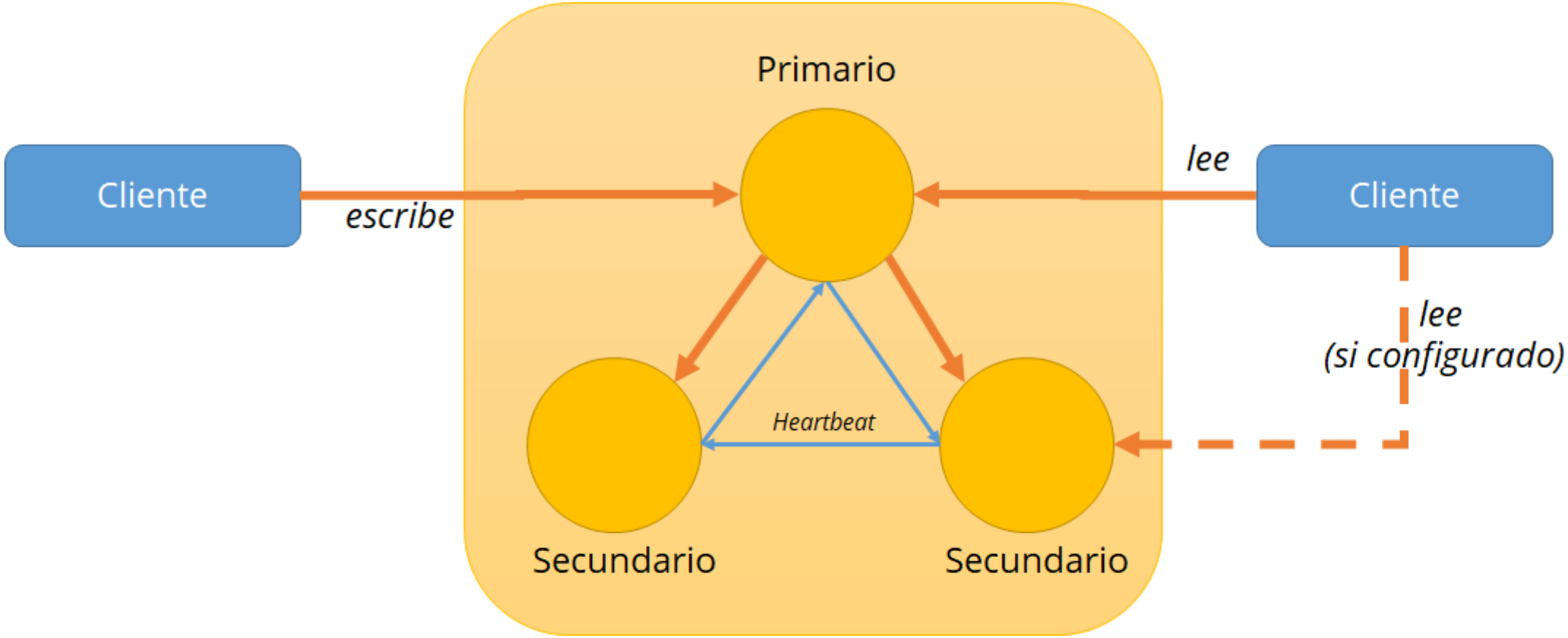
Embedded sub-document

DOCUMENTOS EMBEBIDOS

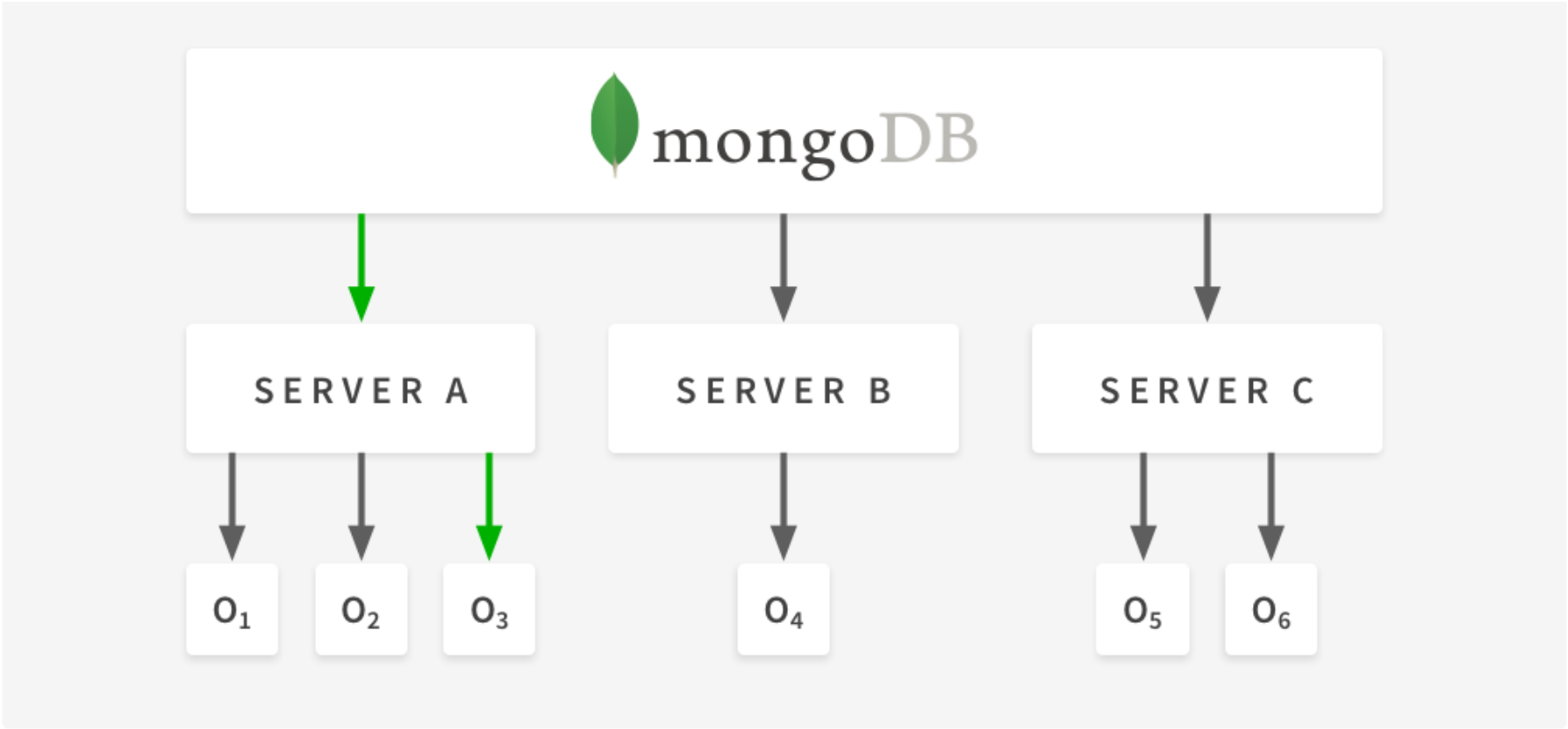


REFERENCIAS

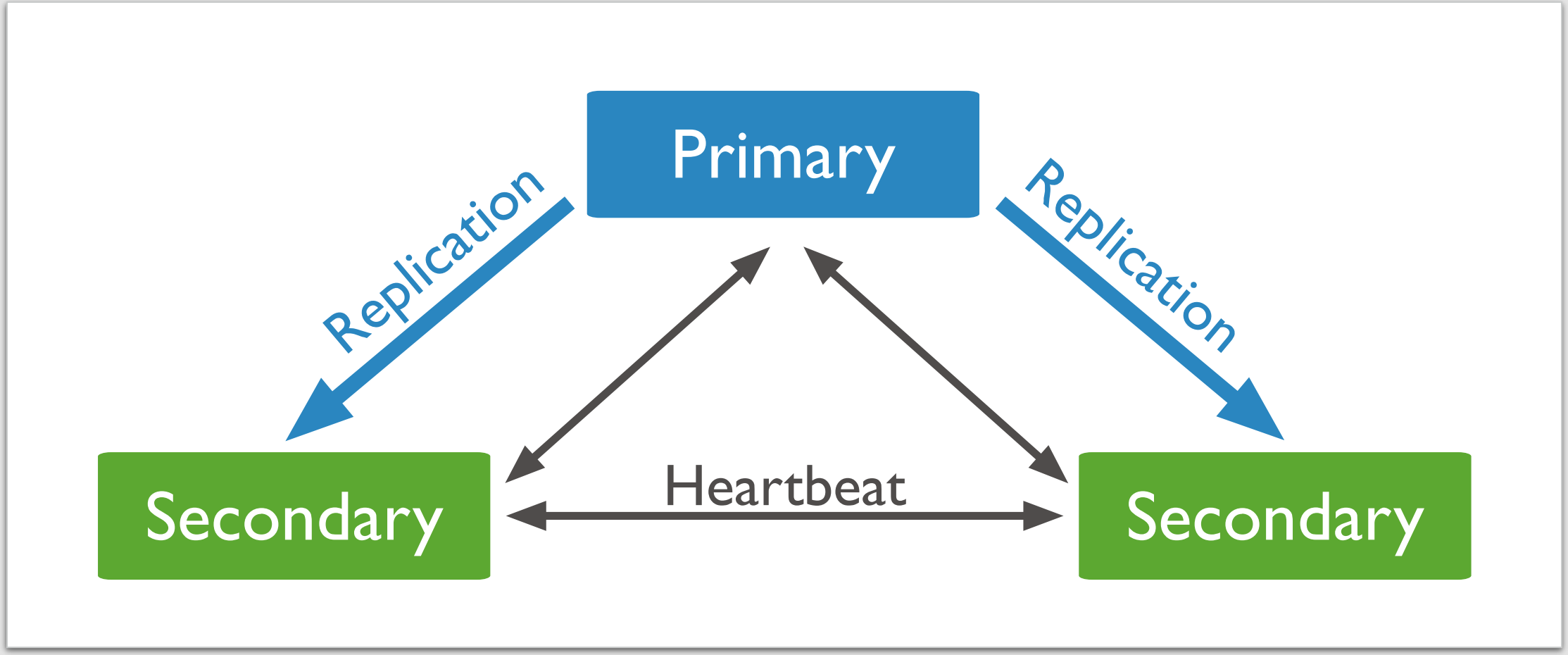
REPLICACIÓN

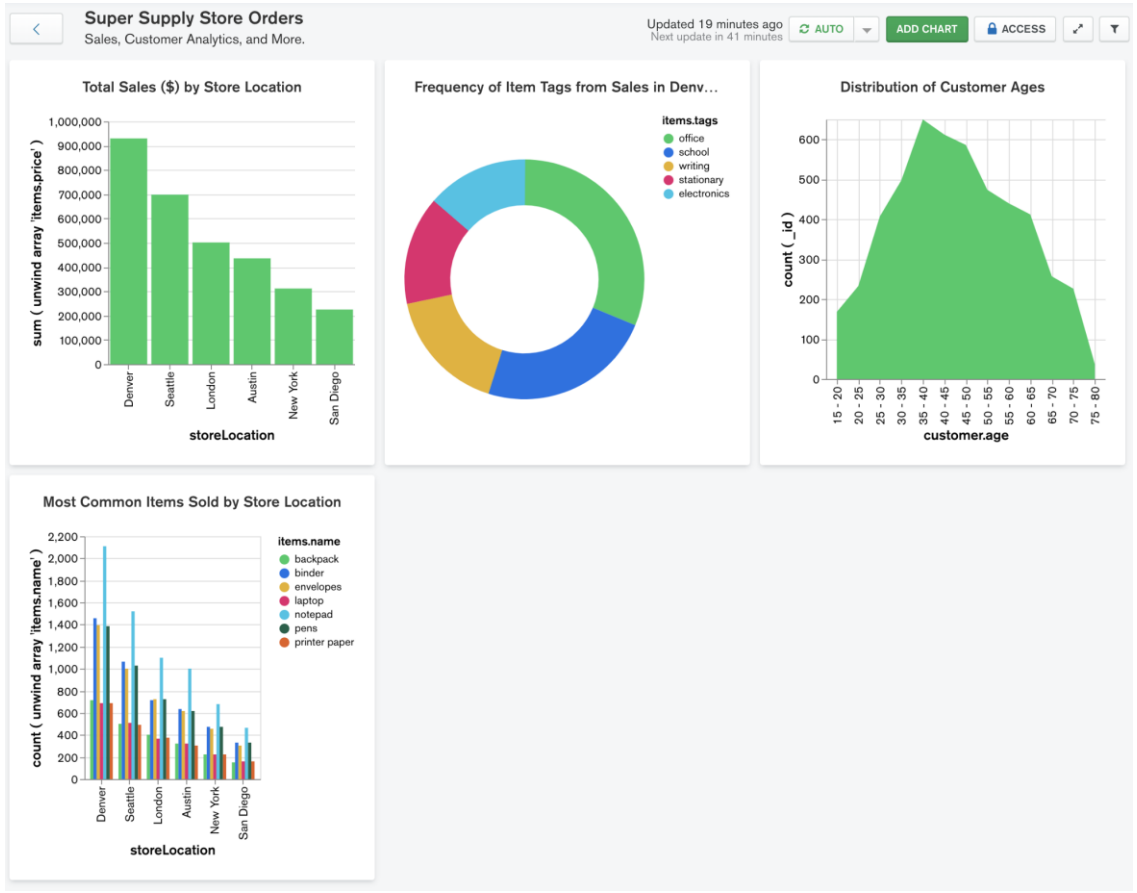


OPLOG



DEMO

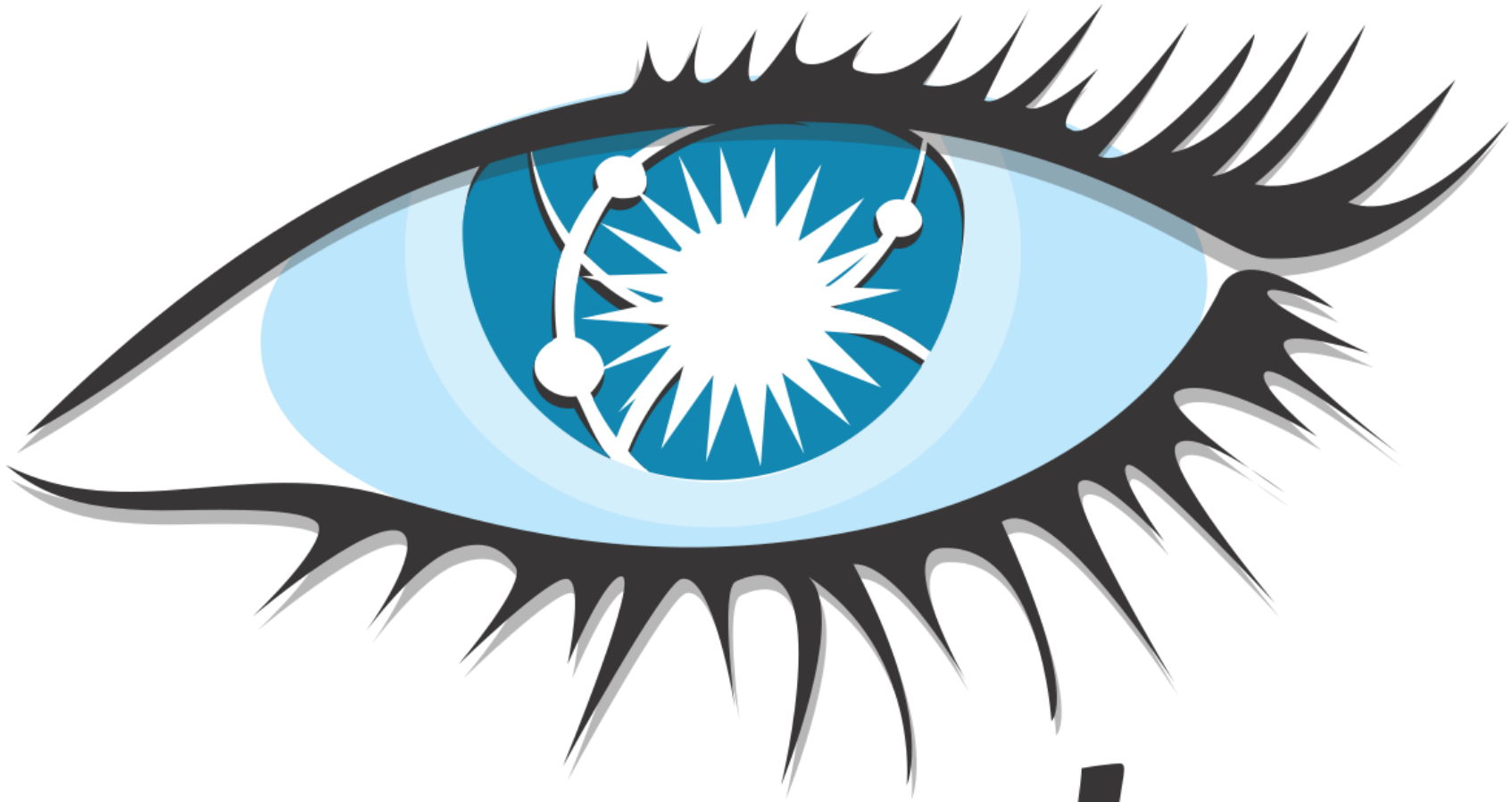

Primary
Replication
Replication
Secondary
Heartbeat
Secondary


Super Supply Store Orders
Sales, Customer Analytics, and More.
Updated 19 minutes ago
Next update in 41 minutes
AUTO
ADD CHART
ACCESS
Total Sales ($) by Store Location
sum (unwind array 'items.price')
1,000,000
900,000
800,000
700,000
600,000
500,000
400,000
300,000
200,000
100,000
0
Denver
Seattle
London
Austin
New York
San Diego
storeLocation
Frequency of Item Tags from Sales in Denv...
items.tags
office
school
writing
stationary
electronics
Distribution of Customer Ages
count (_id)
600
500
400
300
200
100
0
15 - 20
20 - 25
25 - 30
30 - 35
35 - 40
40 - 45
45 - 50
50 - 55
55 - 60
60 - 65
65 - 70
70 - 75
75 - 80
customer.age
Most Common Items Sold by Store Location
count (unwind array 'items.name')
2,200
2,000
1,800
1,600
1,400
1,200
1,000
800
600
400
200
0
Denver
Seattle
London
Austin
New York
San Diego
storeLocation
items.name
backpack
binder
envelopes
laptop
notepad
pens
printer paper

CHARTS

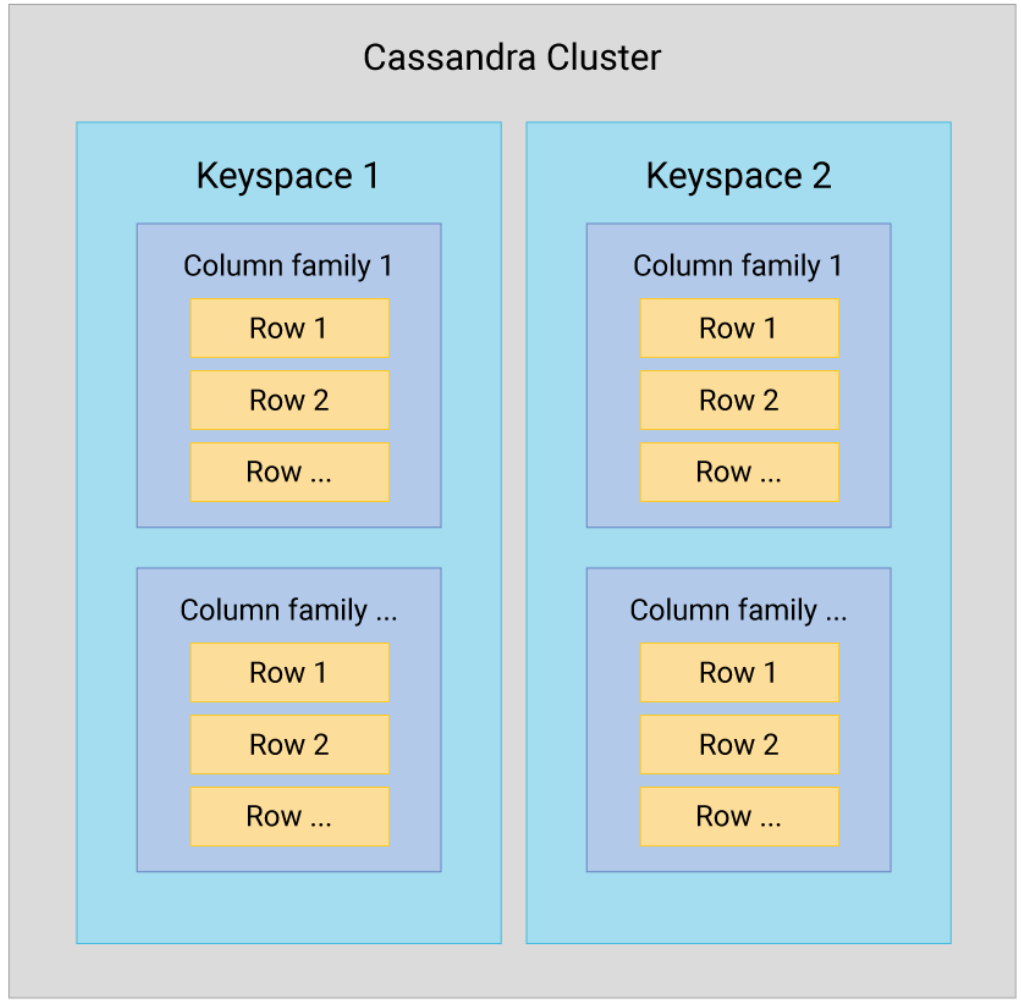
EJERCICIO

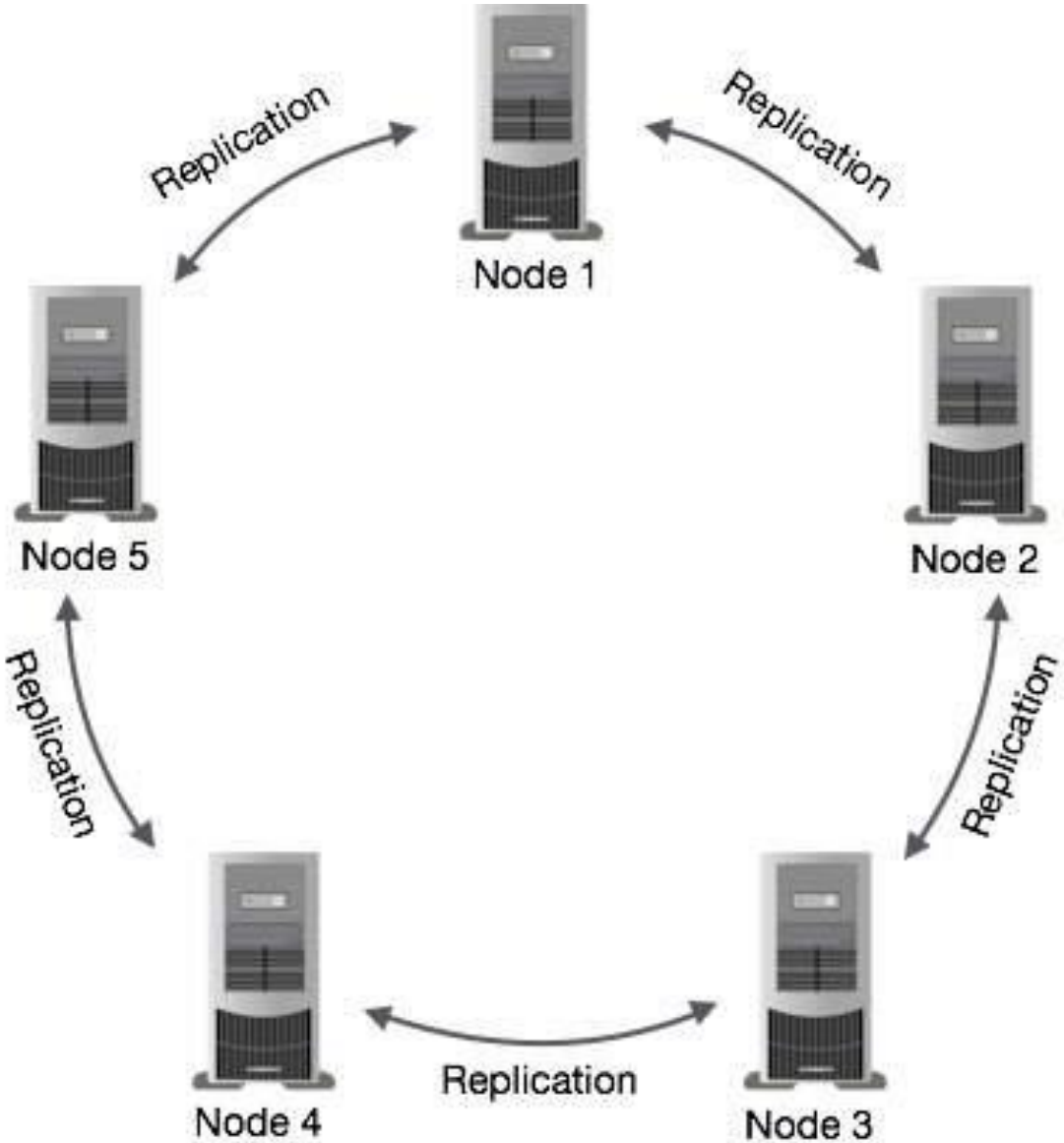
https://www.mongodb.com/docs/charts/tutorial/movie-details/movie-details-tutorial-overview/

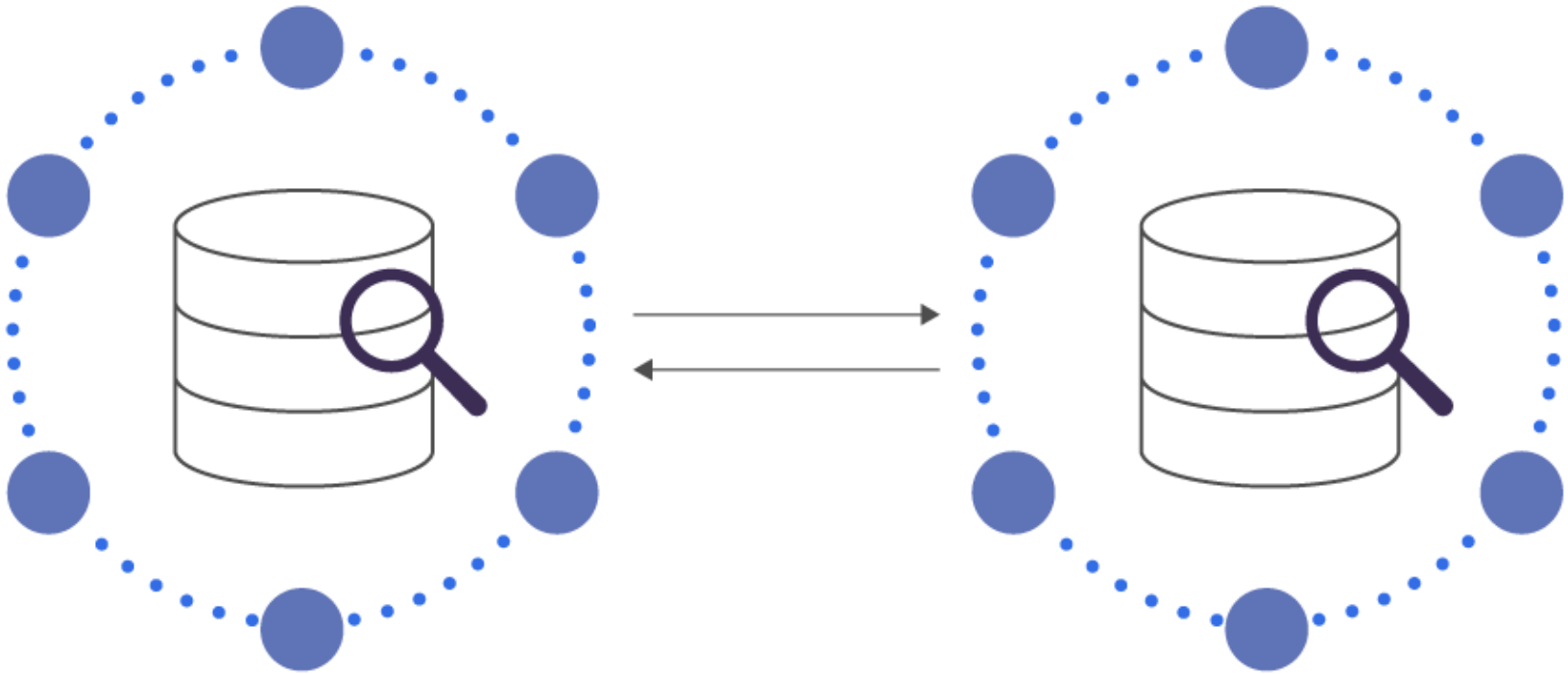


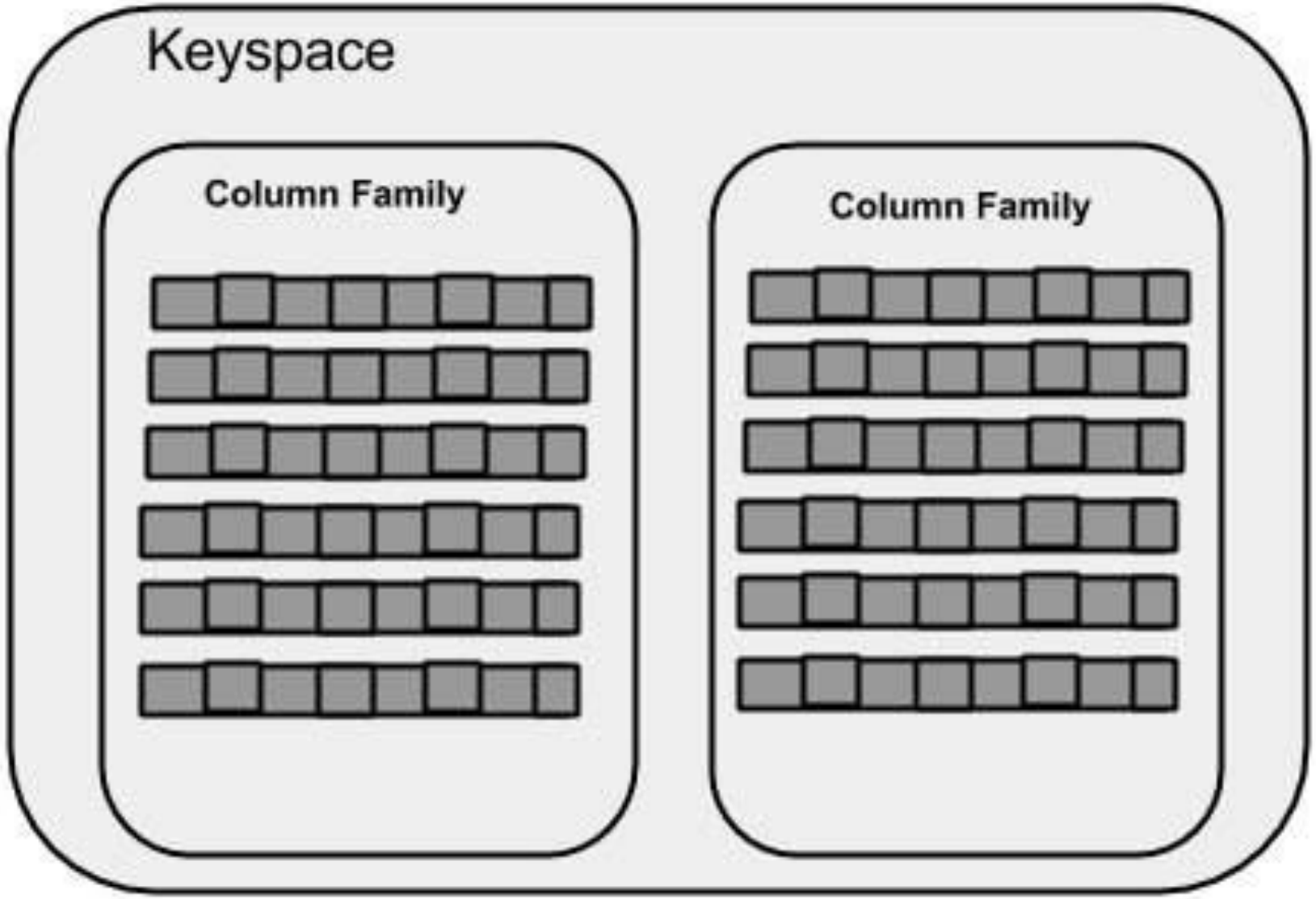
cassandra

Cassandra Data Model




Node 1
Node 2
Node 3
Node 4
Node 5
Replication
Replication
Replication
Replication
Replication

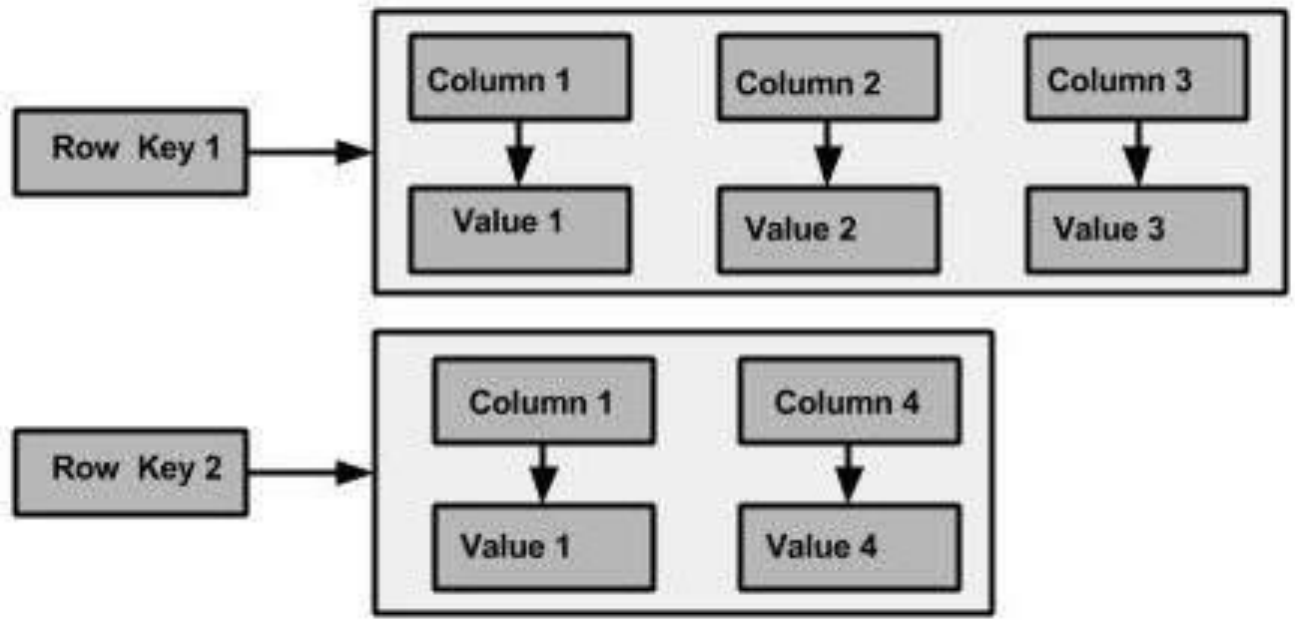



Keyspace
Column Family
Column Family


QWIKLABS

Column		
name : byte[]	value : byte[]	clock : clock[]

Super Column	
name : byte[]	cols : map<byte[], column>



CREATE KEYSPACE <identifier> WITH <properties>

CREATE TABLE tablename(column1 name datatype PRIMARYKEY, column2 name data type, column3 name data type, **PRIMARY KEY (column1)**)

ALTER TABLE table name ADD new column datatype;

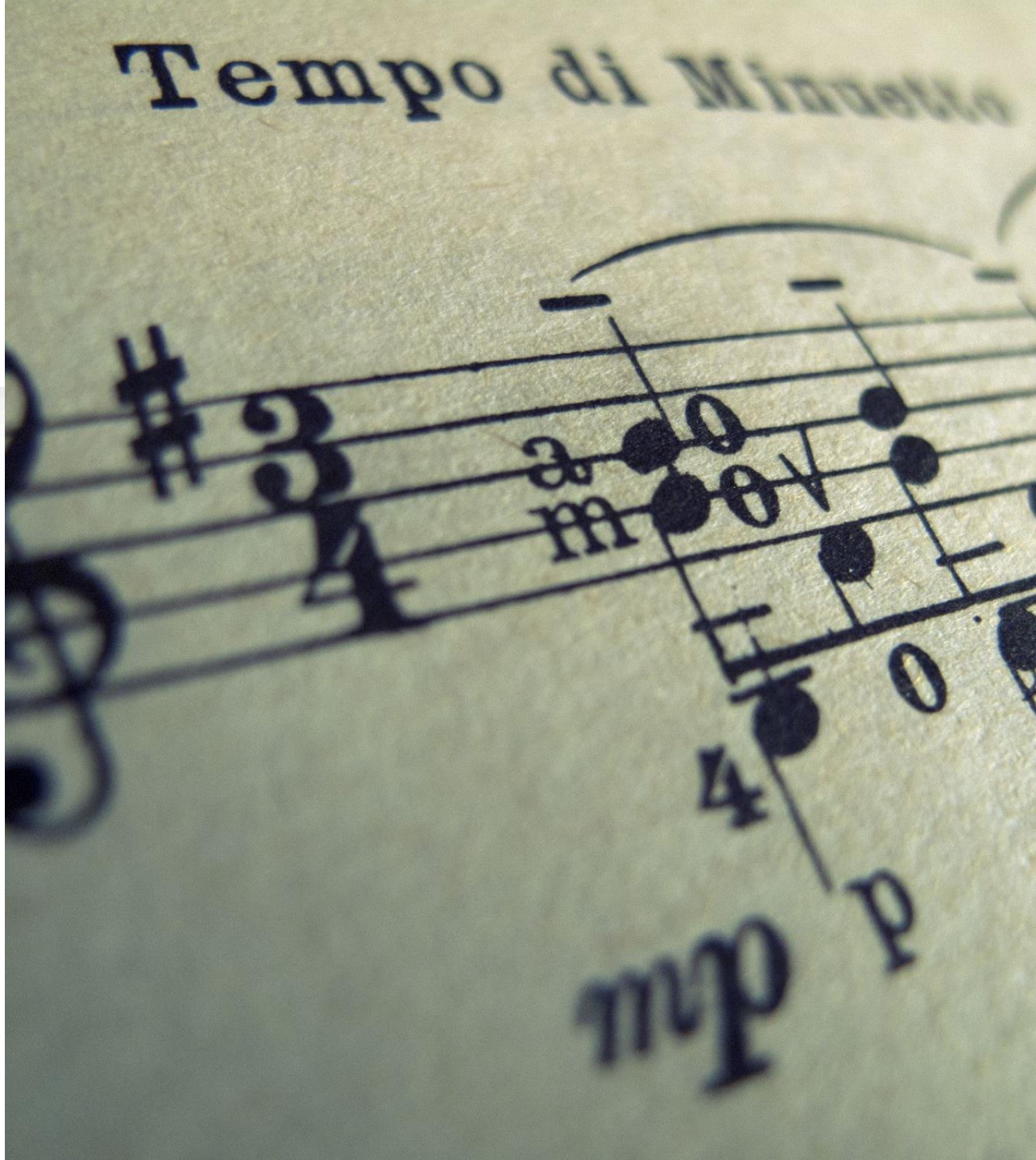
ALTER table name

DROP column name;

DROP TABLE <tablename>

INSERT INTO <tablename> (<column1 name>, <column2 name>....) VALUES (<value1>, <value2>....) USING <option>

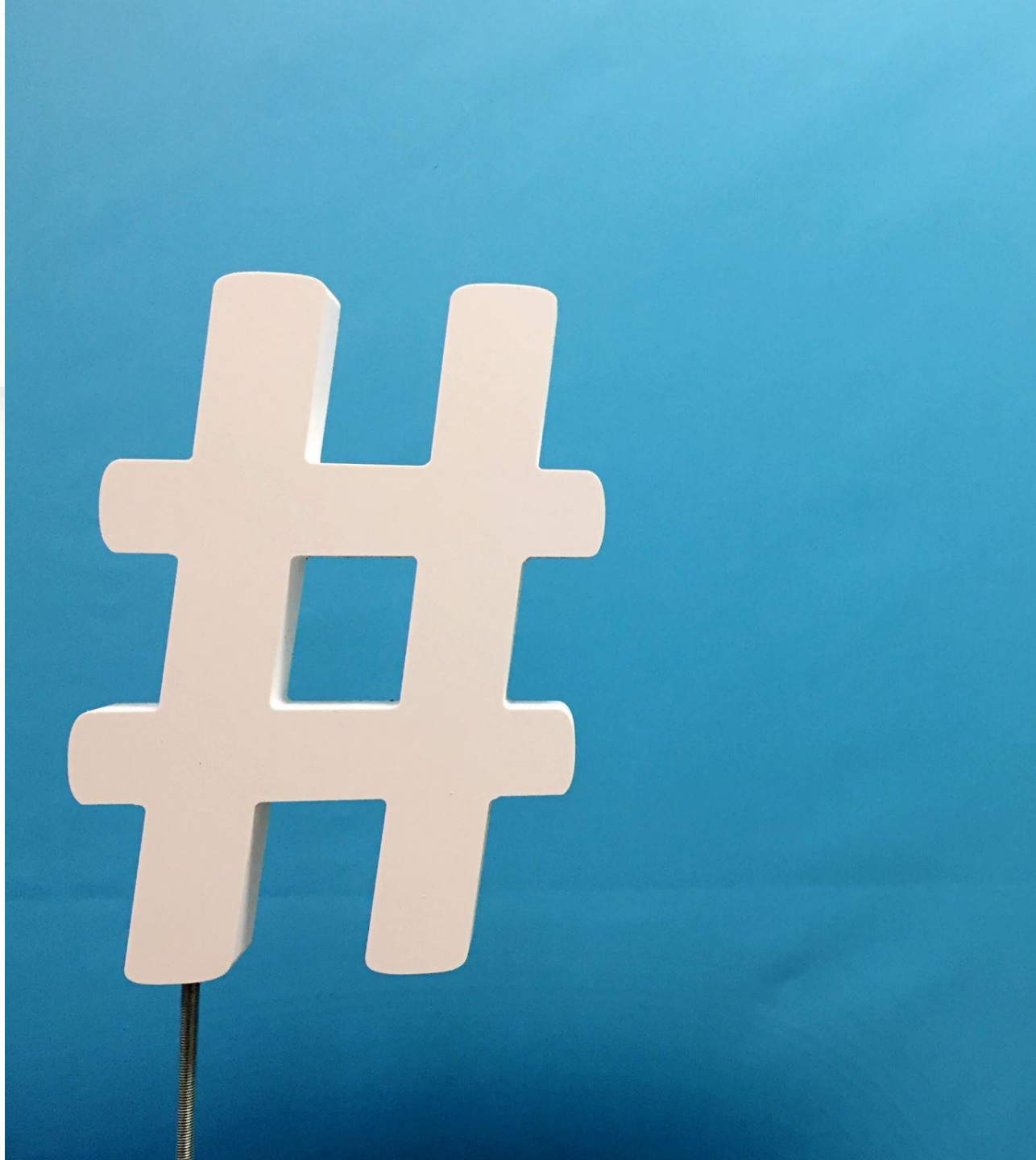
UPDATE <tablename> SET <column name> = <new value> <column name> = <value>.... WHERE <condition>



SELECT FROM <tablename>

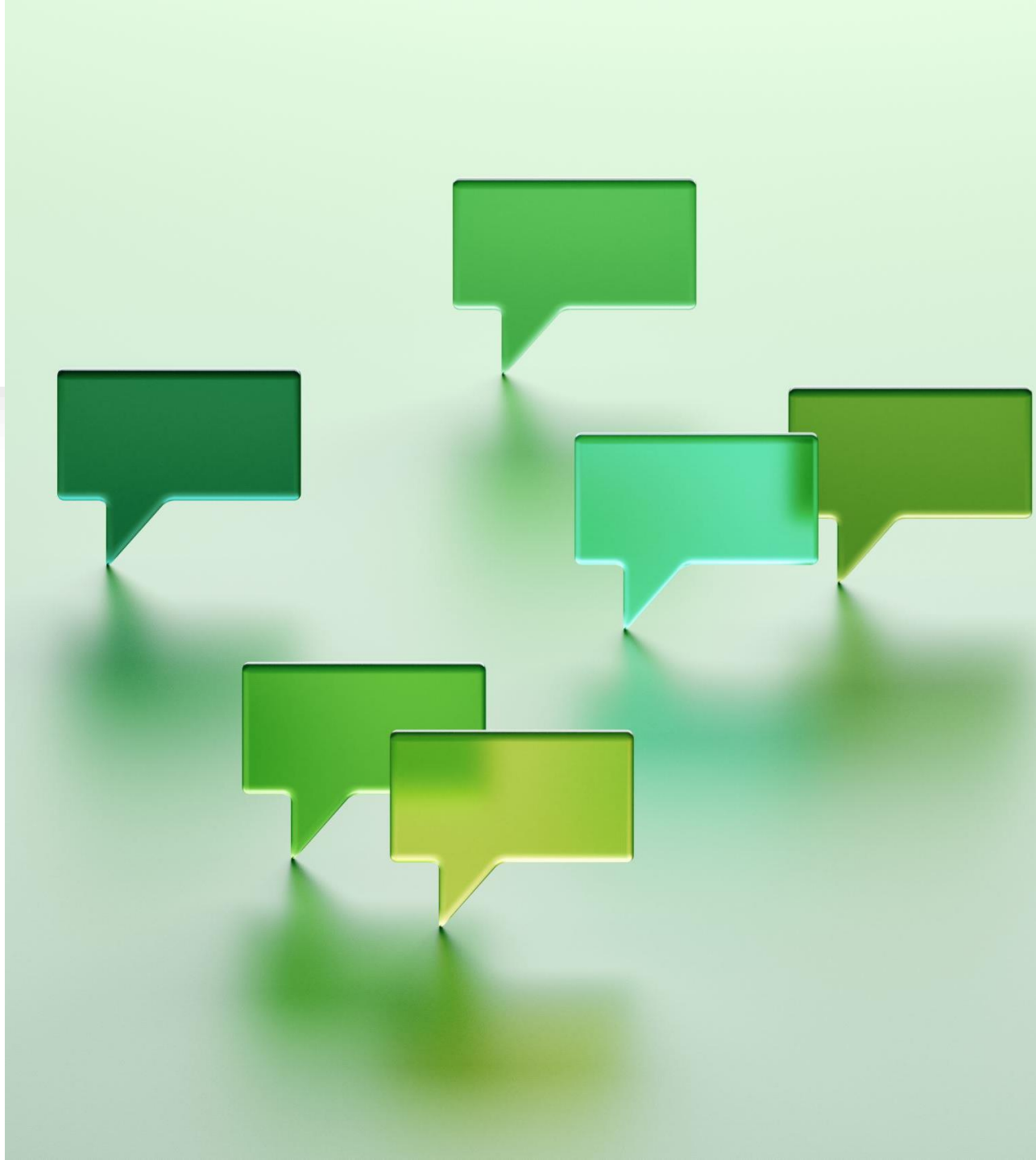
SELECT FROM <table name> WHERE <condition>;

- DELETE FROM <identifier> WHERE <condition>;

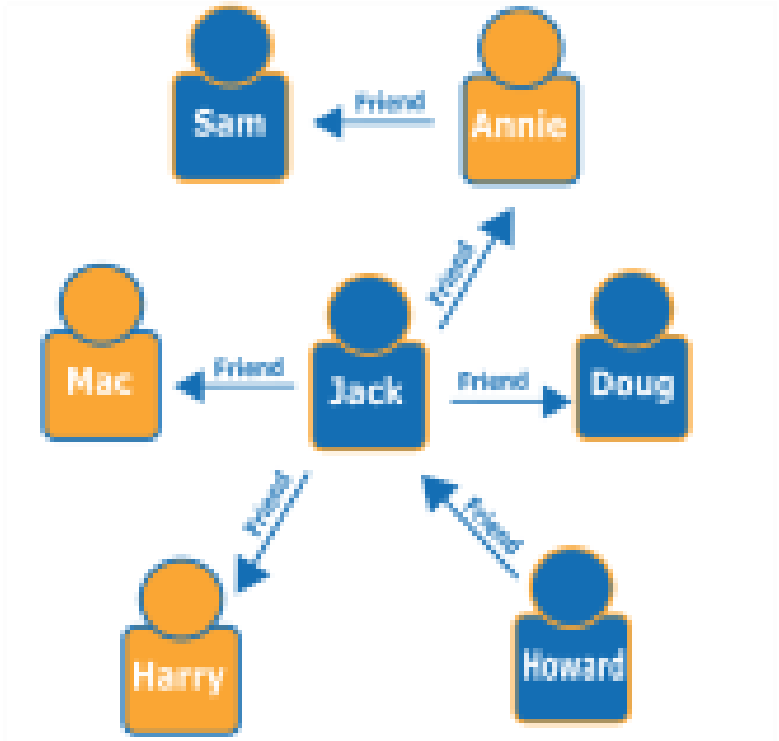


CREATE TABLE data(name text PRIMARY KEY, email list<text>);

CREATE TABLE data2 (name text PRIMARY KEY, phone set<varint>);




C*
Amazon
Keyspaces
for Apache Cassandra


Sam
Friend
Annie
Mac
Friend
Jack
Friend
Friend
Doug
Friend
Harry
Friend
Howard

RDBMS Vs Graph Database

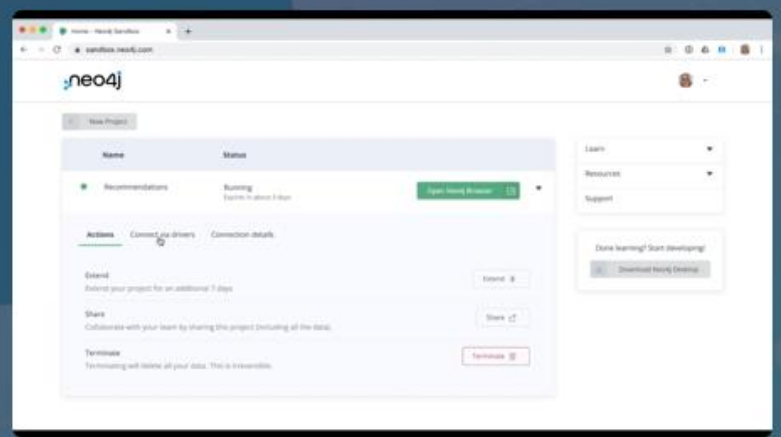
Following is the table which compares Relational databases and Graph databases.

Sr.No	RDBMS	Graph Database
1	Tables	Graphs
2	Rows	Nodes
3	Columns and Data	Properties and its values
4	Constraints	Relationships
5	Joins	Traversal

Comience a utilizar Neo4j Sandbox mientras su café aún se está preparando

Neo4j es una base de datos gráfica nativa, diseñada específicamente para aprovechar las relaciones de datos y permitir aplicaciones más ricas e inteligentes.

Lanzamiento de la zona de pruebas gratuita



Comience con los fundamentos

¿Eres principiante o estás empezando? Crea tus bases de Neo4j con estos cursos seleccionados.

Fundamentos de Neo4j

Aprenda los conceptos básicos de Neo4j y el modelo de gráfico de propiedades

1 hora →

Fundamentos de Cypher

Aprenda Cypher en 1 hora

1 hora →

Fundamentos del modelado de datos gráficos

Aprenda a diseñar un gráfico Neo4j utilizando las mejores prácticas

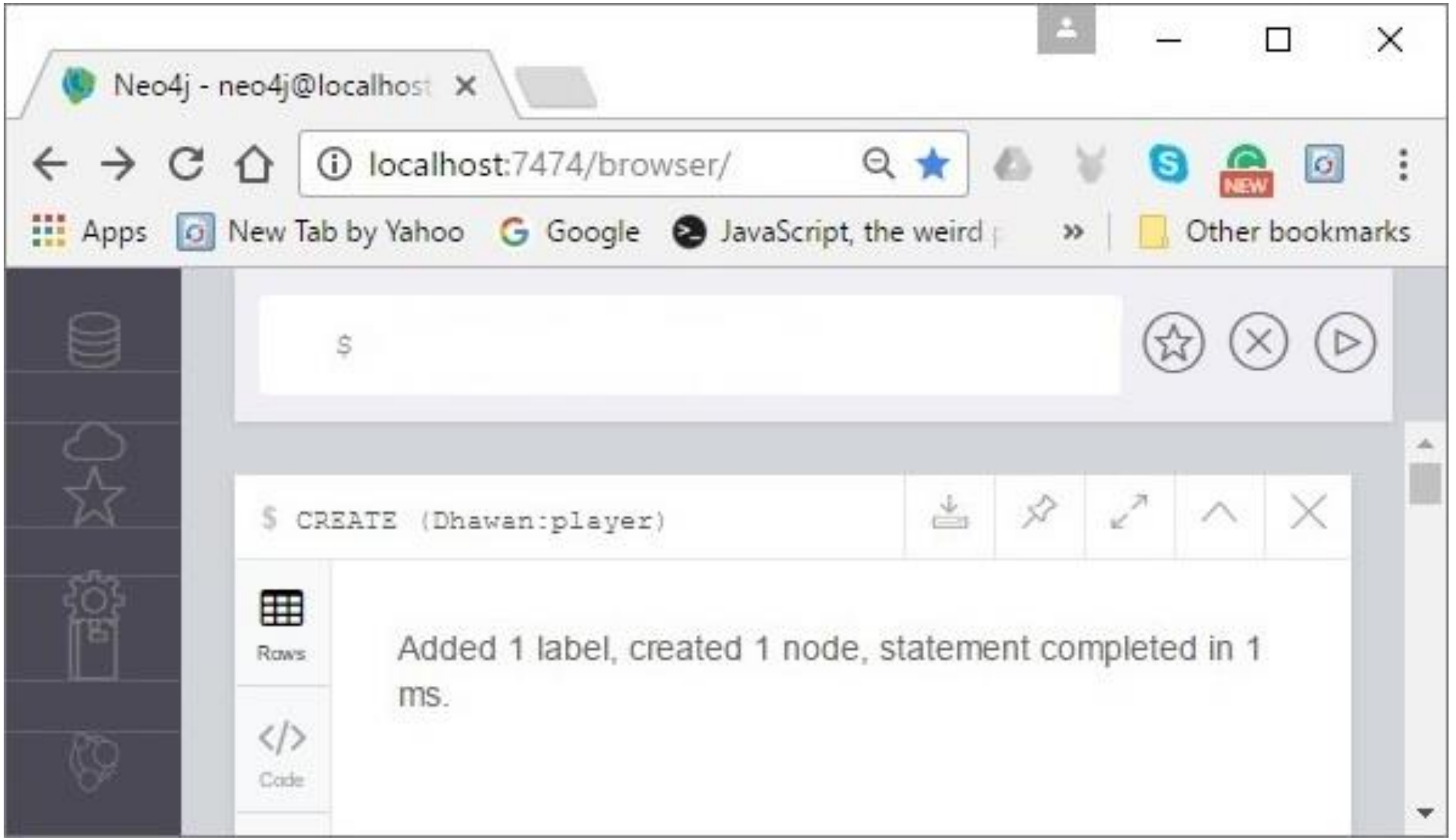
2 horas →

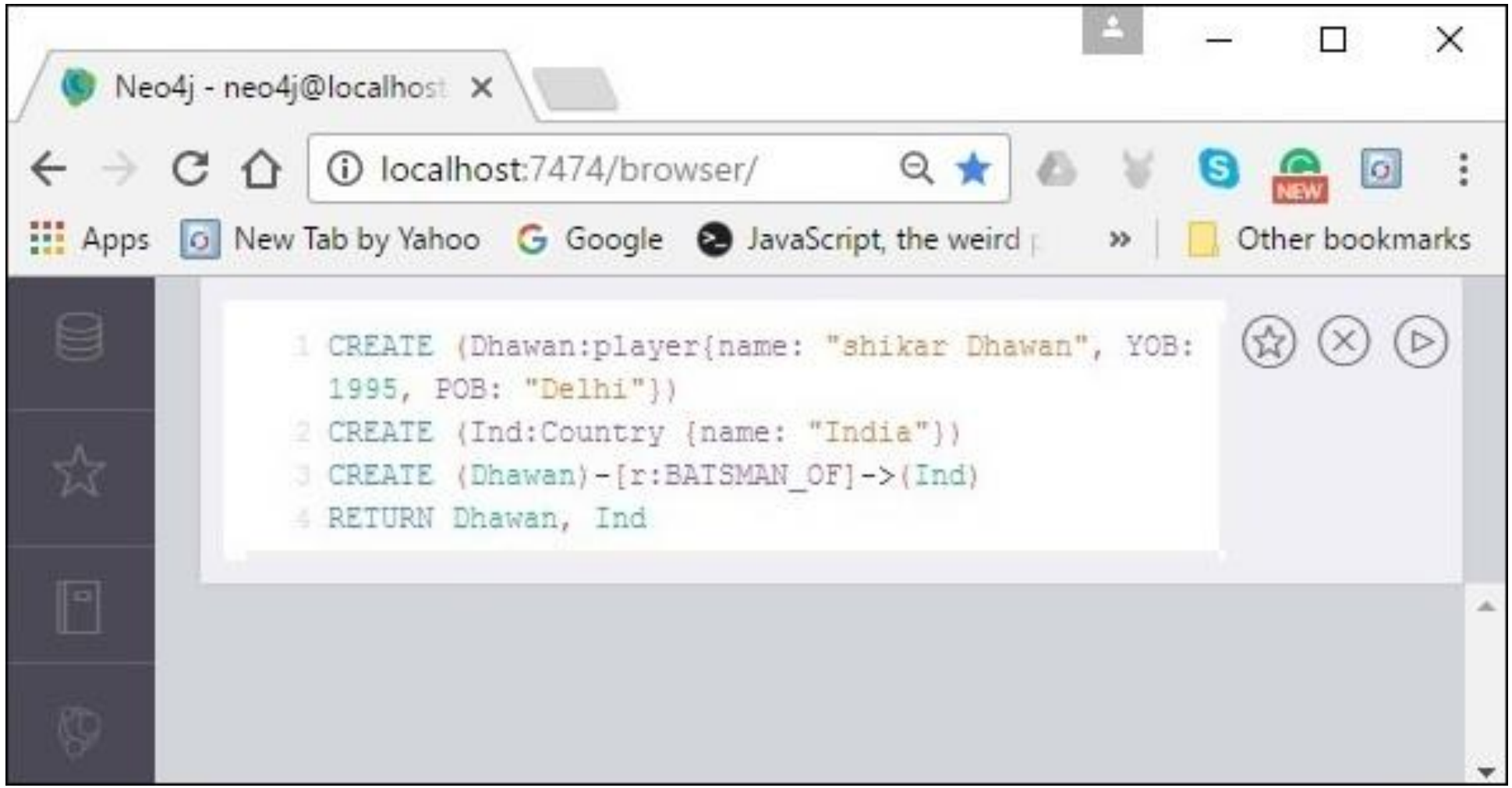
Fundamentos de la importación de datos

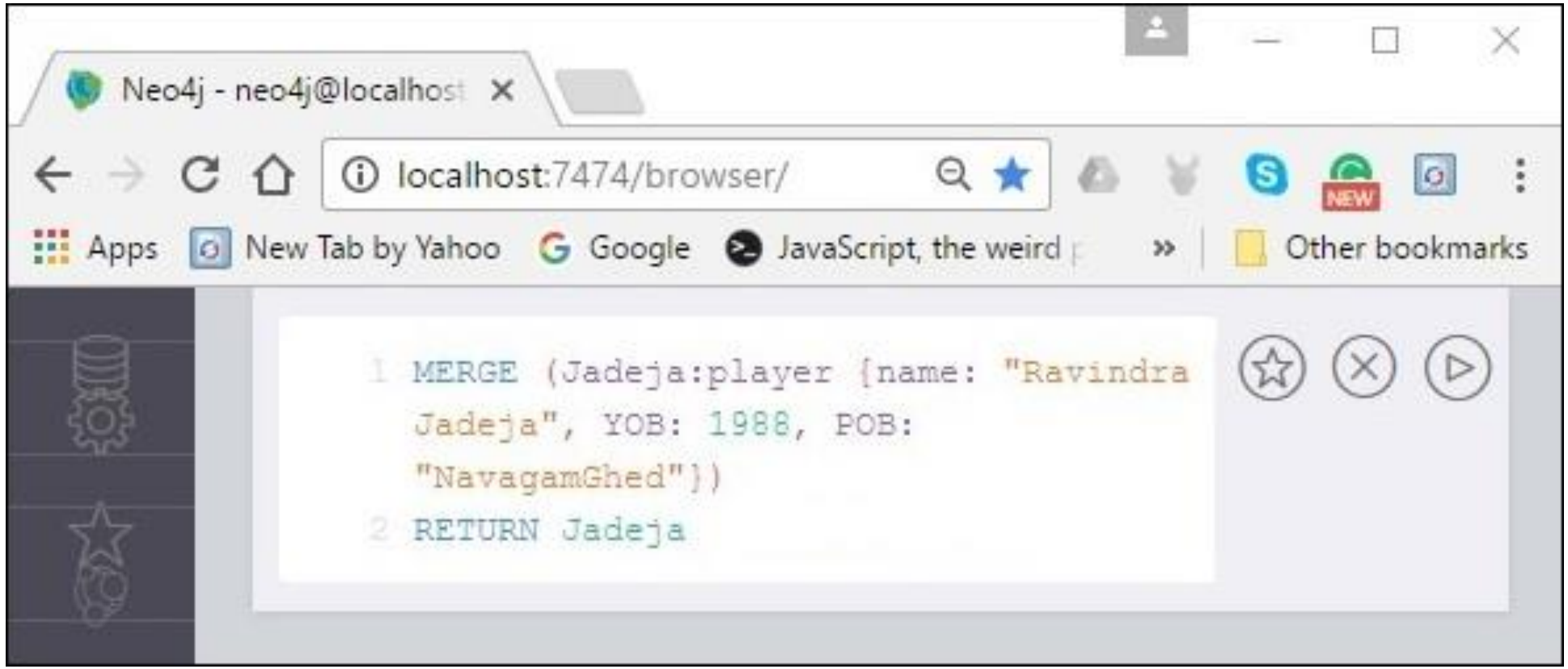
Aprenda a importar datos a Neo4j

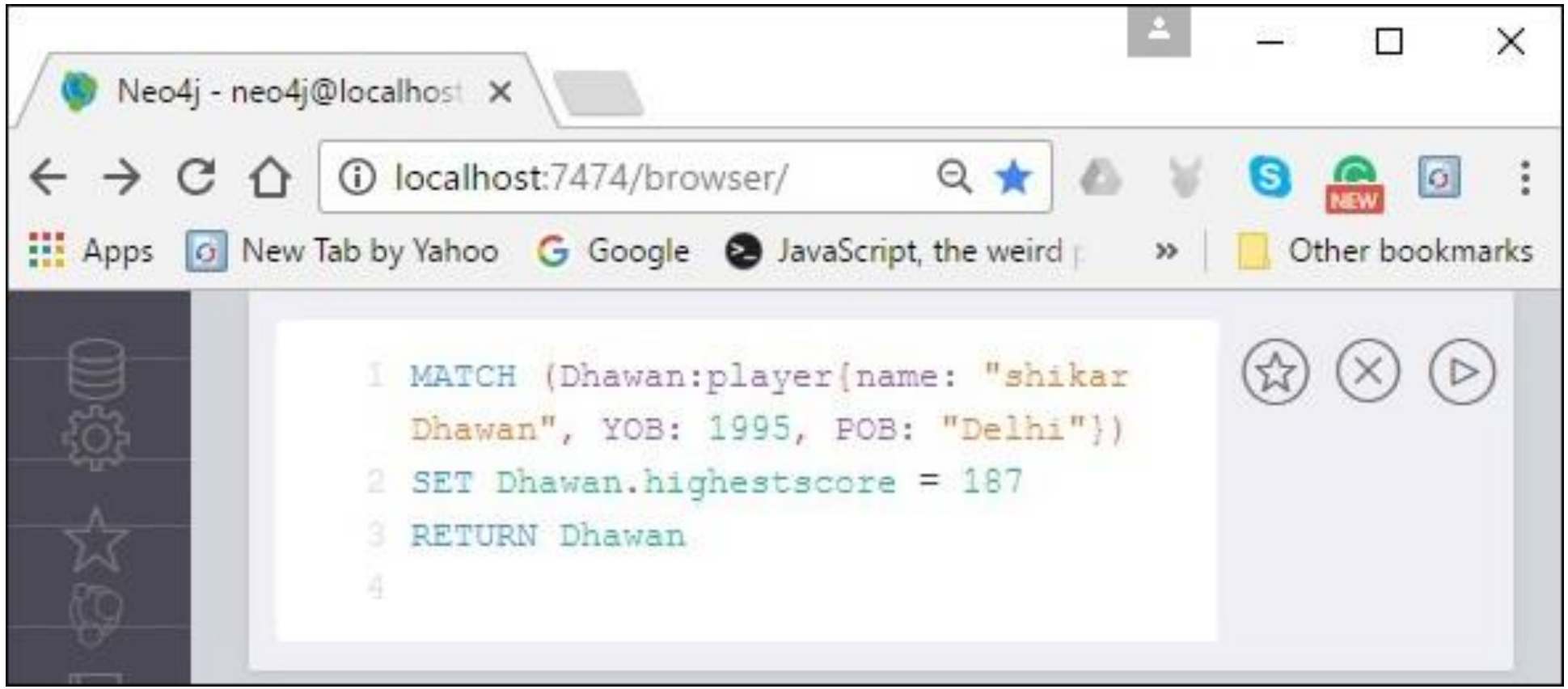
2 horas →

Sr.No	Write Clause	Usage
1	CREATE	This clause is used to create nodes, relationships, and properties.
2	MERGE	This clause verifies whether the specified pattern exists in the graph. If not, it creates the pattern.
3	SET	This clause is used to update labels on nodes, properties on nodes and relationships.
4	DELETE	This clause is used to delete nodes and relationships or paths etc. from the graph.
5	REMOVE	This clause is used to remove properties and elements from nodes and relationships.
6	FOREACH	This class is used to update the data within a list.
7	CREATE UNIQUE	Using the clauses CREATE and MATCH, you can get a unique pattern by matching the existing pattern and creating the missing one.
8	Importing CSV files with Cypher	Using Load CSV you can import data from .csv files.

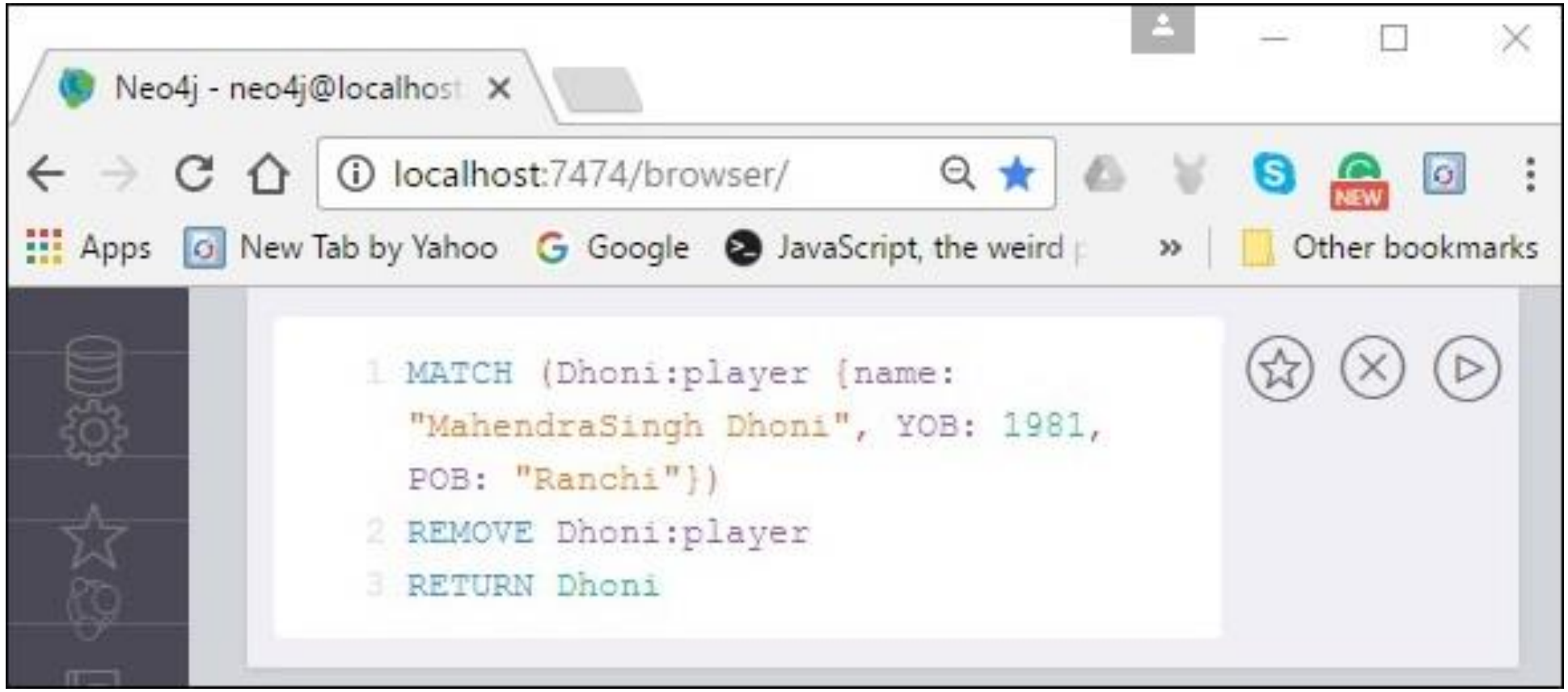

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
$
$ CREATE (Dhawan:player)
Rows
Code
Added 1 label, created 1 node, statement completed in 1 ms.

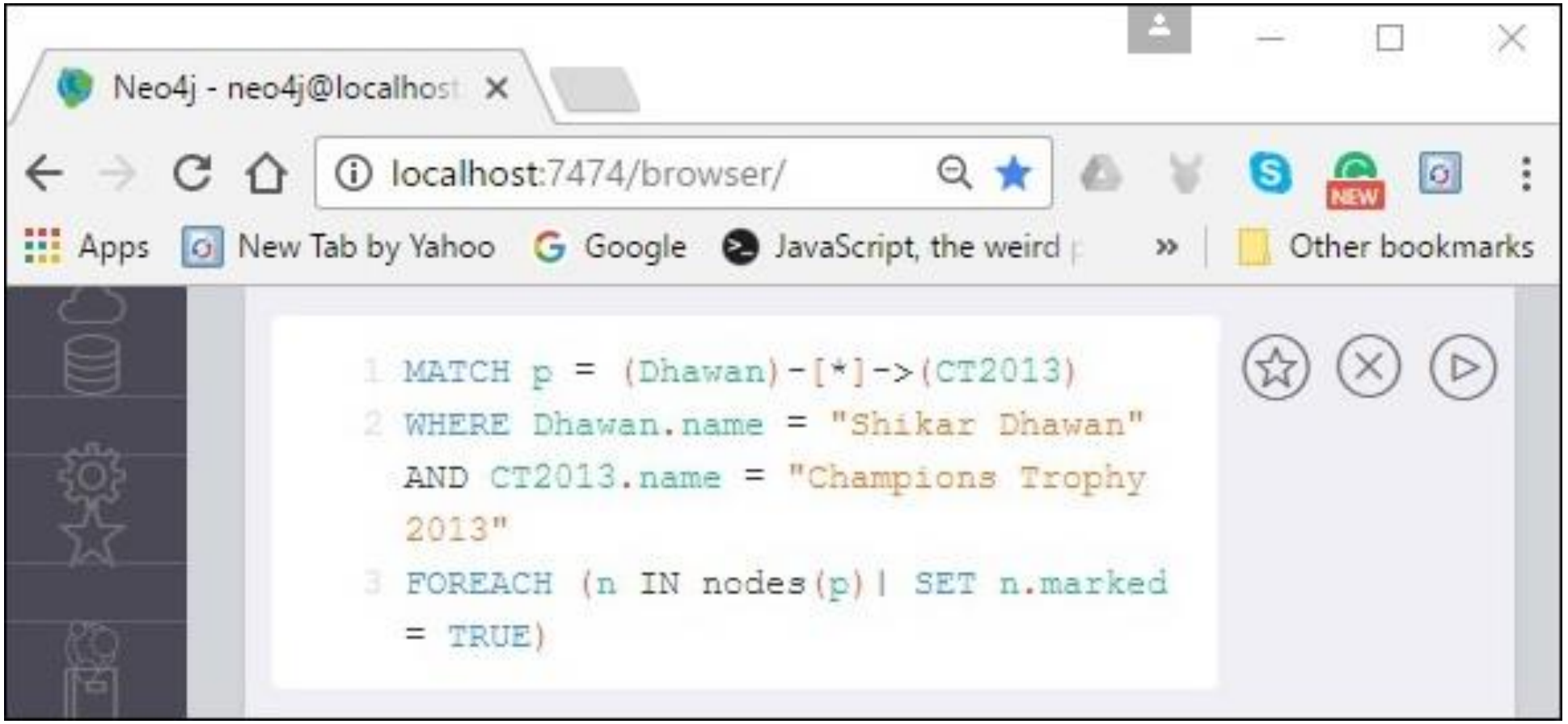

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 CREATE (Dhawan:player{name: "shikar Dhawan", YOB:
1995, POB: "Delhi"})
2 CREATE (Ind:Country {name: "India"})
3 CREATE (Dhawan)-[r:BATSMAN_OF]->(Ind)
4 RETURN Dhawan, Ind

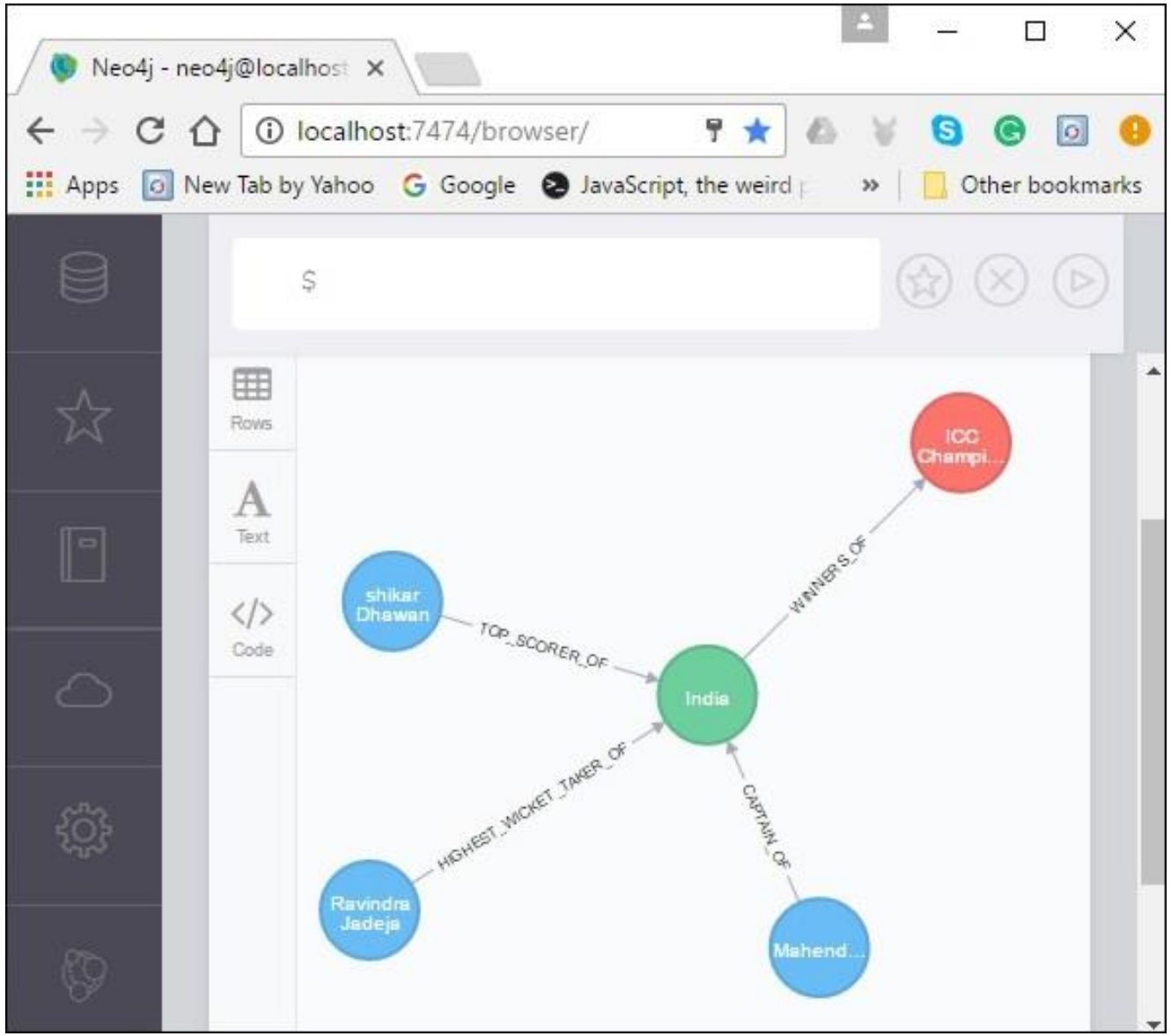

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
MERGE (Jadeja:player {name: "Ravindra Jadeja", YOB: 1988, POB: "NavagamGhed"})
RETURN Jadeja

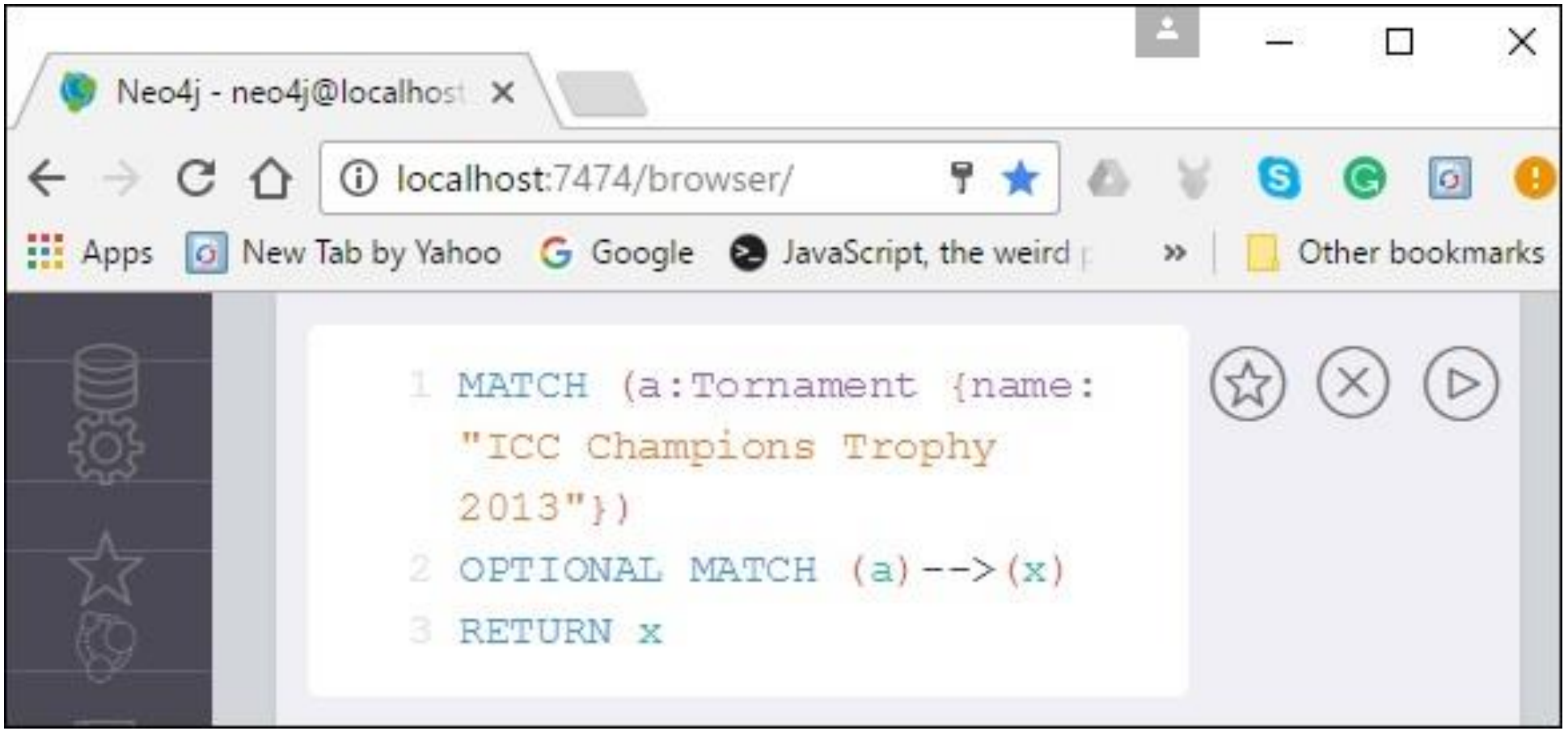

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 MATCH (Dhawan:player{name: "shikar
Dhawan", YOB: 1995, POB: "Delhi"})
2 SET Dhawan.highestscore = 187
3 RETURN Dhawan
4


Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 MATCH (Ishant:player {name: "Ishant Sharma", YOB: 1988, POB: "Delhi"})
2 DETACH DELETE Ishant


Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 MATCH (Dhoni:player {name:
"MahendraSingh Dhoni", YOB: 1981,
POB: "Ranchi"})
2 REMOVE Dhoni:player
3 RETURN Dhoni

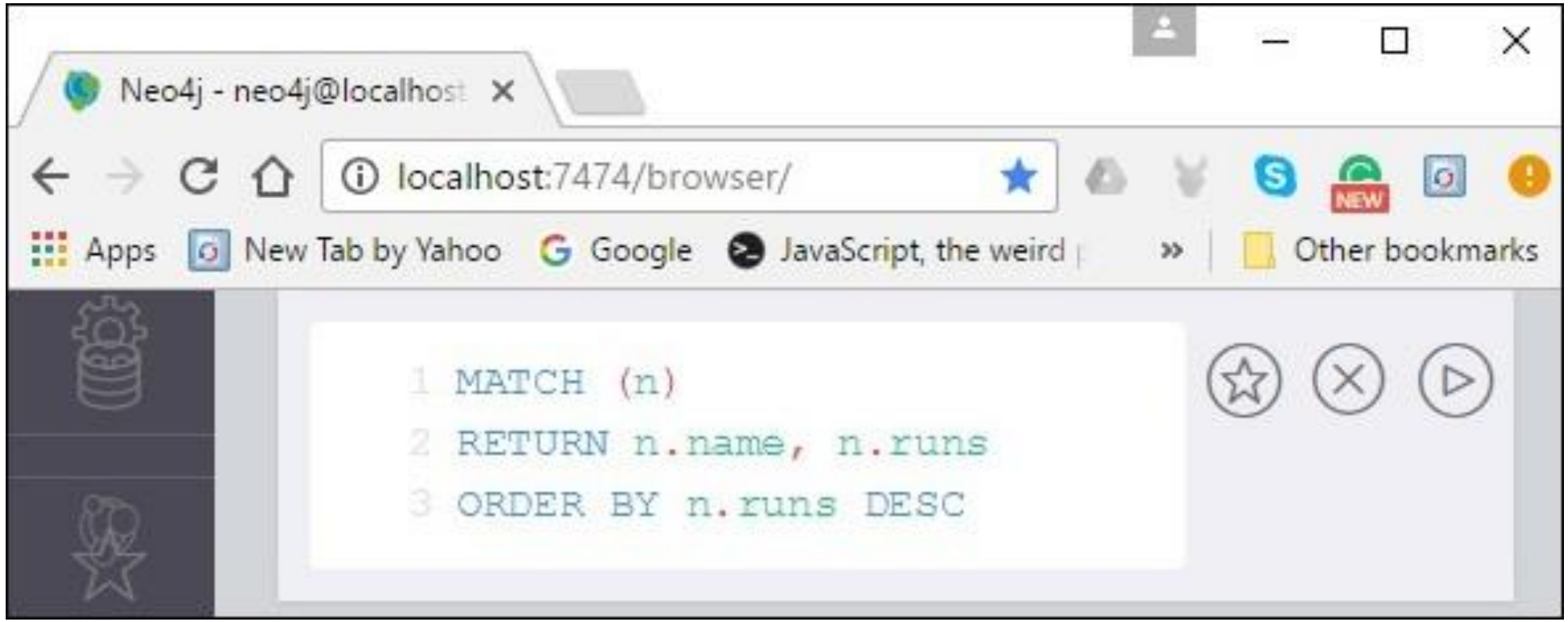

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 MATCH p = (Dhawan)-[*]->(CT2013)
2 WHERE Dhawan.name = "Shikar Dhawan"
AND CT2013.name = "Champions Trophy
2013"
3 FOREACH (n IN nodes(p)| SET n.marked
= TRUE)

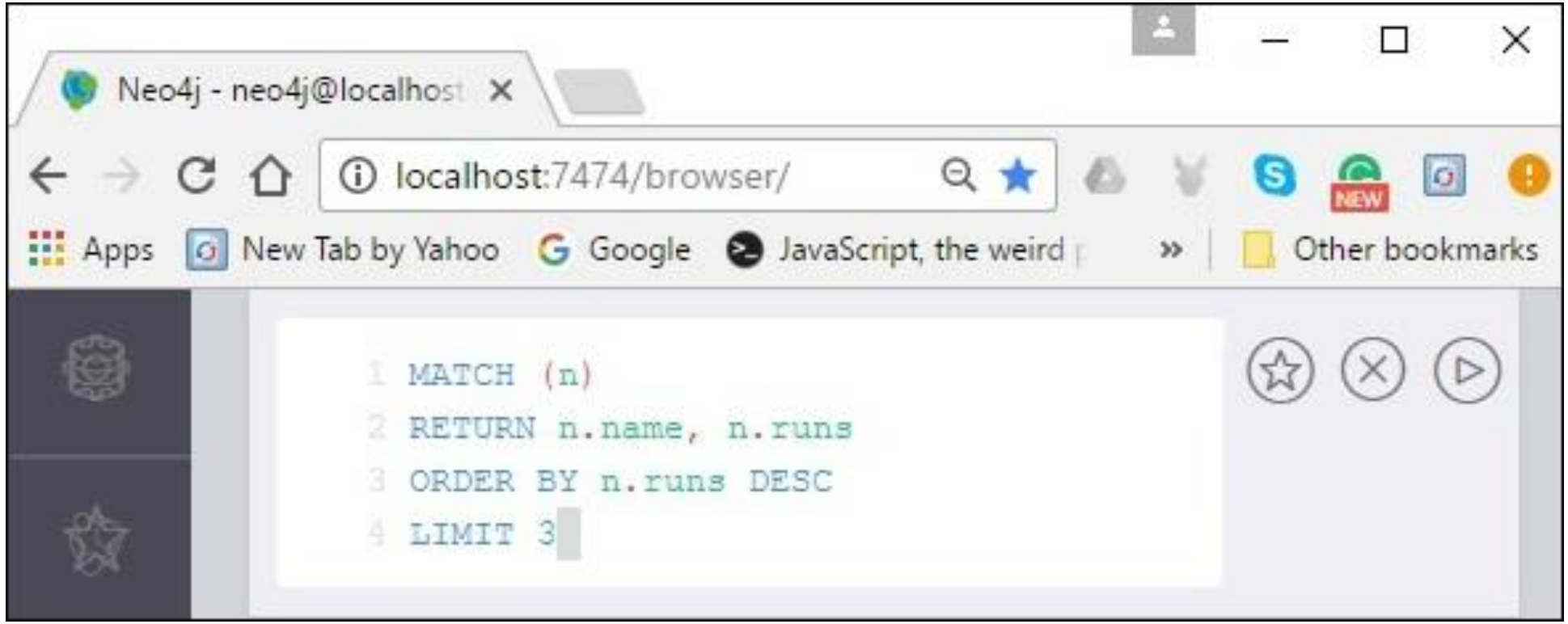

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
$
Rows
Text
Code
shikar Dhawan
TOP_SCORER_OF
India
WINNERS_OF
ICC Champi...
Ravindra Jadeja
HIGHEST_WICKET_TAKER_OF
CAPTAIN_OF
Mahend...


Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 MATCH (a:Tornament {name:
"ICC Champions Trophy
2013"})
2 OPTIONAL MATCH (a)-->(x)
3 RETURN x


Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
MATCH (player)
WHERE player.country = "India"
RETURN player


Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
$ Match(n{name: "India",
result: "Winners"})--(x)
RETURN n, count(*)


Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
1 MATCH (n)
2 RETURN n.name, n.runs
3 ORDER BY n.runs DESC

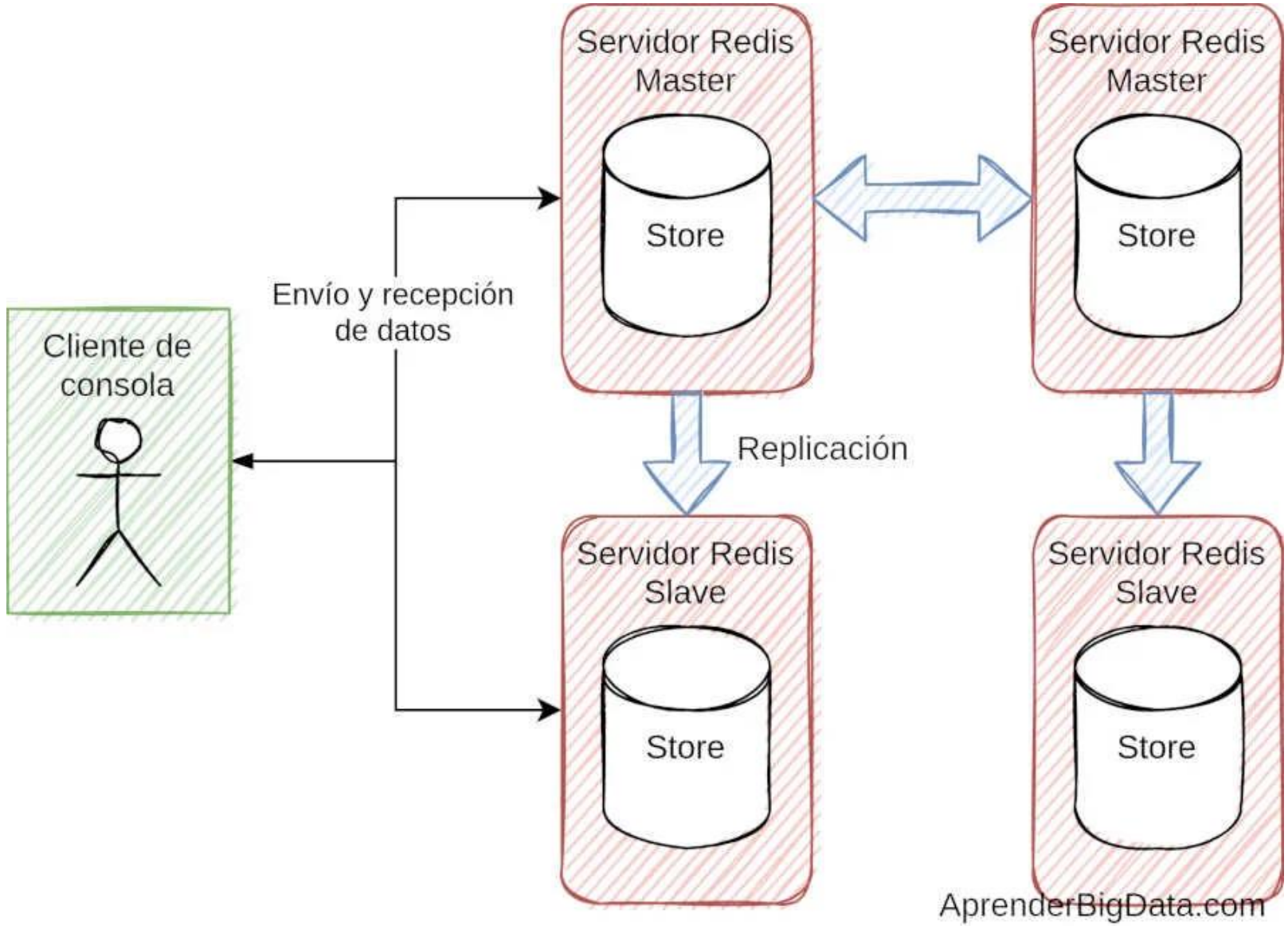

Neo4j - neo4j@localhost
localhost:7474/browser/
Apps
New Tab by Yahoo
Google
JavaScript, the weird
Other bookmarks
MATCH (n)
RETURN n.name, n.runs
ORDER BY n.runs DESC
LIMIT 3

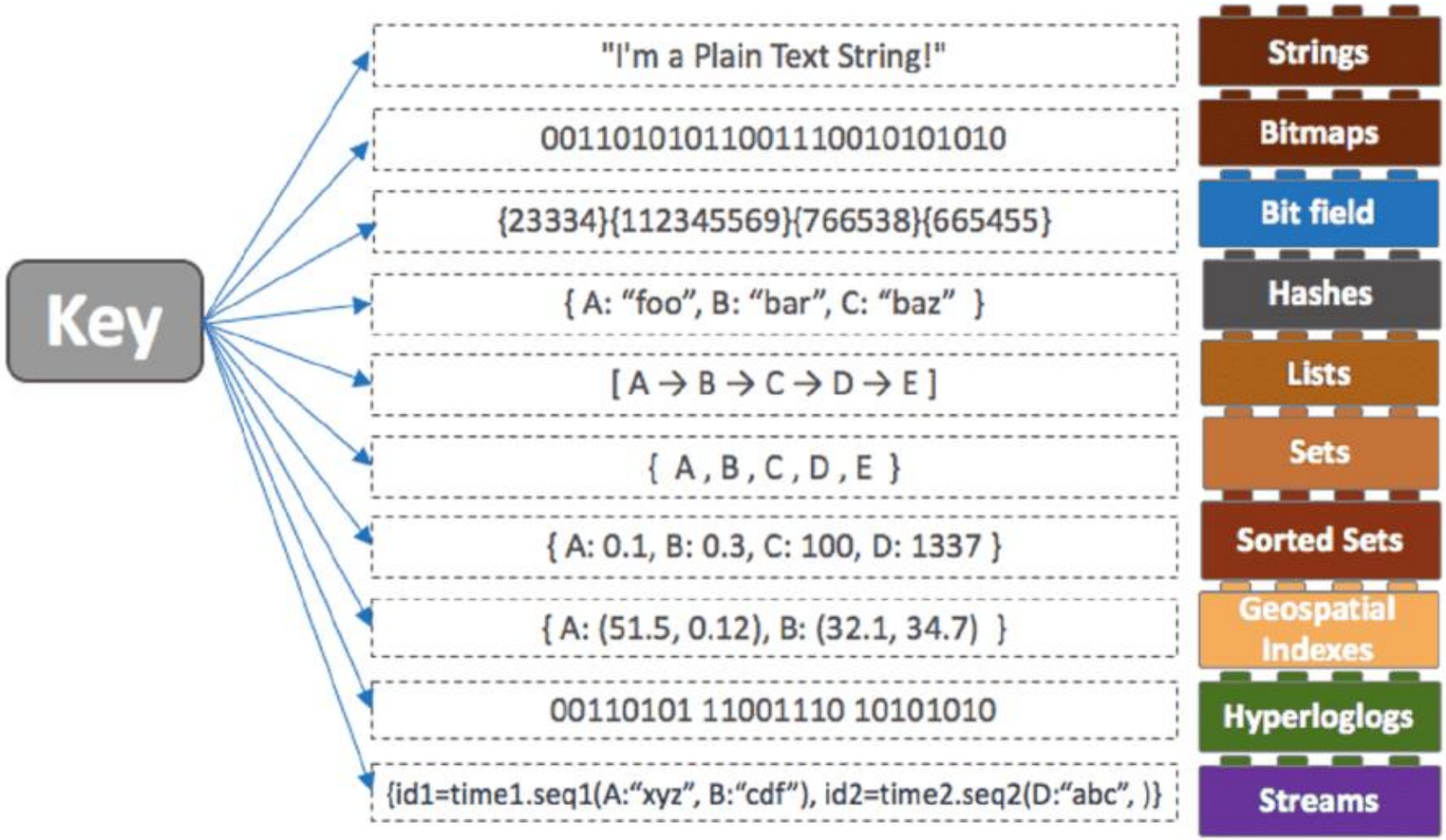
EJERCICIO

https://neo4j.com/docs/getting-started/appendix/tutorials/guide-cypher-basics/

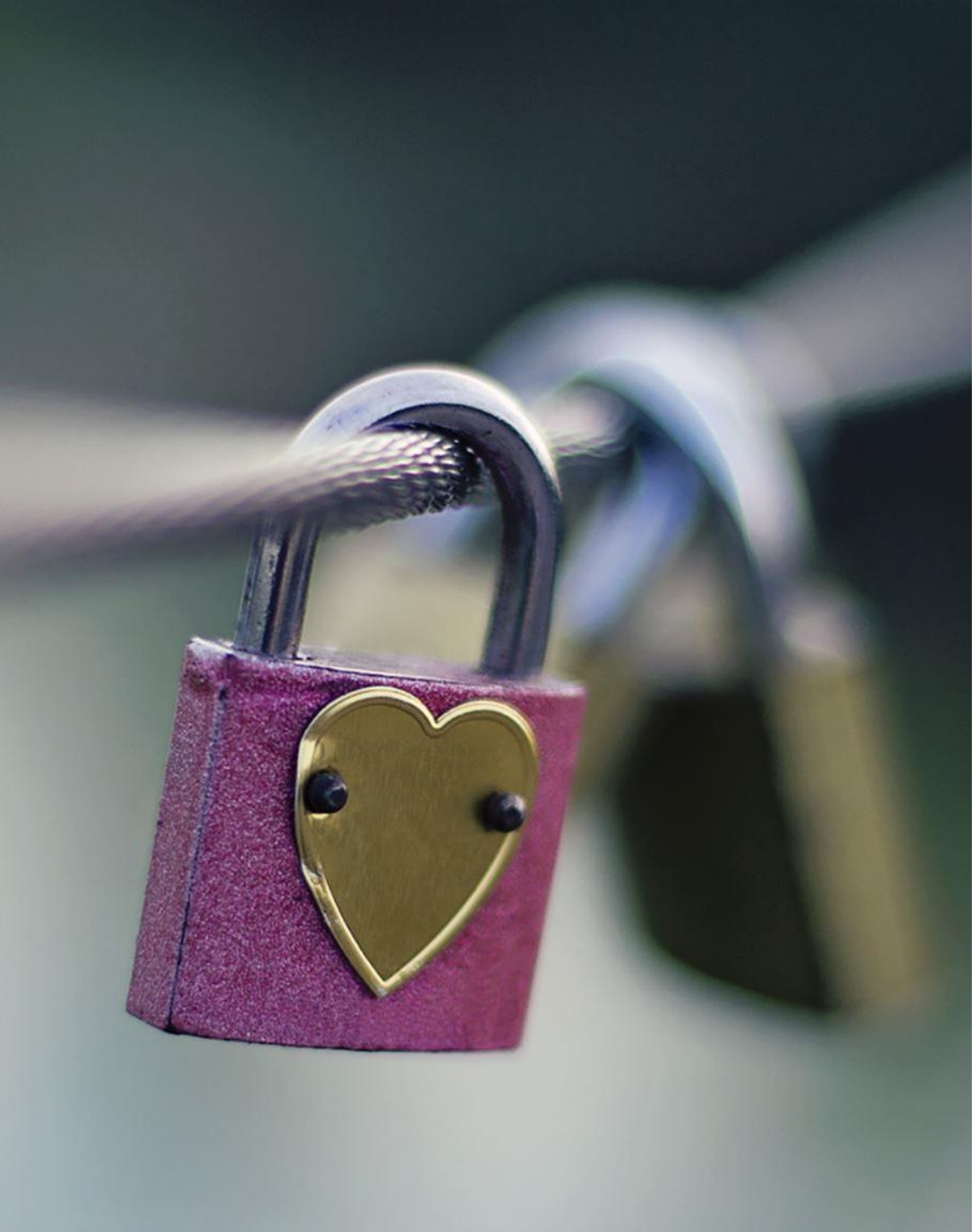
REDIS

- Redis es una estructura de datos en memoria que se utiliza para acceder a los datos con mayor rapidez. Se utiliza para almacenar datos a los que se debe acceder con frecuencia y rapidez. No se utiliza para almacenar grandes cantidades de datos. Si desea almacenar y recuperar grandes cantidades de datos, debe utilizar una base de datos tradicional como MongoDB o MYSQL. Redis ofrece una variedad de estructuras de datos, como conjuntos, cadenas, hashes y listas.
- El servidor Redis es un programa que se ejecuta y almacena datos en la memoria.
- Puede simplemente conectarse a ese servidor y usarlo para almacenar y recuperar datos más rápido.
- Por ese motivo, Redis no se utiliza para el almacenamiento persistente de datos, ya que se perderán todos los datos si el sistema falla.
- Redis es escalable ya que puedes ejecutar múltiples instancias del servidor.
- A menudo se utiliza como caché que almacena datos temporalmente y proporciona un acceso más rápido a los datos utilizados con frecuencia.


Servidor Redis Master
Store
Servidor Redis Master
Store
Envío y recepción de datos
Cliente de consola
Replicación
Servidor Redis Slave
Store
Servidor Redis Slave
Store
AprenderBigData.com

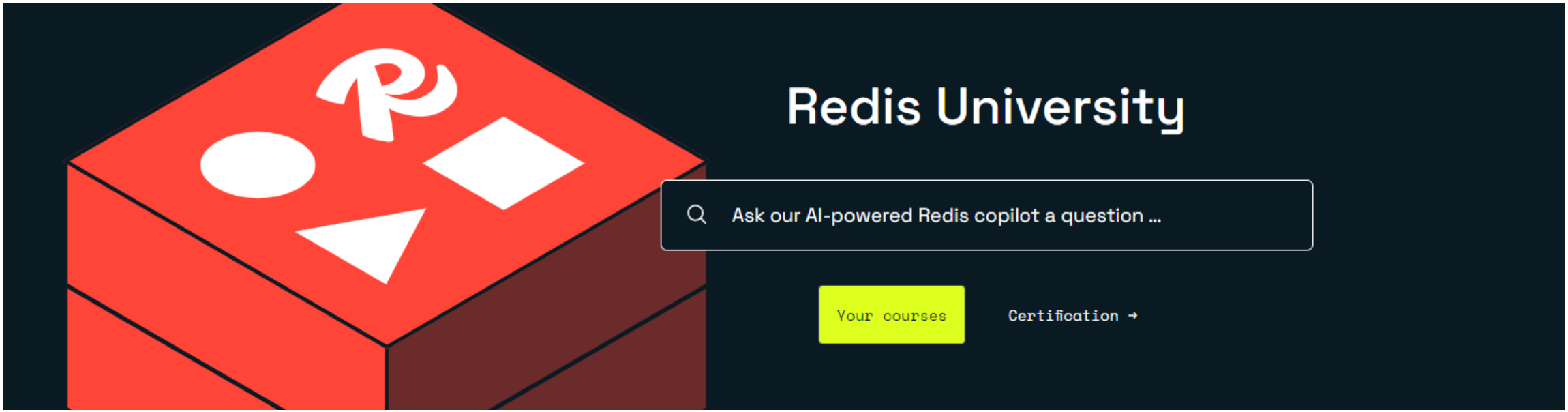

Key
"I'm a Plain Text String!"
00110101011001110010101010
{23334}{112345569}{766538}{665455}
{ A: "foo", B: "bar", C: "baz" }
[A → B → C → D → E]
{ A , B , C , D , E }
{ A: 0.1, B: 0.3, C: 100, D: 1337 }
{ A: (51.5, 0.12), B: (32.1, 34.7) }
00110101 11001110 10101010
{id1=time1.seq1(A:"xyz", B:"cdf"), id2=time2.seq2(D:"abc",)}
Strings
Bitmaps
Bit field
Hashes
Lists
Sets
Sorted Sets
Geospatial Indexes
Hyperloglogs
Streams


EJERCICIO



TRANSACCIONES

- multi
- set key_MeaningOfLife 1
- incr key_MeaningOfLife
- incrby key_MeaningOfLife 40
- get key_MeaningOfLife
- exec


Redis University
Ask our AI-powered Redis copilot a question ...
Your courses
Certification →