

NoSQL I

Alex Di Genova

15/06/2023

Información

Control 2 Jueves 22/06

Votación para acordar fecha del control 2.

Martes 20 de Junio 8

18.6%

Jueves 22 de Junio 35

81.4%

Categoría Consultas

Abierta hasta Ayer, a las 14:00 hrs.
Cerrada

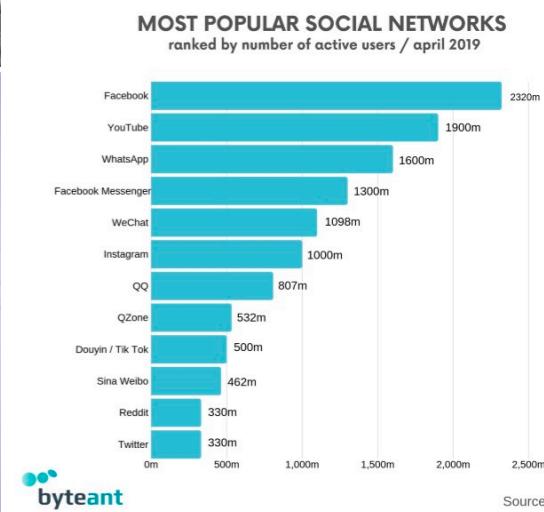
Estado Publicada

Última Modificación Martes 13 de Junio a las 12:43 hrs.

- Sala será informada la proxima semana.

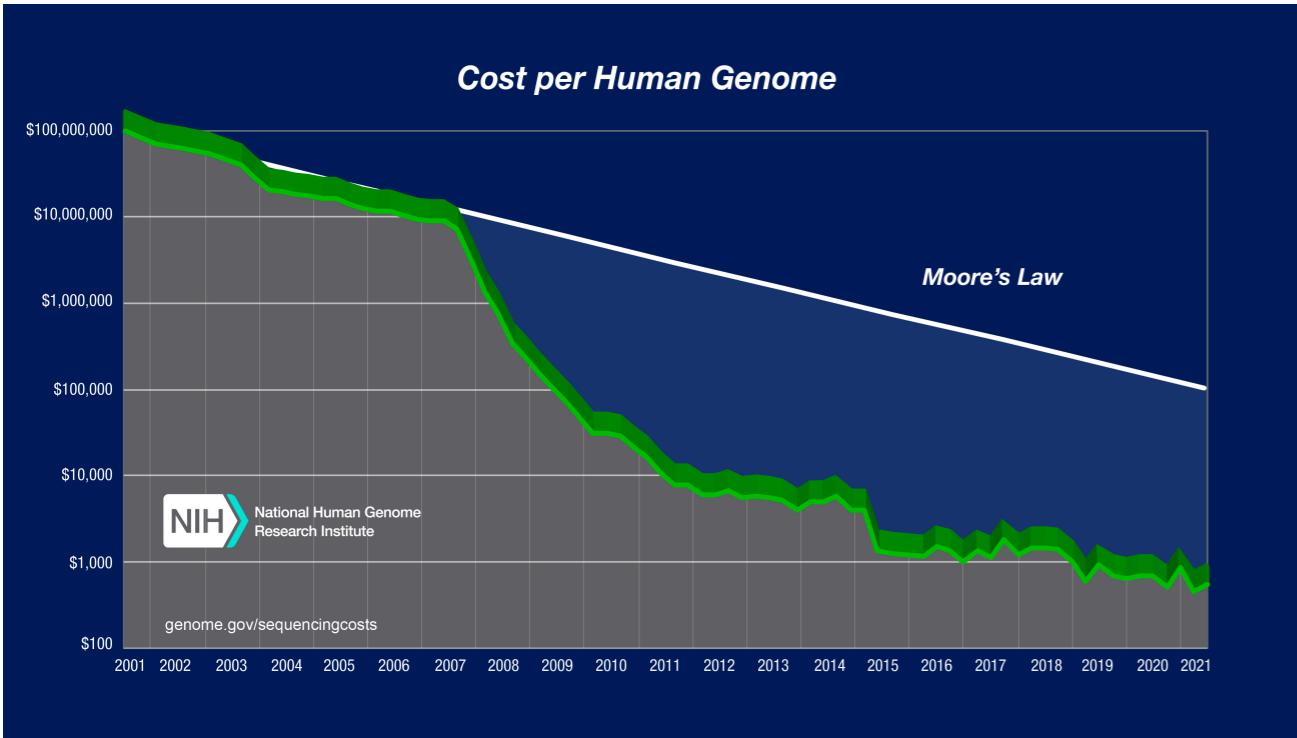
NoSQL

- El término NoSQL se utilizó por primera vez en 1998 para una base de datos que (aunque relacional) no tenía una interfaz SQL.
- Más importante durante la década de 2000, especialmente con la rápida expansión de Internet (web-scale databases).
- Los componentes de procesamiento orientados a la consistencia a menudo dificultan el procesamiento eficiente de grandes cantidades de datos (big data).

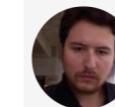


NoSQL

Ejemplo genómica



Ultima Genomics



Alex Di Genova @digenoma · May 30

\$1/Gb go Ultima genomics!!!! reads of ~400bp!!!

...

bioRxiv @biorxivpreprint · May 30

Cost-efficient whole genome-sequencing using novel mostly natural sequencing-by-synthesis chemistry and open fluidics platform
biorxiv.org/cgi/content/sh... #bioRxiv

1

5

29

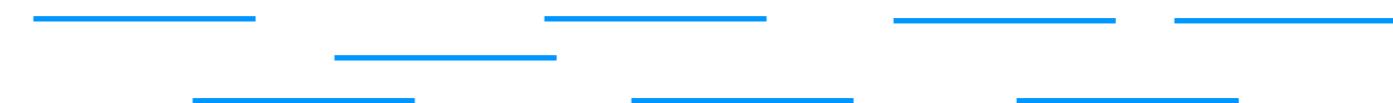
↑

...

Genome

Sequencing technology

Reads



Assembler

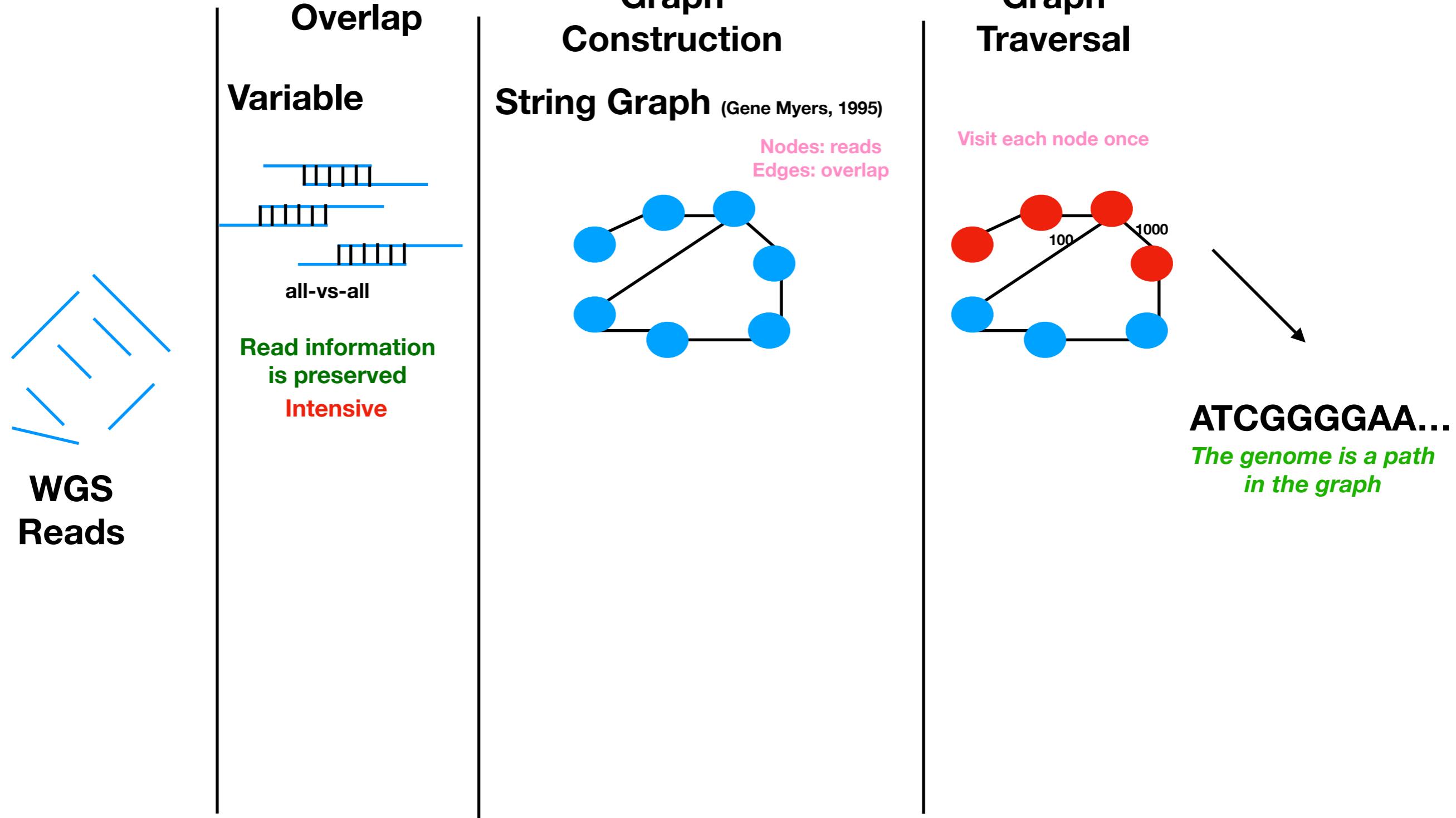
Overlap



Contigs

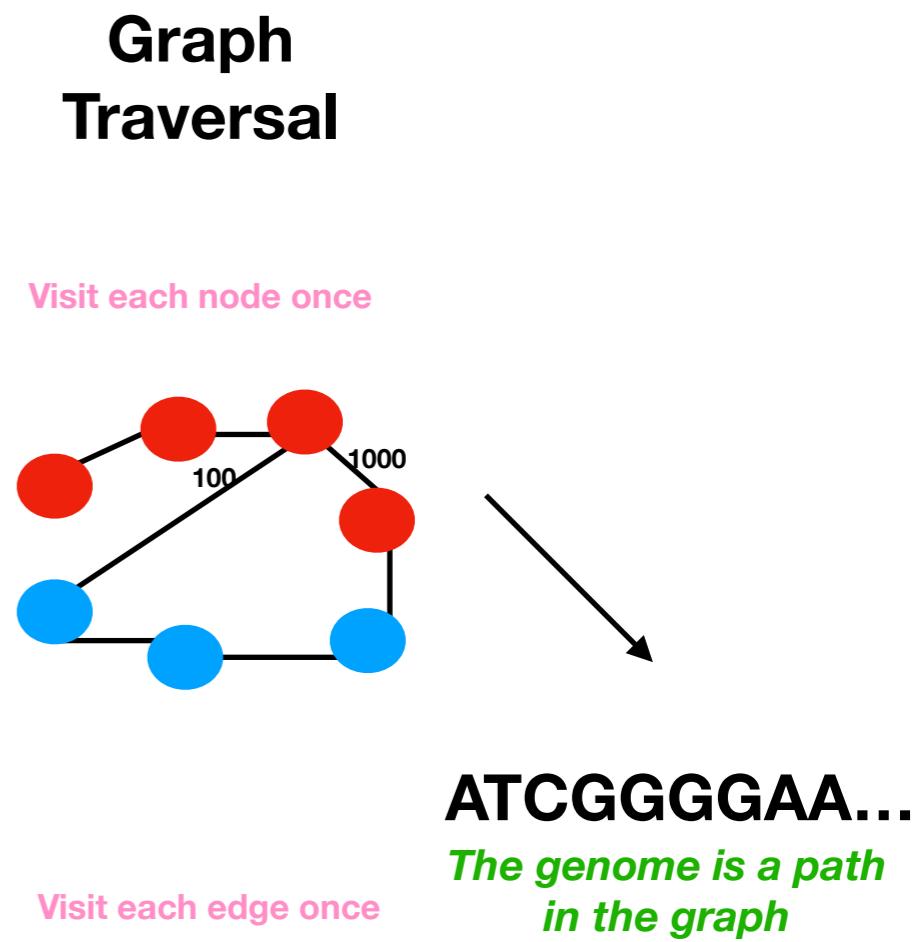
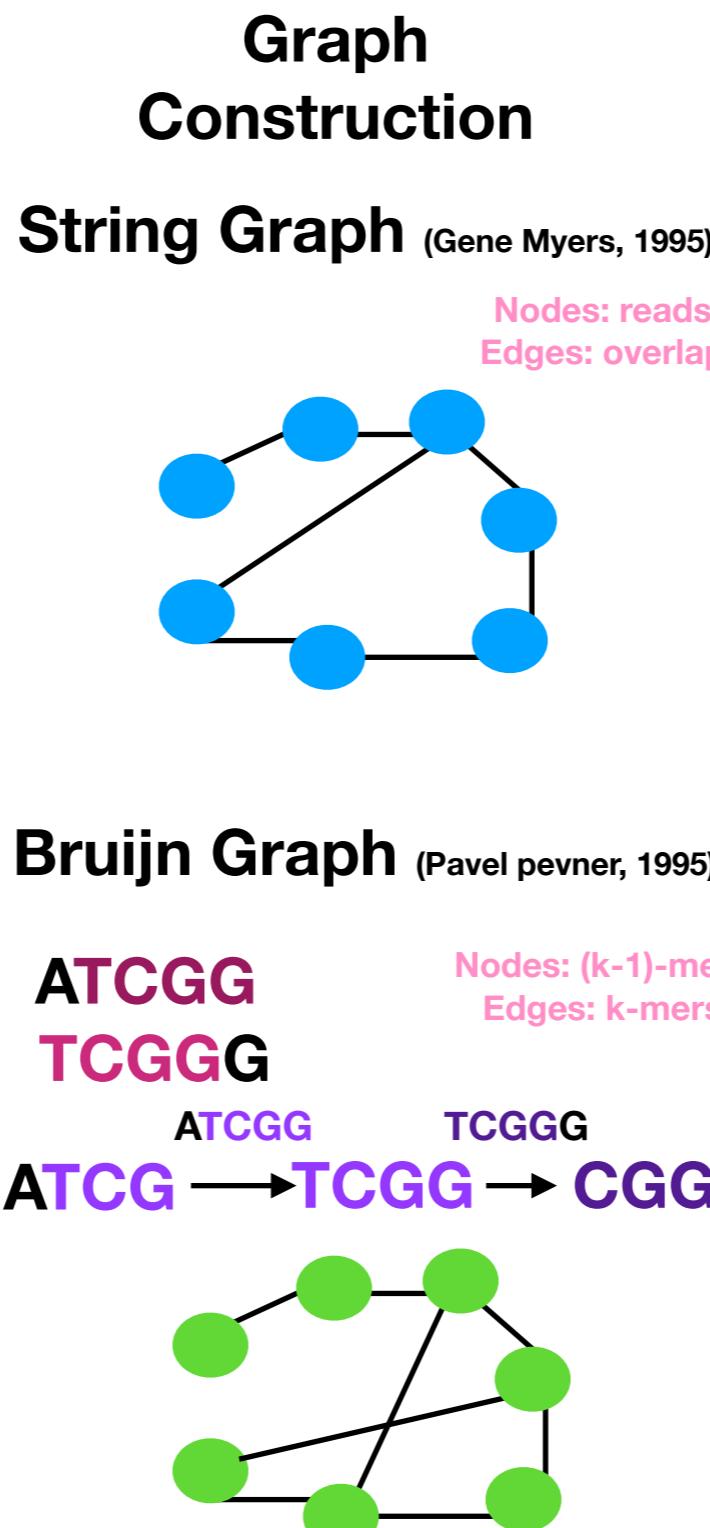
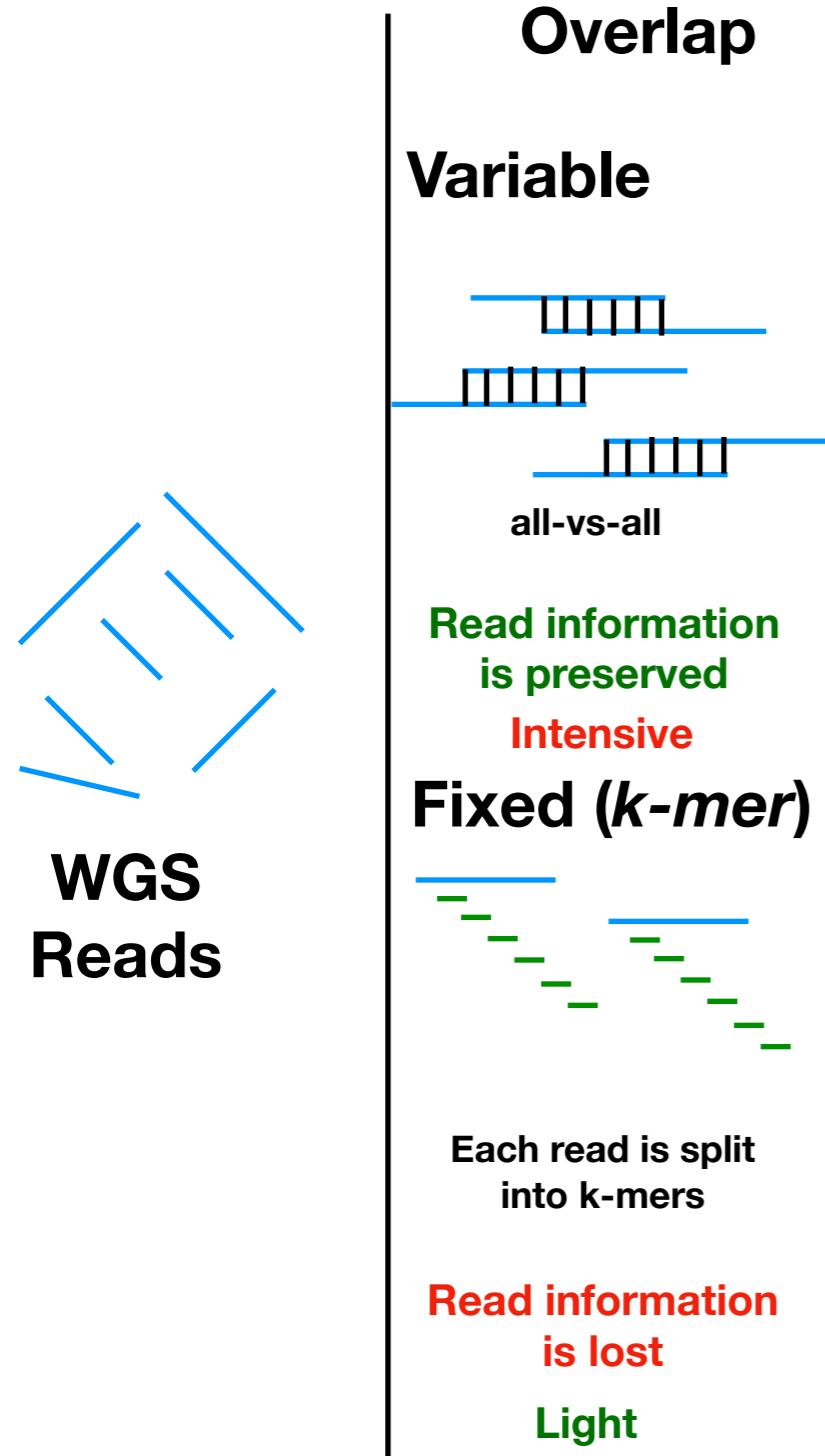
NoSQL

Ejemplo genómica



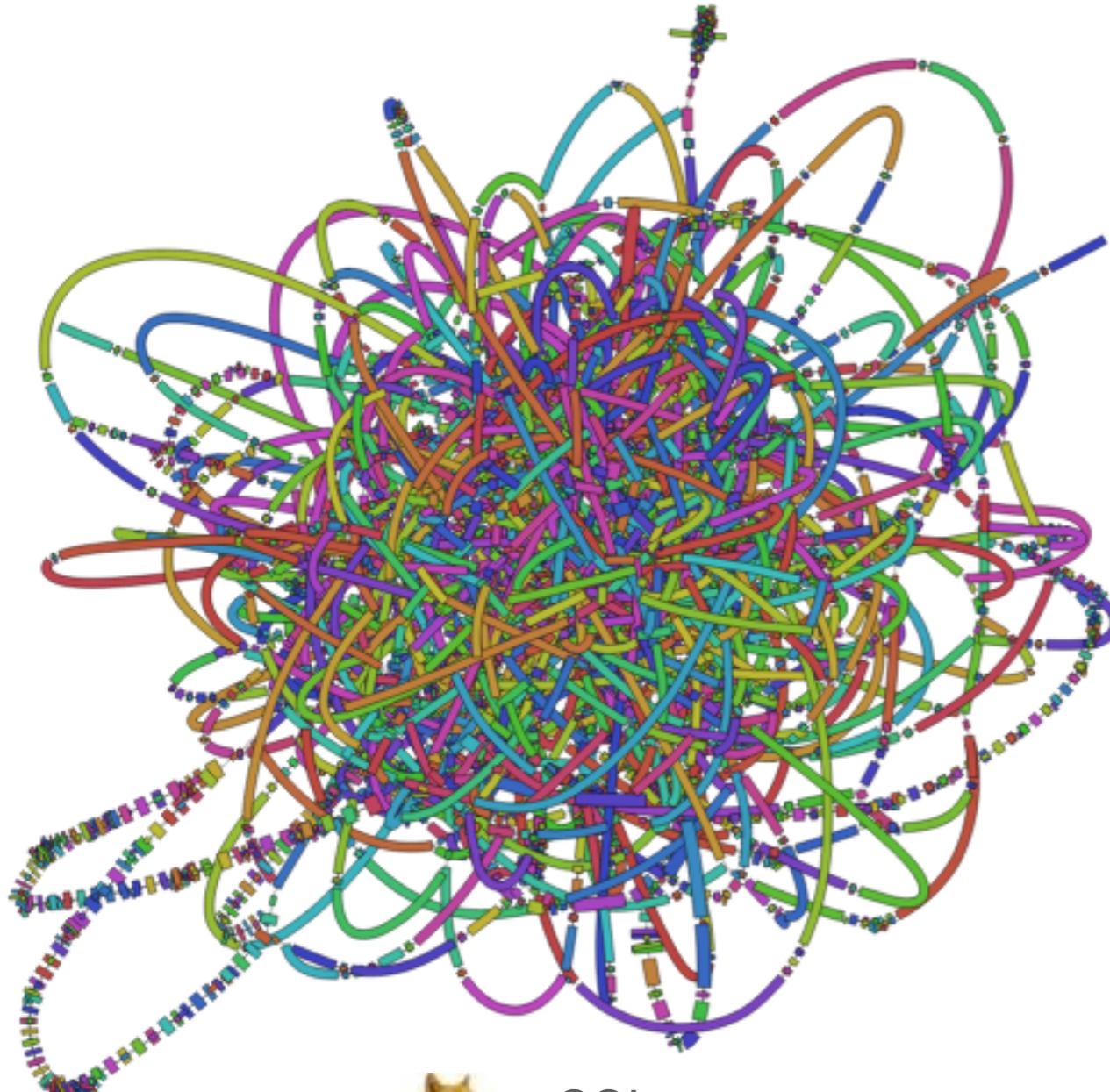
NoSQL

Ejemplo genómica



NoSQL

Ejemplo genómica



Hospitales
Ventas
Organizaciones



- Genómica
- Astronomía
- Web
- Redes sociales

- Genoma Humano:
- 3Gb
- 180Gb por persona
- Grafo
- >3000,000,000 nodos.
- >3000,000,000 conexiones.,,

ARTICLES

<https://doi.org/10.1038/s41587-020-00747-w>

nature
biotechnology

Check for updates

OPEN

Efficient hybrid de novo assembly of human genomes with WENGAN

Alex Di Genova^{1,2}, Elena Buena-Atienza^{3,4}, Stephan Ossowski^{3,4} and Marie-France Sagot^{1,2}

Generating accurate genome assemblies of large, repeat-rich human genomes has proved difficult using only long, error-prone reads, and most human genomes assembled from long reads add accurate short reads to polish the consensus sequence. Here we report an algorithm for hybrid assembly, WENGAN, that provides very high quality at low computational cost. We demonstrate de novo assembly of four human genomes using a combination of sequencing data generated on ONT PromethION, PacBio Sequel, Illumina and MGI technology. WENGAN implements efficient algorithms to improve assembly contiguity as well as consensus quality. The resulting genome assemblies have high contiguity (contig NG50: 17.24–80.64 Mb), few assembly errors (contig NGA50: 11.8–59.59 Mb), good consensus quality (QV: 27.84–42.88) and high gene completeness (BUSCO complete: 94.6–95.2%), while consuming low computational resources (CPU hours: 187–1,200). In particular, the WENGAN assembly of the haploid CHM13 sample achieved a contig NG50 of 80.64 Mb (NGA50: 59.59 Mb), which surpasses the contiguity of the current human reference genome (GRCh38 contig NG50: 57.88 Mb).

NoSQL

Actualmente

- El término ha adquirido diferentes significados.
- Una interpretación común es "no solo SQL"
- La mayoría de los sistemas NoSQL modernos difieren del modelo relacional o la funcionalidad RDBMS estándar:

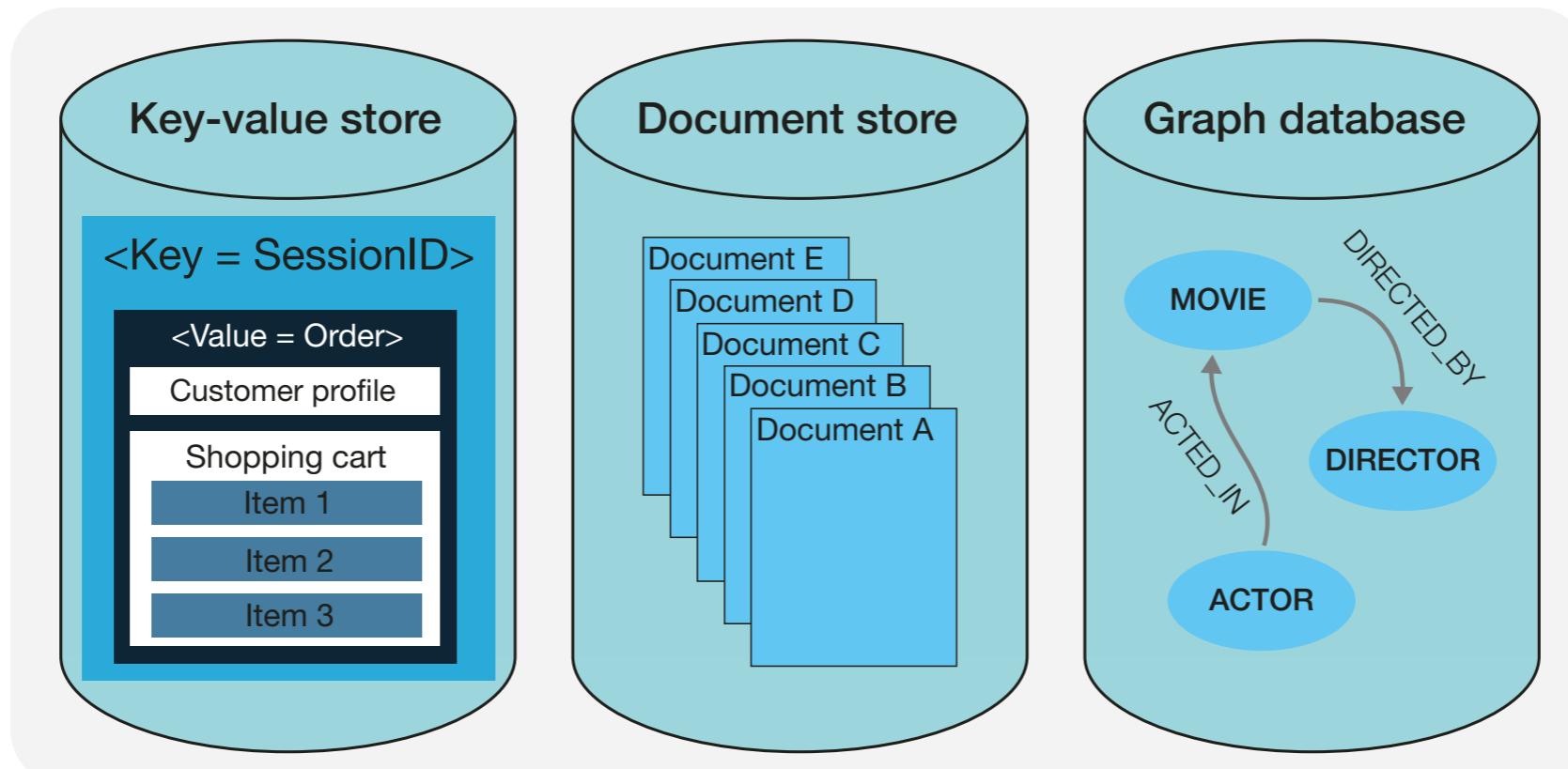
	SQL	NoSQL
Modelo de datos	Relaciones Tuplas Atributos Dominios Normalización	Documentos Grafos clave/valor
Modelo de consultas	SQL Algebra relacional	Recorridos en grafos Busqueda en texto Map/reduce
Implementación	Esquemas rígidos Propiedades ACID	Esquemas flexibles BASE

NoSQL hoy en día es más comúnmente destinado a ser algo así como "no relacional".

NoSQL

Categorías primarias de NoSQL

- Categorías generales de sistemas NoSQL
 - Almacenamiento basado en Clave/valor
 - Almacenamiento basado en documentos
 - Almacenamiento basado en grafos

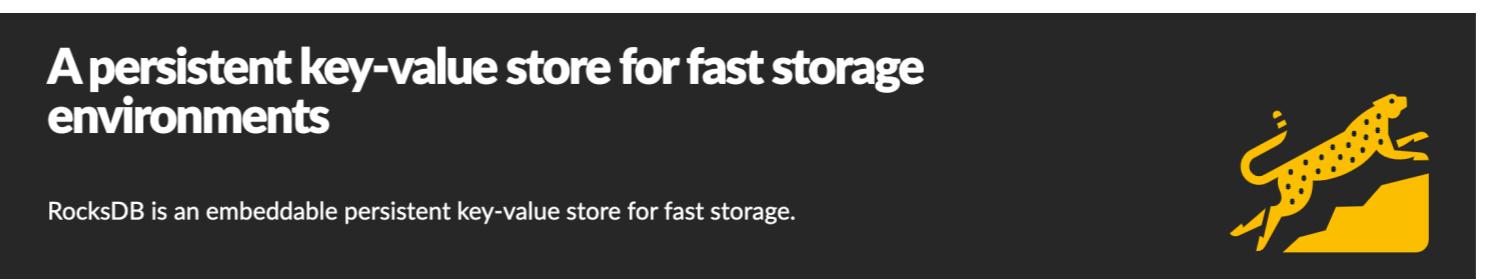


DynamoDB
Azure Table Storage
Riak
Redis
Aerospike
FoundationDB
LevelDB
Berkeley DB
Oracle NoSQL Database
GenieDb
BangDB
Chordless
Scalaris
Tokyo Cabinet/Tyrant
Scalien
Voldemort
Dynomite
KAI
MemcacheDB
Faircom C-Tree
LSM
KitaroDB
HamsterDB
STSdb
TarantoolBox
Maxtable
Quasardb
Pincaster
RaptorDB
TIBCO Active Spaces
Allegro-C
nessDB
HyperDex
SharedHashFile
Symas LMDB
Sophia
PickleDB
Mnesia
LightCloud
Hibari
OpenLDAP
Genomu
BinaryRage
Elliptics
Dbreeze
RocksDB
TreodeDB
www.nosql-database.org
www.db-engines.com
www.wikipedia.com

NoSQL

Almacenamiento basado en Clave/valor

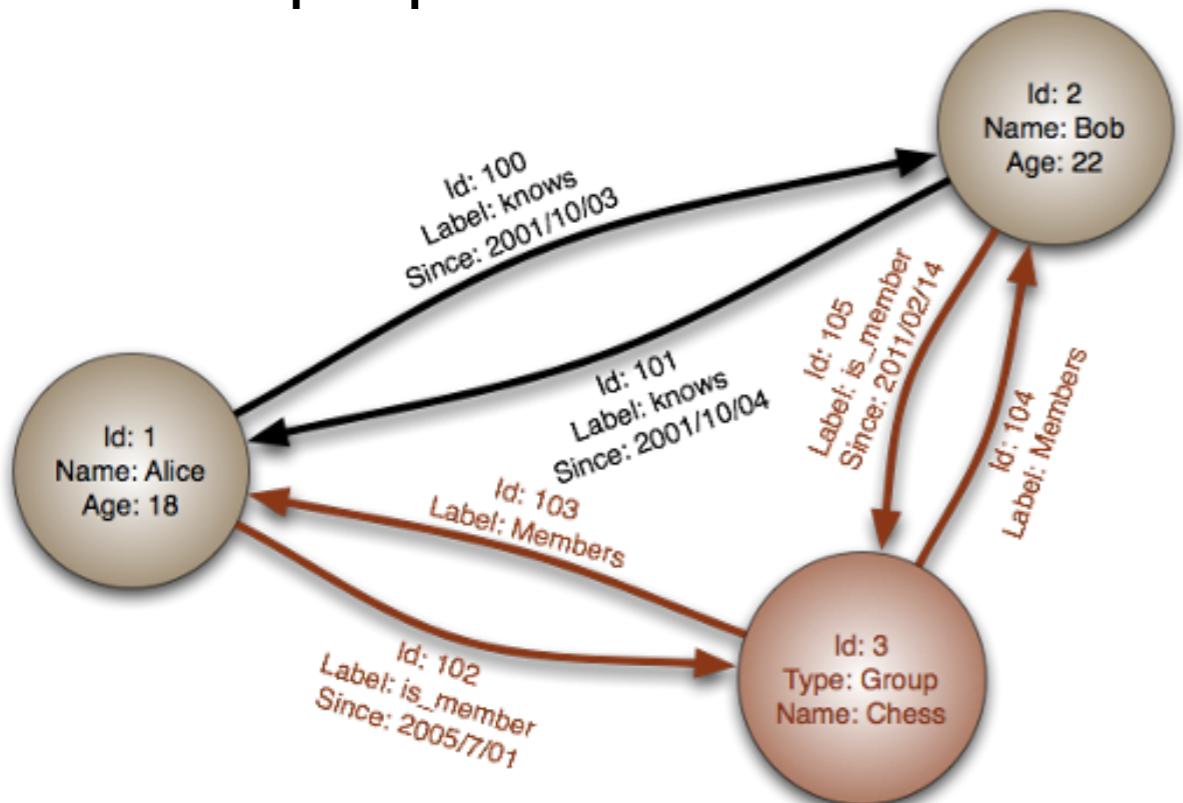
- El modelo básico de datos:
 - La base de datos es una colección de pares clave/valor
 - La clave para cada par es única.
- Operaciones primarias
 - `insert(key, value)`
 - `delete(key, value)`
 - `update(key, value)`
 - `lookup(key)`



NoSQL

Almacenamiento basado en grafos

- El modelo básico de datos:
 - Grafos dirigidos
 - Nodos y arcos con propiedades



NEO4J GRAPH DATA PLATFORM

Blazing-Fast Graph, Petabyte Scale

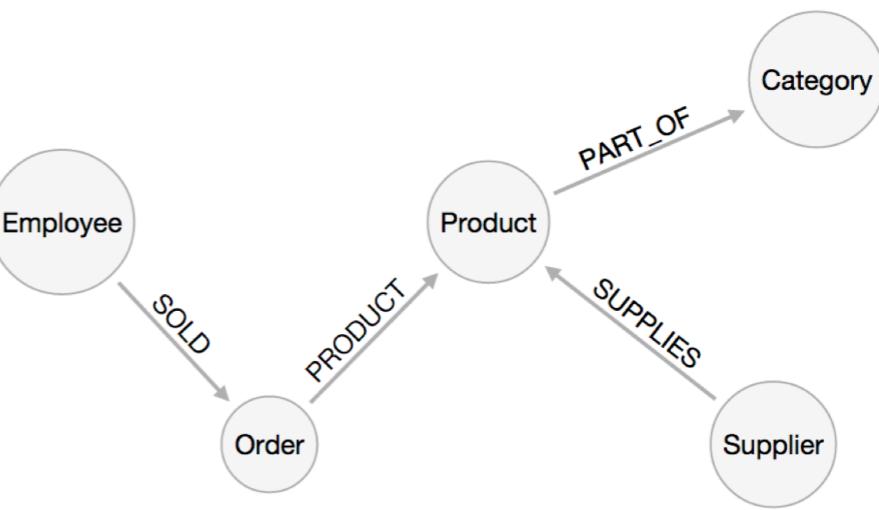
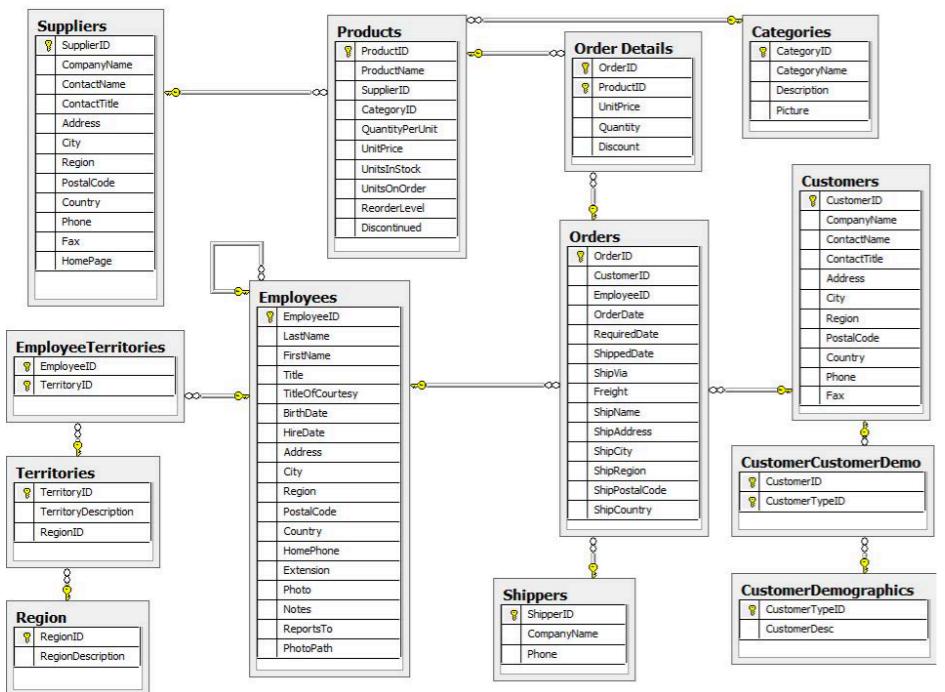
With proven [trillion+ entity performance](#), developers, data scientists, and enterprises rely on Neo4j as the top choice for high-performance, scalable analytics, intelligent app development, and advanced AI/ML pipelines.

AllegroGraph
ArangoDB
Bigdata
Betsy
BrightstarDB
DEX/Sparksee
Execom IOG
Fallen *
Filament
FlockDB
GraphBase
Graphd
Horton
HyperGraphDB
Item G Native Store
InfiniteGraph
InfoGrid
jCoreDB Graph
MapGraph
Meronymy
Neo4j
Orly
OpenLink virtuoso
Spatial and Graph
Simple NoSQL Database
OrientDB
OQGraph
Ontotext OWLIM
R2DF
ROIS
Sones GraphDB
SPARQLCity
Sqrrl Enterprise
Stardog
Teradata Aster
Titan
Trinity
TripleBit
VelocityGraph
VertexDB
WhiteDB
www.nosql-database.org
www.db-engines.com
www.wikipedia.com

NoSQL

Almacenamiento basado en grafos

- El modelo básico de datos:
 - Grafos dirigidos
 - Nodos y arcos con propiedades



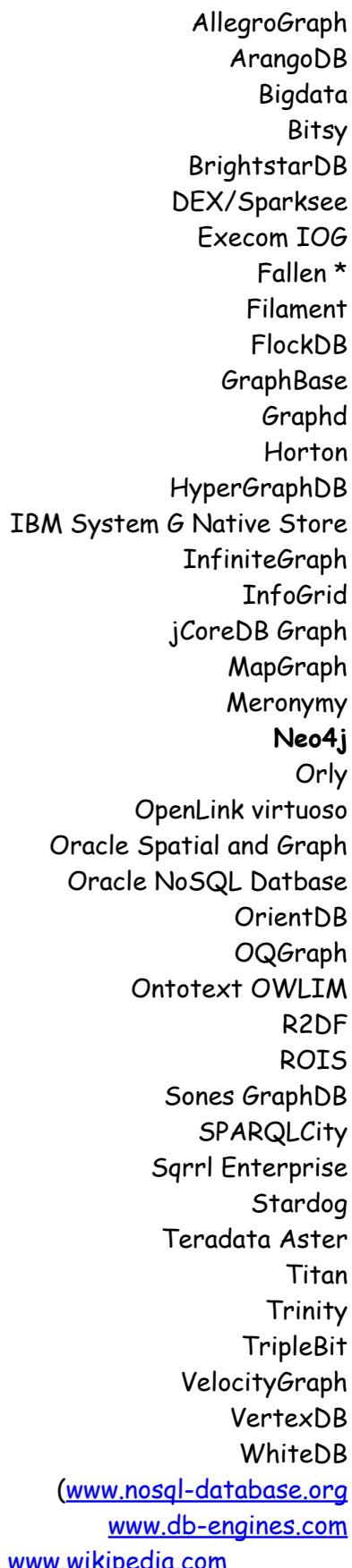
SQL

```
SELECT p.ProductName, p.UnitPrice  
FROM products AS p  
WHERE p.ProductName = 'Chocolade';
```

Cypher es un lenguaje de consulta potente, intuitivo y optimizado para grafos que comprende y aprovecha las conexiones de datos.

Cypher

```
MATCH (p:Product)  
WHERE p.productName = "Chocolade"  
RETURN p.productName, p.unitPrice;
```



NoSQL

Almacenamiento basado en documentos

- El modelo básico de datos:
 - La noción general de un documento: palabras, frases, oraciones, párrafos, secciones, subsecciones, notas al pie, etc.
 - Esquema flexible: la estructura de los subcomponentes se puede anidar y variar de documento a documento.
 - Metadatos: título, autor, fecha, etiquetas incrustadas, etc.
 - Clave/identificador.

```
{  
    "_id": 1,  
    "first_name": "Tom",  
    "email": "tom@example.com",  
    "cell": "765-555-5555",  
    "likes": [  
        "fashion",  
        "spas",  
        "shopping"  
    ],  
    "businesses": [  
        {  
            "name": "Entertainment 1080",  
            "partner": "Jean",  
            "status": "Bankrupt",  
            "date_founded": {  
                "$date": "2012-05-19T04:00:00Z"  
            }  
        },  
        {  
            "name": "Swag for Tweens",  
            "date_founded": {  
                "$date": "2012-11-01T04:00:00Z"  
            }  
        }  
    ]  
}
```

AmisaDB
ArangoDB
BaseX
Cassandra
Cloudant
Clusterpoint
Couchbase
CouchDB
Densodb
Djondb
EJDB
Elasticsearch
eXist
FleetDB
iBoxDB
Inquire
JasDB
MarkLogic
MongoDB
MUMPS
NeDB
NoSQL embedded db
OrientDB
RaptorDB
RavenDB
RethinkDB
SDB
SisoDB
Terrastore
ThruDB

(www.nosql-database.org
www.db-engines.com
www.wikipedia.com)

NoSQL

Almacenamiento basado en documentos

- Las bases de datos de documentos suelen tener una API o un lenguaje de consulta que permite a los desarrolladores ejecutar operaciones (crear, leer, actualizar y eliminar).
 - **Crear:** Los documentos se pueden crear en la base de datos. Cada documento tiene un identificador único.
 - **Leer:** Los documentos se pueden leer desde la base de datos. La API o lenguaje de consulta permite a los desarrolladores consultar documentos utilizando sus identificadores únicos o valores de campo. Se pueden agregar índices a la base de datos para aumentar el rendimiento de lectura.
 - **Actualizar:** los documentos existentes se pueden actualizar, ya sea en su totalidad o en parte.
 - **Eliminar:** los documentos se pueden eliminar de la base de datos.

```
db.inventory.insertMany([
  { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
  { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
  { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
  { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
  { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

```
SELECT * FROM inventory
db.inventory.find( {} )

SELECT * FROM inventory WHERE status = "D"
db.inventory.find( { status: "D" } )

SELECT * FROM inventory WHERE status = "A" AND qty < 30
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

bases de datos multimodelo

Definición

- Una base de datos que consta de diferentes mecanismos de almacenamiento de datos (base de datos relacional, documentos, clave/valor, grafos)
 - Motor de base de datos todo en uno
 - Con un lenguaje de consulta y una API unificadores
 - Incluyan todos los modelos de datos e incluso permitan mezclarlos en una única consulta.

bases de datos multimodelo

Ejemplos

- ArangoDB – documentos (JSON), grafo, clave-valor
- CouchBase: relacional (SQL), documentos
- CrateDB: relacional (SQL), documentos (Lucene)
- MarkLogic: documentos (XML y JSON), grafos (RDF con OWL/RDFS), texto, geoespacial, binario, SQL
- OrientDB: documentos (JSON), grafos, clave-valor, texto, binario, reactivo, SQL
- Datastax: clave-valor, tabular, grafos
-

bases de datos multimodelo

Desarrollo

- Benchmarking (comparación con modelos estandar)
- Extension de los lenguajes de consultas existentes
- Procesamiento de consultas
 - Joins complejos entre modelos de datos.
 - Nuevas estructuras para indices.
- Transacciones y consistencia de la DB.

bases de datos multimodelo

Benchmark

- Basado en ArangoDB
- <https://www.arangodb.com/2018/02/nosql-performance-benchmark-2018-mongodb-postgresql-orientdb-neo4j-arangodb/>



bases de datos multimodelo

Benchmark

- **Lectura única:** lectura de un solo documento de perfiles (100.000 documentos)
- **Escritura única:** escrituras de documento único de perfil (100.000 documentos)
- **Agregación:** agregación ad-hoc sobre una sola colección (1.632.803 registros). Calculo de edad de cada individuo en la red.
- **Vecinos:** encontrar vecinos directos (distintos) más vecinos de vecinos, devolviendo ID (por 1.000 vértices)
- **Vecinos con datos:** encontrar vecinos directos (distintos) más los vecinos de los vecinos y devolver sus perfiles (para 100 vértices)
- **ruta más corta:** estas son las 1000 rutas más cortas que se encuentran en un grafo social altamente conectado. Esto responde a la pregunta de qué tan cerca están dos personas en la red social.
- **memoria:** este es el promedio del consumo máximo de memoria principal durante las ejecuciones de prueba.

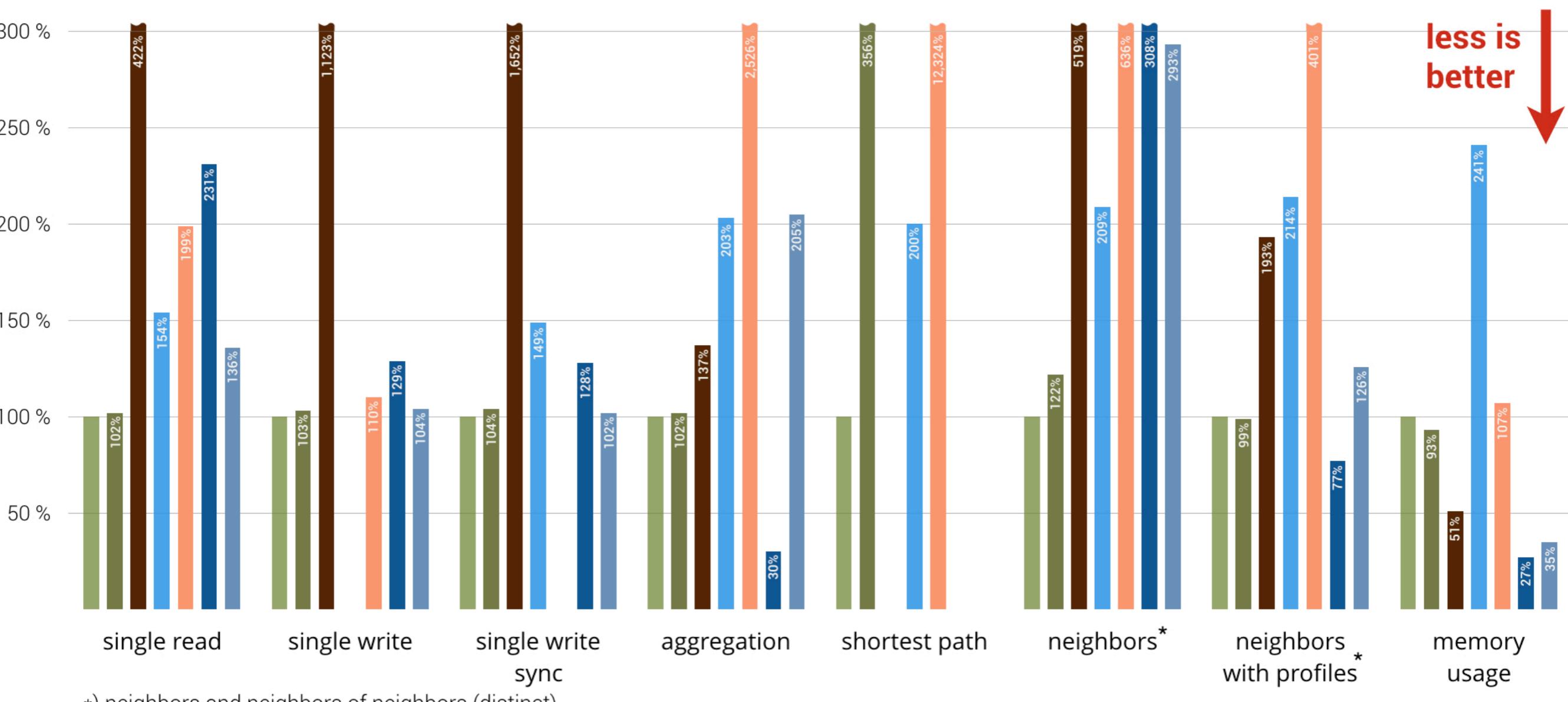
Las mediciones de rendimiento para ArangoDB, con RocksDB como motor de almacenamiento, definieron la línea de base (100 %) para las comparaciones. Los porcentajes más bajos indican un mayor rendimiento.

bases de datos multimodelo

NoSQL Performance Benchmark 2018
ArangoDB, MongoDB, Neo4j, OrientDB and PostgreSQL

ArangoDB Rocks MongoDB OrientDB
ArangoDB MMfiles Neo4j Postgres (tab)
Postgres (jsonb)

less is better



arangodb.com/performance – 2018-02-27

almacenamiento, definieron la línea de base (100 %) para las comparaciones. Los porcentajes más bajos indican un mayor rendimiento.

bases de datos multimodelo

Resultados benchmark

NoSQL Performance Bechmark 2018

Absolute & normalized results for ArangoDB, MongoDB, Neo4j and OrientDB

	single read (s)	single write (s)	single write sync (s)	aggregation (s)	shortest (s)	neighbors 2nd (s)	neighbors 2nd data (s)	memory (GB)
ArangoDB 3.3.3 (rocksdb)	100%	100%	100%	100%	100%	100%	100%	100%
	23.25	28.07	28.27	01.08	0.42	1.43	5.15	15.36
ArangoDB 3.3.3 (mmfiles)	102.16%	102.55%	103.89%	102.40%	816.06%	122.07%	99.32%	92.87%
	23.76	28.79	29.37	1.10	3.40	1.75	5.12	14.27
MongoDB 3.6.1 (Wired Tiger)	422.38%	1123.36%	1652.09%	136.65%		518.83%	192.88%	50.64%
	98.24	315.33	466.99	1.47		7.42	9.94	7.70
Neo4j 3.3.1	153.65%		149.37%	203.45%	199.94%	208.96%	214.22%	240.68%
	35.73		43.22	2.18	0.83	2.99	11.04	37.00
PostGres 10.1 (tabular)	231.17%	129.03%	127.70%	29.62%		307.96%	76.87%	26.68%
	53.77	36.22	36.10	0.32		4.41	3.96	4.10
PostGres 10.1 (jsonb)	135.96%	104.34%	101.55%	204.55%		292.57%	126.14%	35.36%
	31.62	29.29	28.70	2.20		4.19	6.50	5.43
OrientDB 2.2.29	198.84%	110.37%		2526.29%	12323.67%	636.45%	400.97%	107.04%
	46.25	30.98		27.19	51.34	9.11	20.67	16.45

Conclusión: el rendimiento y la flexibilidad de una DB multimodelo es una ventaja clave del motor ArangoDB.

ArangoDB

ArangoDB Google Colab

The screenshot shows a Google Colab notebook titled "Ejemplo_ArangoDB.ipynb". The notebook contains the following content:

ArangoDB

Tutorial de Arango DB

ArangoDB es una base de datos multimodelo con modelos de datos flexibles para documentos, grafos y clave/valor. Posee el lenguaje AQL que es similar al SQL para realizar consultas.

Instalación

Antes de comenzar con ArangoDB, debemos preparar nuestro entorno y crear una base de datos temporal en el Servicio Oasis.

```
[1] %%capture
2 git clone -b oasis_connector --single-branch https://github.com/arangodb/interactive_tutorials.git
3 lsync -av interactive_tutorials/. ./ --exclude=.git
4 pip3 install pyarango
5 !pip3 install "python-arango>=5.0"
```

```
[2]
1 import json
2 import requests
3 import sys
4 import oasis
5 import time
6
7 from pyArango.connection import *
8 from arango import ArangoClient
```

Creamos una base de datos temporal

```
[3]
1 # Retrieve tmp credentials from ArangoDB Tutorial Service
2 login = oasis.getTempCredentials(tutorialName="AQLCrudTutorial", credentialProvider='https://tutorials.arangodb.cloud:8529/_db/_system/tutorialDB/tutorialDB')
3 # Connect to the temp database
4 conn = oasis.connect(login)
5 pyAr_db = conn[login["dbName"]]

Requesting new temp credentials.
Temp database ready to use.
```

```
[4]
1 python_arango_db = oasis.connect_python_arango(login)
2 aql = python_arango_db.aql
```

```
[5]
1 print("https://{}:{}.".format(login["hostname"], login["port"]))
2 print("Username: " + login["username"])
3 print("Password: " + login["password"])
4 print("Database: " + login["dbName"])

https://tutorials.arangodb.cloud:8529
Username: TUTyldg5mspz09jvsgn79708
Password: TUTE56xixvmuxchib9htljdrv
Database: TUTy29y9plwkarpz9mkn2c628
```

Podemos usar la URL anterior para revisar nuestra base de datos arango temporal.

https://github.com/adigenova/uohdb/blob/main/code/NoSQL_ArangoDB_ejemplo.ipynb
<https://www.arangodb.com/docs/stable/aql/tutorial.html>

Consultas?

Consultas o comentarios?

Muchas gracias