

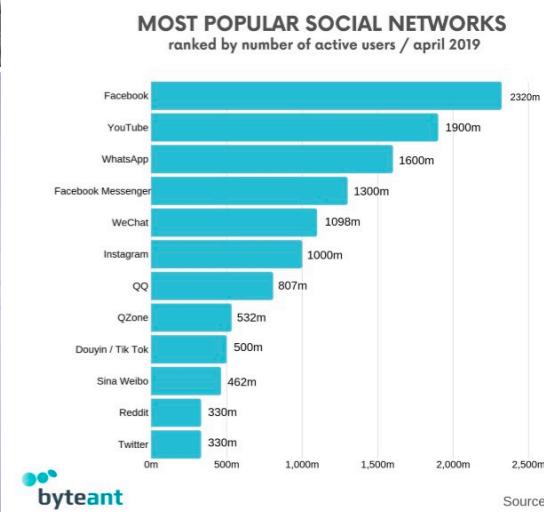
# NoSQL

Alex Di Genova

30/10/2023

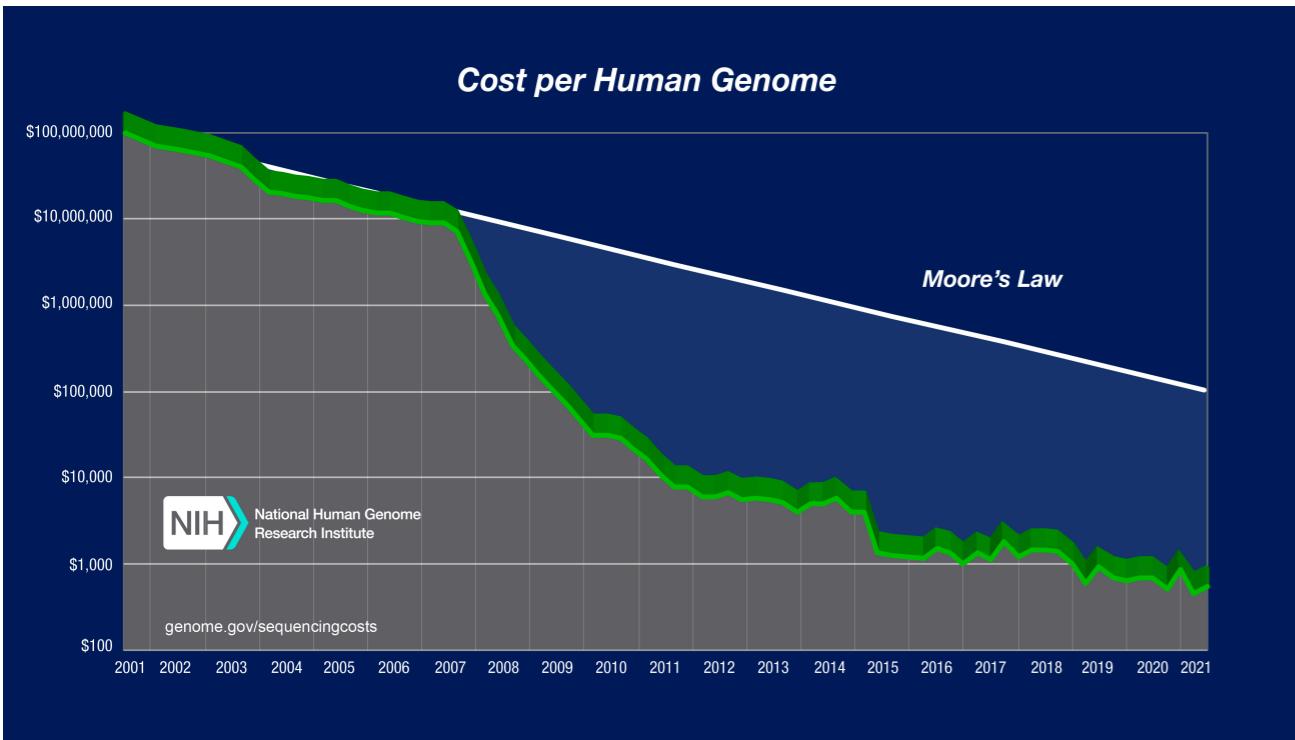
# NoSQL

- 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).

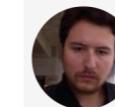


# NoSQL

## 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

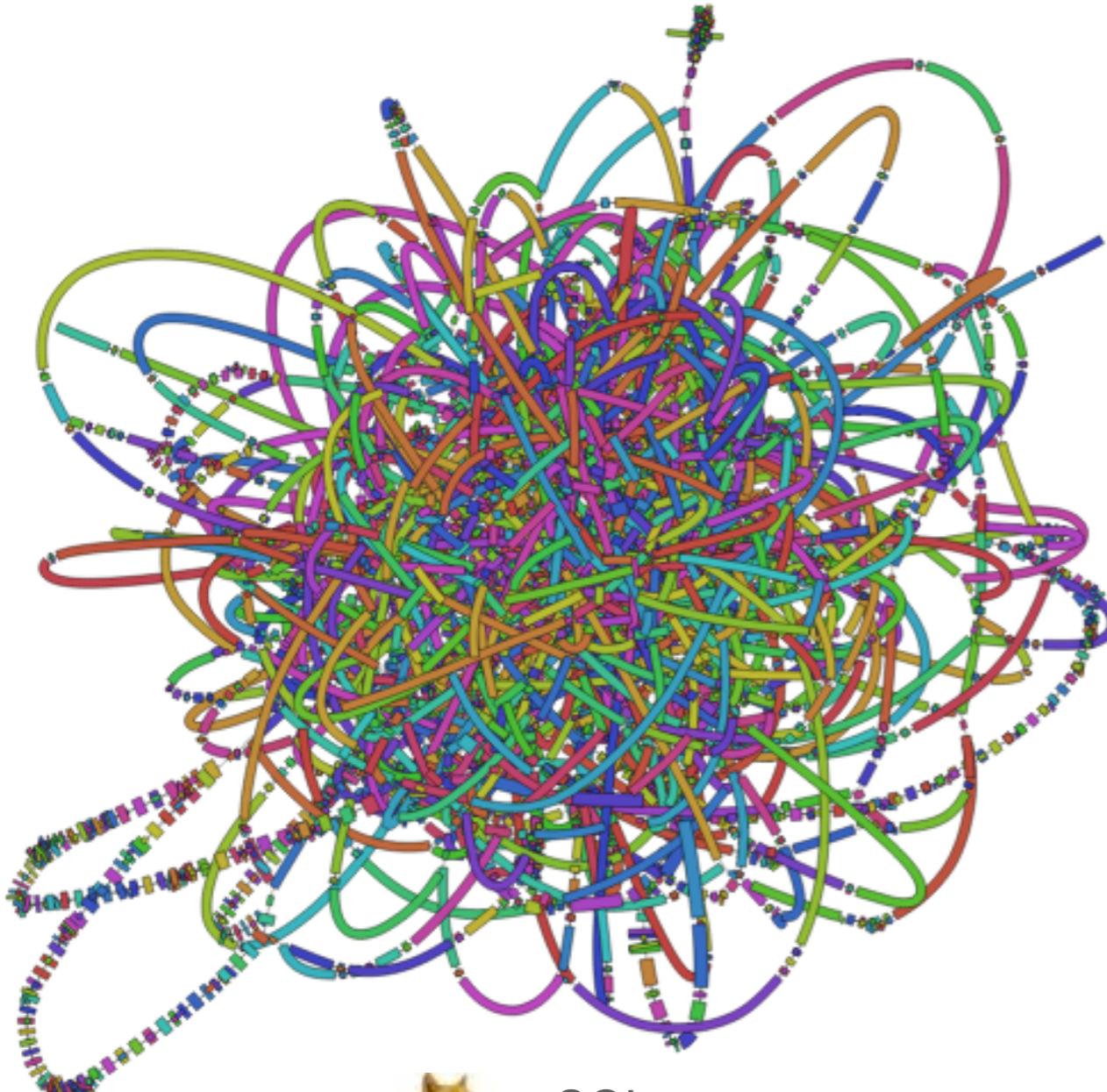
### Assembler

### Overlap

### Contigs

# NoSQL

## Ejemplo genómica



- 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<sup>1,2</sup>, Elena Buena-Atienza<sup>3,4</sup>, Stephan Ossowski<sup>3,4</sup> and Marie-France Sagot<sup>1,2</sup>

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:

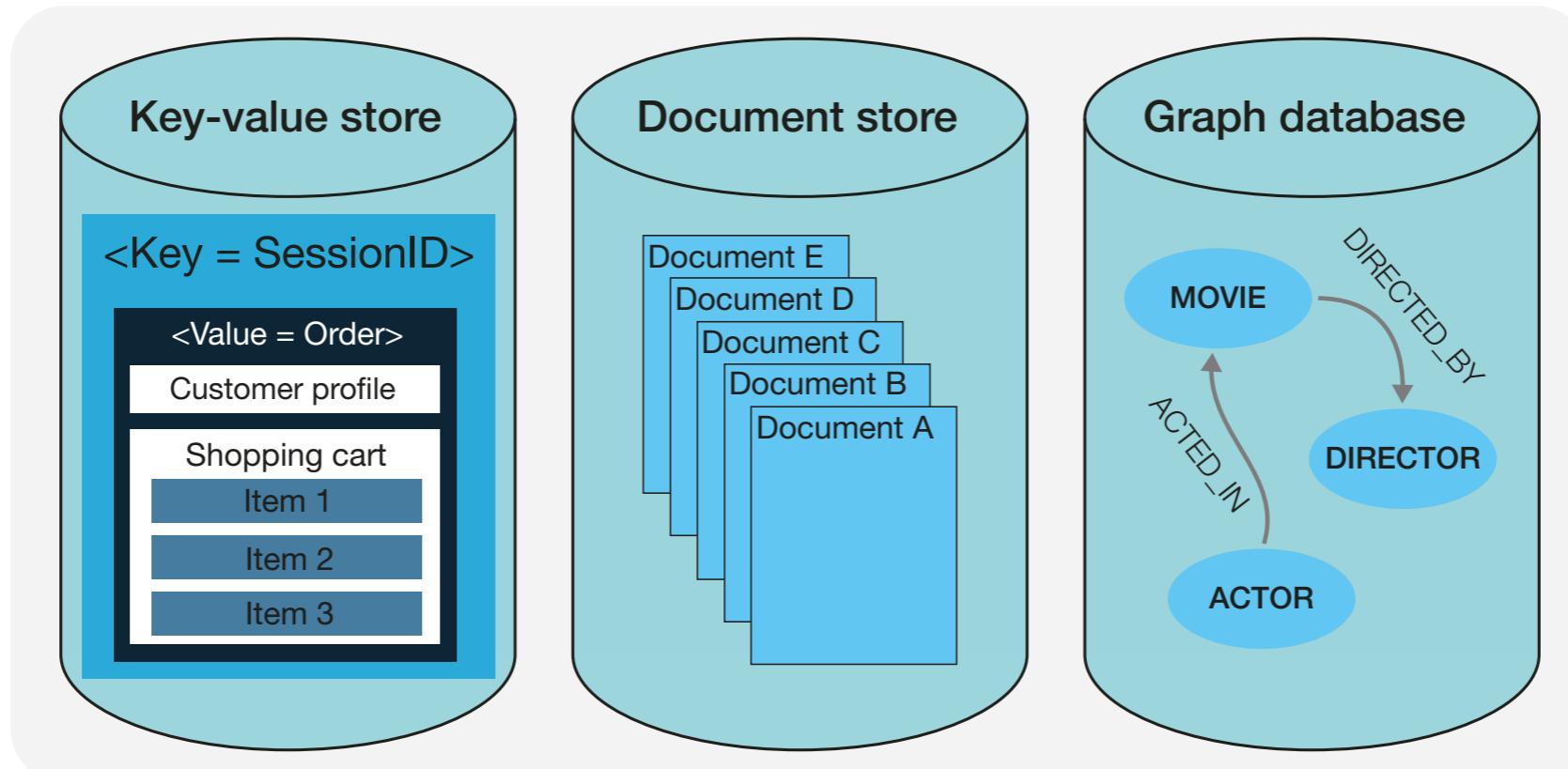
|                            | <b>SQL</b>   | <b>NoSQL</b>  |
|----------------------------|--|---|
| <b>Modelo de datos</b>     | Relaciones<br>Tuplas<br>Atributos<br>Dominios<br>Normalización | Documentos<br>Grafos<br>clave/valor                     |
| <b>Modelo de consultas</b> | SQL<br>Algebra relacional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

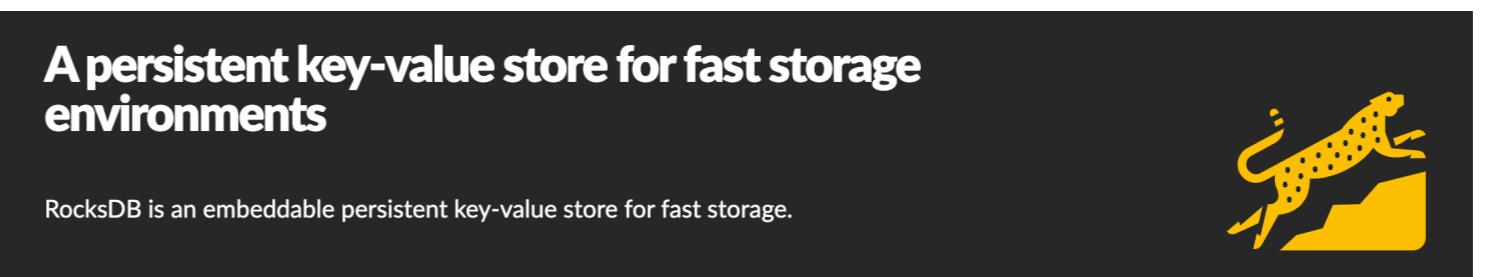


DynamoDB  
Azure Table Storage  
Riak  
Redis  
Aerospike  
FoundationDB  
LevelDB  
Berkeley DB  
Oracle NoSQL Database  
GenieDb  
BangDB  
Chordless  
Scalaris  
Tokyo Cabinet/Tyrant  
Scalien  
Voldemort  
Dynomite  
KAI  
MemcacheDB  
Faircom C-Tree  
LSM  
KitaroDB  
HamsterDB  
STSdb  
TarantoolBox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

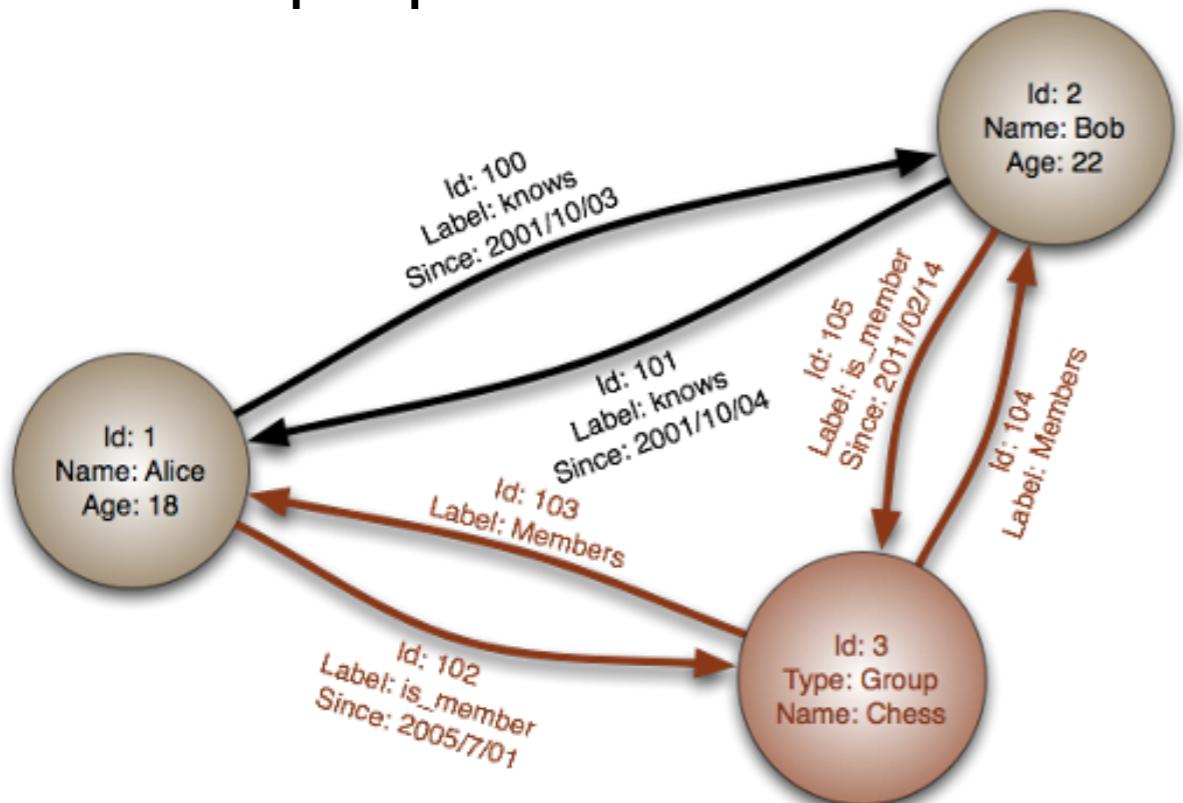


**RocksDB**

# 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

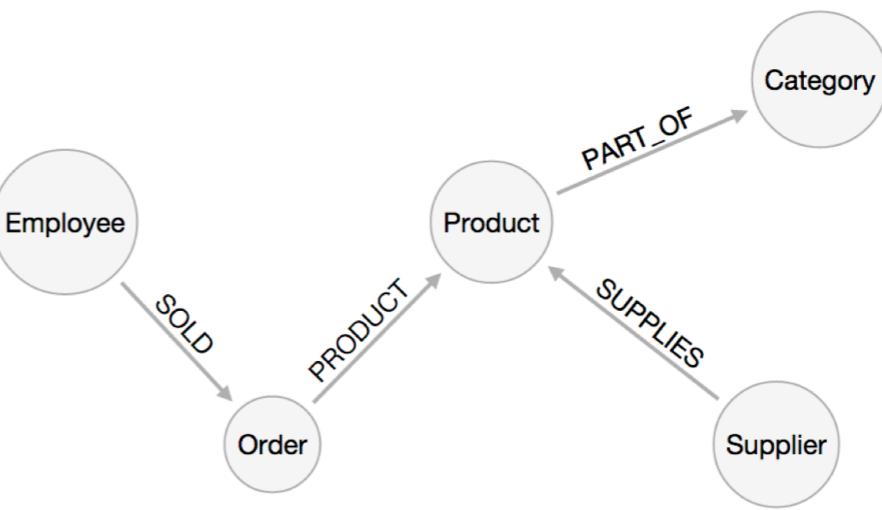
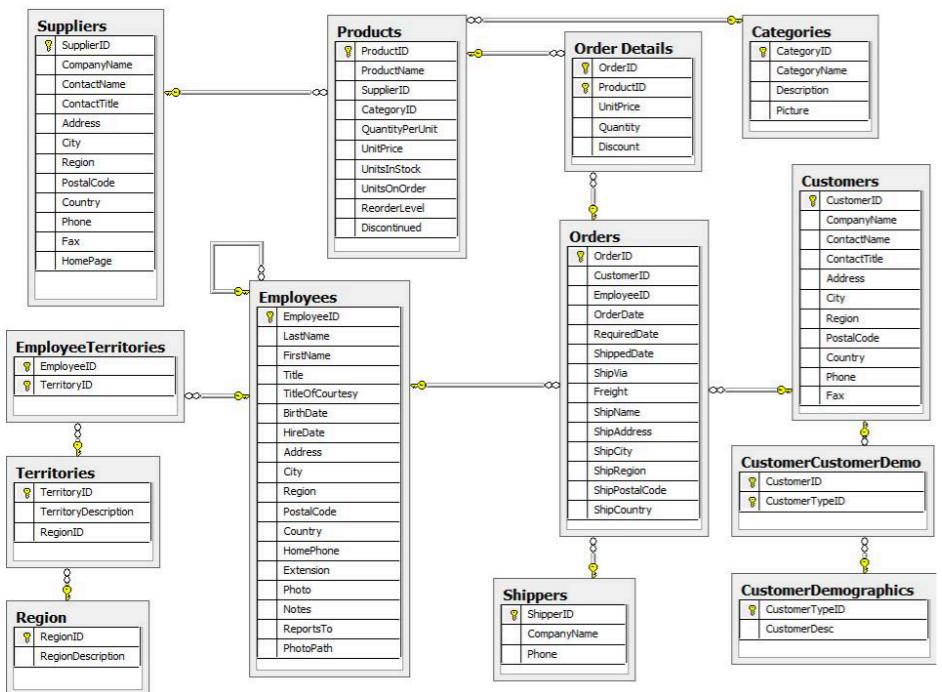
With proven [trillion+ entity performance](#), developers, data scientists, and enterprises rely on Neo4j as the top choice for high-performance, scalable analytics, intelligent app development, and advanced AI/ML pipelines.

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](http://www.nosql-database.org)  
[www.db-engines.com](http://www.db-engines.com)  
[www.wikipedia.com](http://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;
```

AllegroGraph  
ArangoDB  
Bigdata  
Betsy  
BrightstarDB  
DEX/Sparksee  
Execom IOG  
Fallen \*  
Filament  
FlockDB  
GraphBase  
Graphd  
Horton  
HyperGraphDB  
IBM System G Native Store  
InfiniteGraph  
InfoGrid  
jCoreDB Graph  
MapGraph  
Meronymy  
Neo4j  
Orly  
OpenLink virtuoso  
Oracle Spatial and Graph  
Oracle 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](http://www.nosql-database.org)  
[www.db-engines.com](http://www.db-engines.com)  
[www.wikipedia.com](http://www.wikipedia.com)

# 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](http://www.nosql-database.org)  
[www.db-engines.com](http://www.db-engines.com)  
[www.wikipedia.com](http://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.

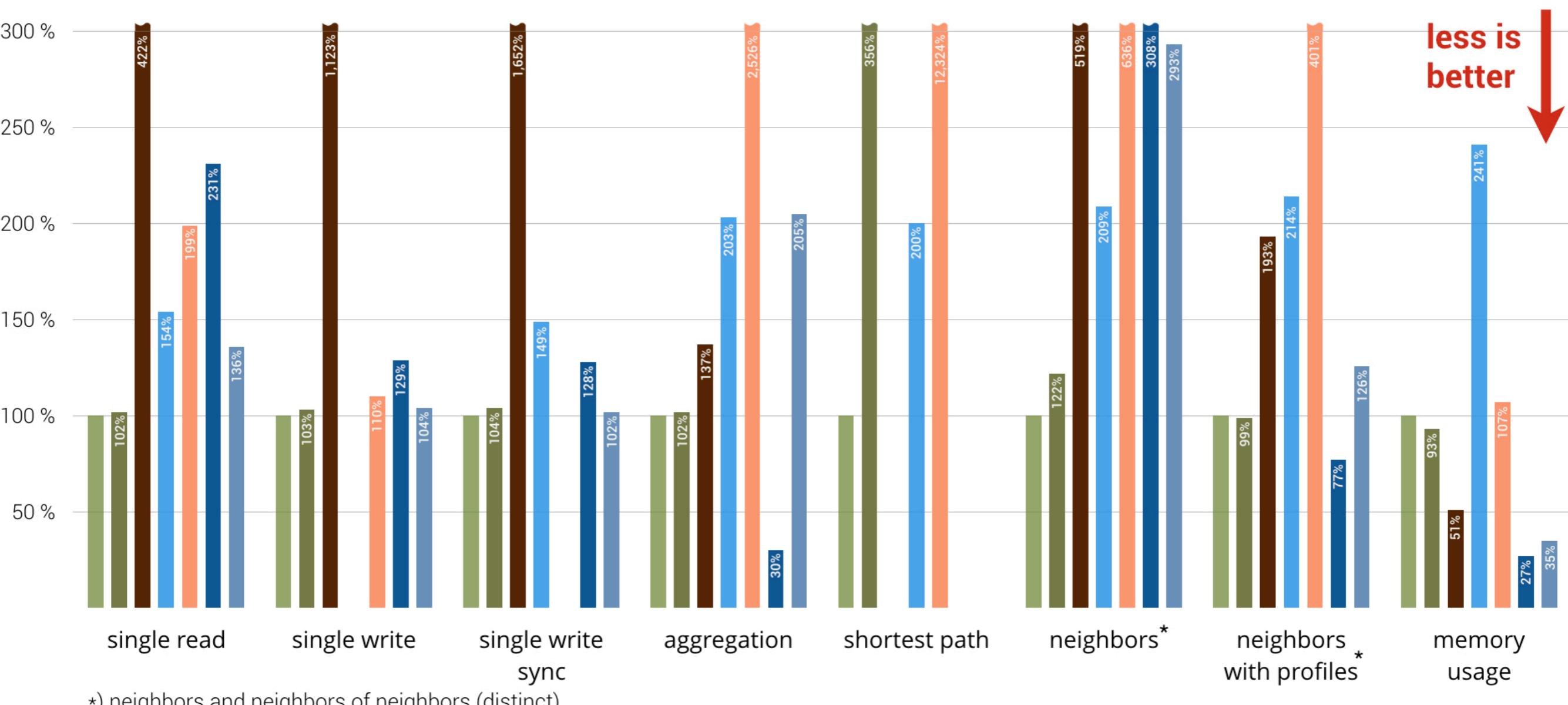
Las mediciones de rendimiento para ArangoDB, con RocksDB como motor de almacenamiento, definieron la línea de base (100 %) para las comparaciones. Los porcentajes más bajos indican un mayor rendimiento.

# 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<br>(s) | single write<br>(s) | single write<br>sync (s) | aggregation<br>(s) | shortest<br>(s) | neighbors<br>2nd (s) | neighbors<br>2nd data (s) | memory<br>(GB) |
|---------------------------------------|--------------------|---------------------|--------------------------|--------------------|-----------------|----------------------|---------------------------|----------------|
| <b>ArangoDB</b><br>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          |
| <b>ArangoDB</b><br>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          |
| <b>MongoDB</b><br>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           |
| <b>Neo4j</b><br>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          |
| <b>PostGres</b><br>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           |
| <b>PostGres</b><br>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           |
| <b>OrientDB</b><br>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          |

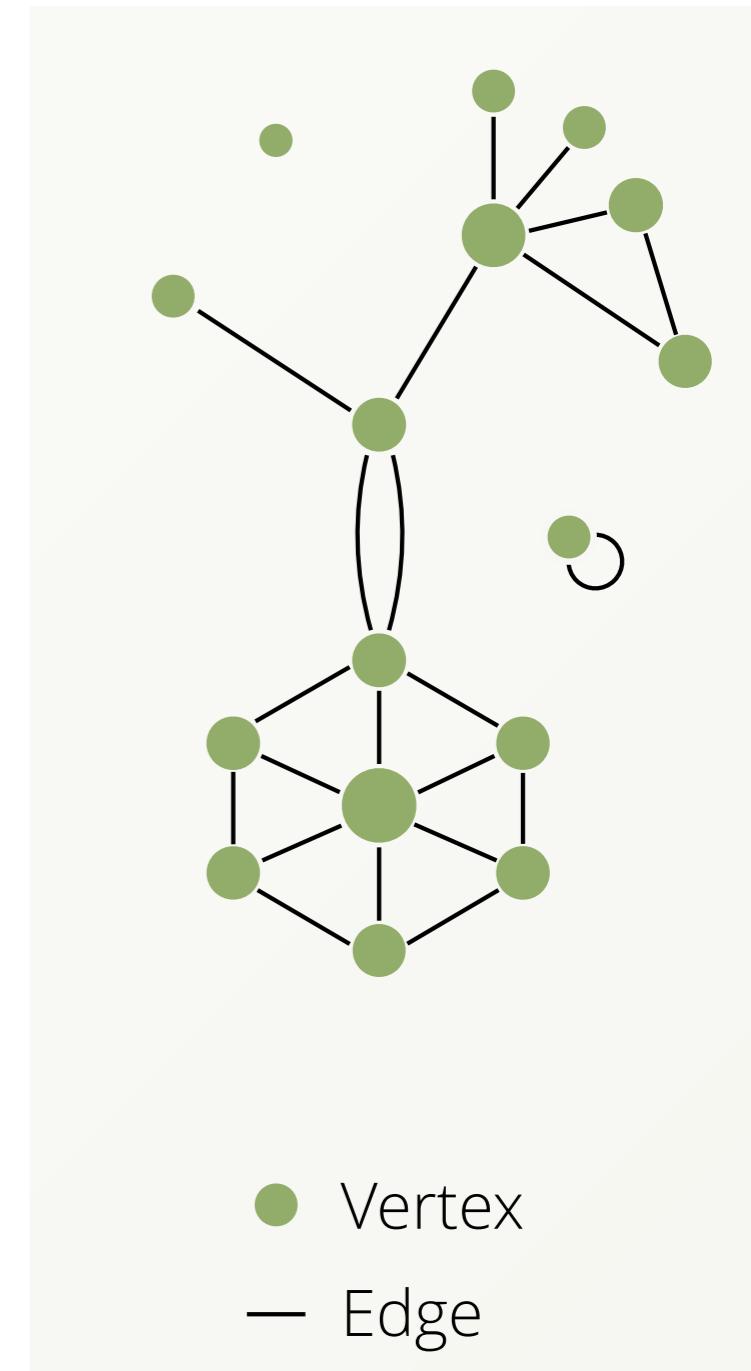
Conclusión: el rendimiento y la flexibilidad de una DB multimodelo es una ventaja clave del motor ArangoDB.

# Bases de datos en grafos.

# Grafos

## Que es un grafo?

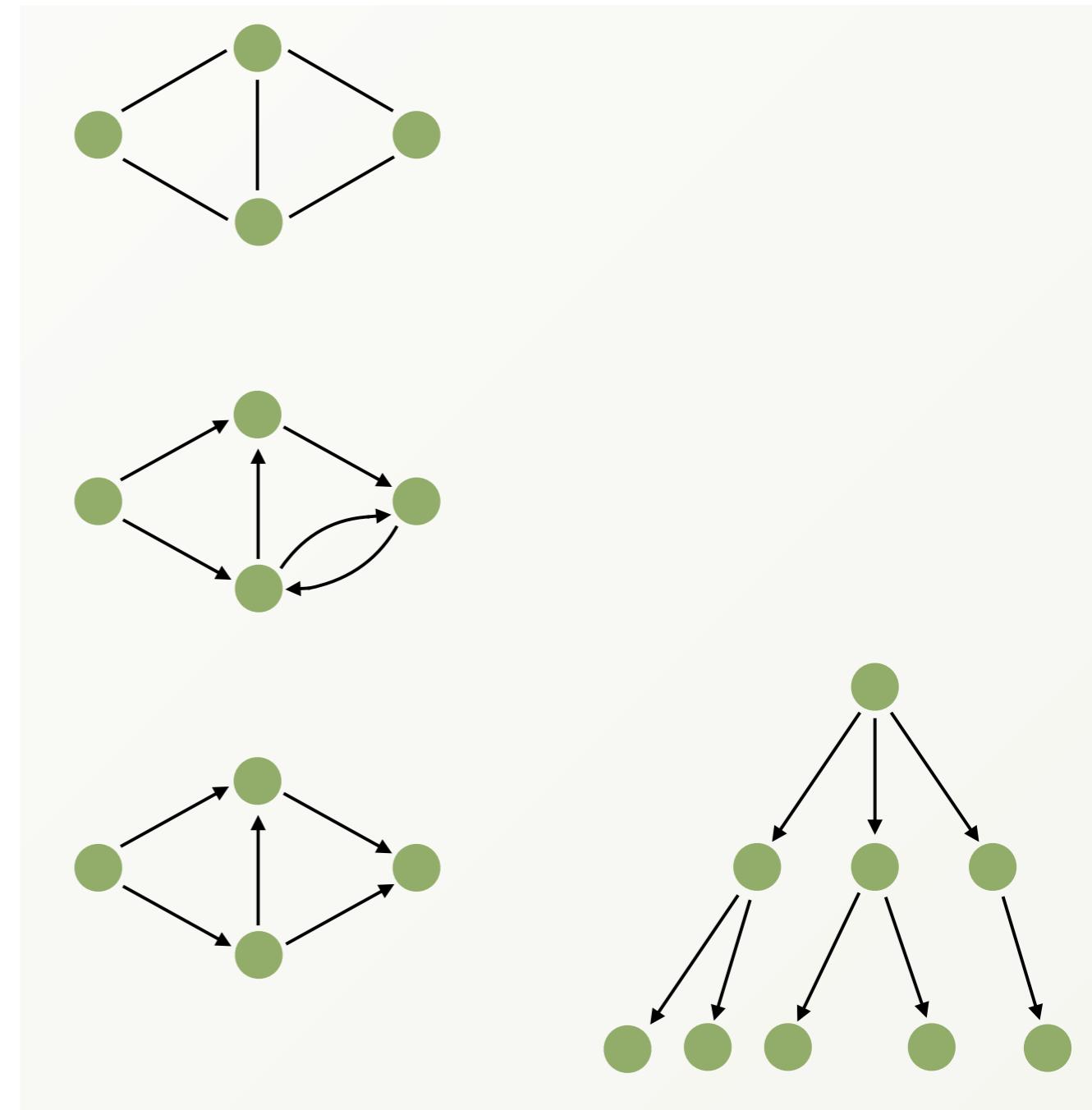
- Existen múltiples definiciones y tipos. Una breve reseña:
- En matemáticas discretas, un grafo se define como un **conjunto de vértices y aristas**. En computación se considera un **tipo de datos abstracto** que es muy útil para representar conexiones o relaciones.
  - Las estructuras de los sistemas de bases de datos relacionales (datos tabulares), son rígidas para expresar relaciones.
- Los términos **nodo** y **vértice** se usan indistintamente.
  - Por lo general, los vértices están conectados por aristas, formando un grafo.
  - Los vértices no tienen que estar conectados, pero también pueden estar conectados con más de un vértice a través de **múltiples aristas**.
  - También puede encontrar **vértices conectados entre sí**.



# Grafos

## Tipos de Grafos

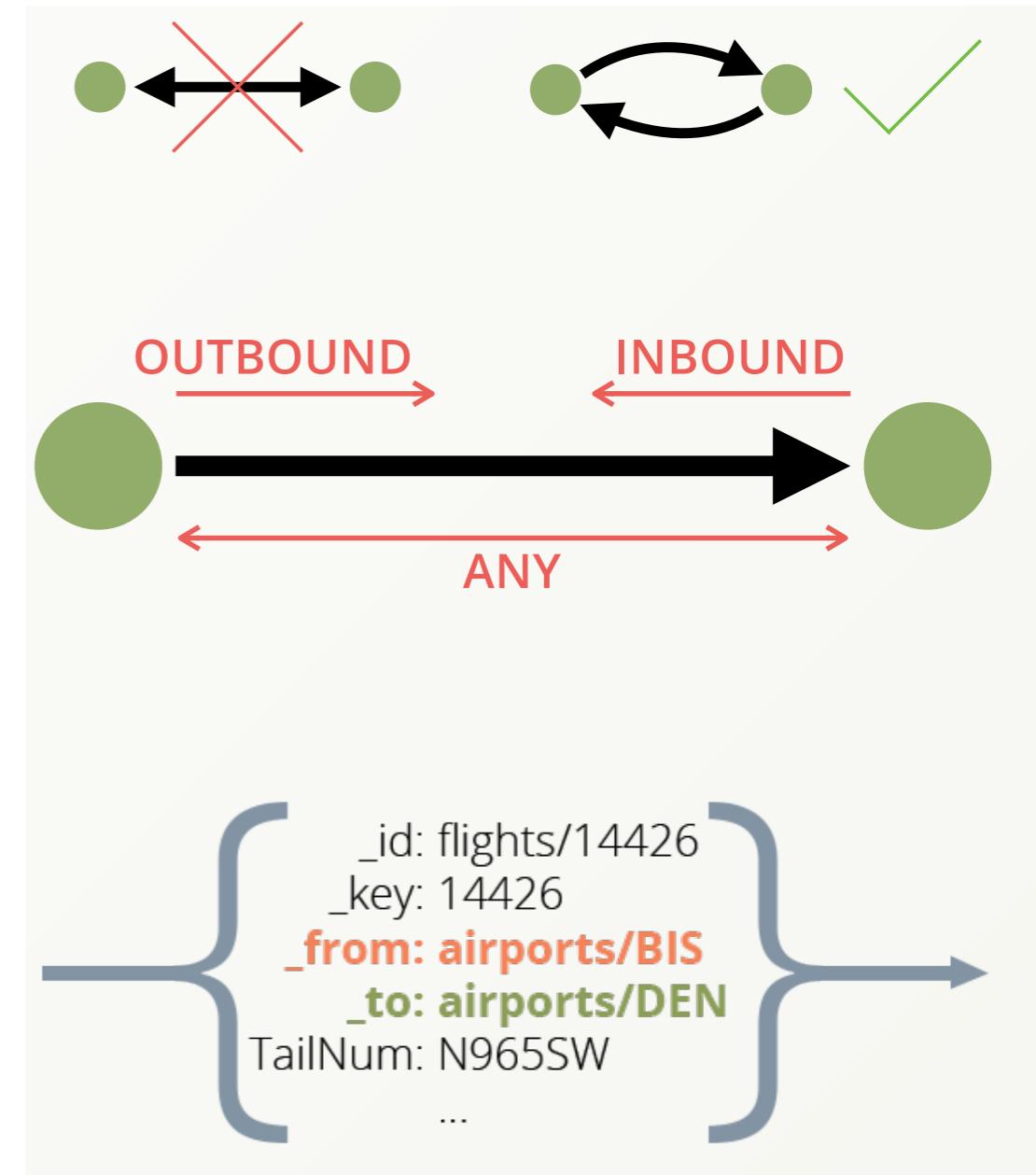
- **No dirigido:** las aristas conectan pares de nodos sin tener una noción de dirección
- **Dirigido:** las aristas (arco) tienen una dirección asociada.
- DAG – **Grafo acíclico dirigido:** las aristas tienen una **dirección y no hay ciclos**.
  - En el caso más simple, esto significa que si tenemos los vértices A y B y una arista de A a B, entonces no puede haber otra arista de B a A.
  - Un ejemplo para un **DAG** es un **árbol**.



# DB en Grafos

## ArangoDB

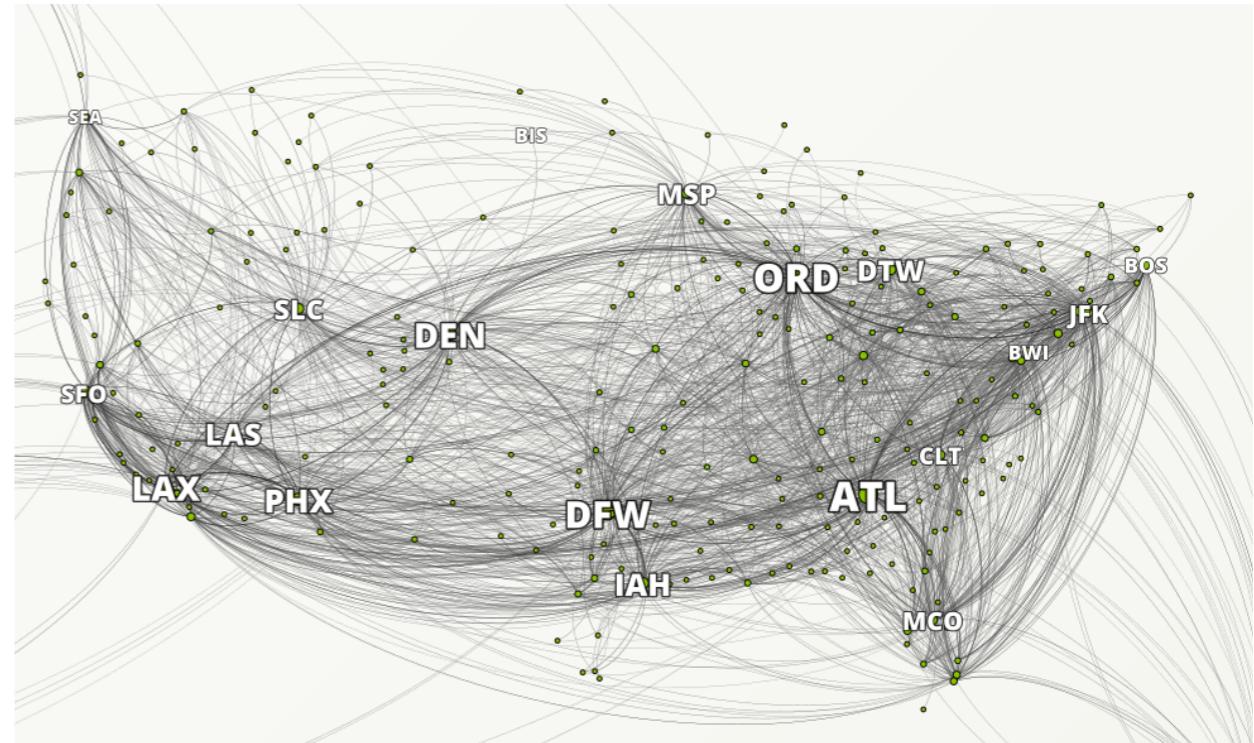
- En ArangoDB, cada arista tiene una sola dirección, no puede apuntar en ambos sentidos a la vez. Este modelo también se conoce como **grafo dirigido**.
- Pero la dirección se puede ignorar (seguir en CUALQUIER dirección - **ANY**) cuando nos movemos en el grafo, o seguir las aristas en dirección inversa (**INBOUND**) en lugar de ir en la dirección a la que realmente apuntan (**OUTBOUND**). Moverse en un grafo se llama recorrido.
- **ArangoDB** permite almacenar todo tipo de grafos en diferentes formas y tamaños, con y sin ciclos. **Podemos guardar uno o más aristas entre dos vértices o incluso con el mismo vértice**.
- Las **aristas** son **documentos JSON** completos, por lo tanto podemos almacenar tanta información como deseemos/necesitemos.



# DB en grafos

## Aeropuertos y vuelos

- Aeropuertos : 3,375
- Vuelos : 286,463
- Consultas:
  - Listar todos los vuelos que salen de **JFK** (aeropuerto de Nueva York)
  - Listar todos los vuelos que aterrizan en **LAX** (aeropuerto de Los Ángeles) el 5 de enero.
  - ¿Cuál es la cantidad mínima de escalas para volar desde **BIS** (Aeropuerto Municipal de Bismarck en Dakota del Norte) a **LAX**?

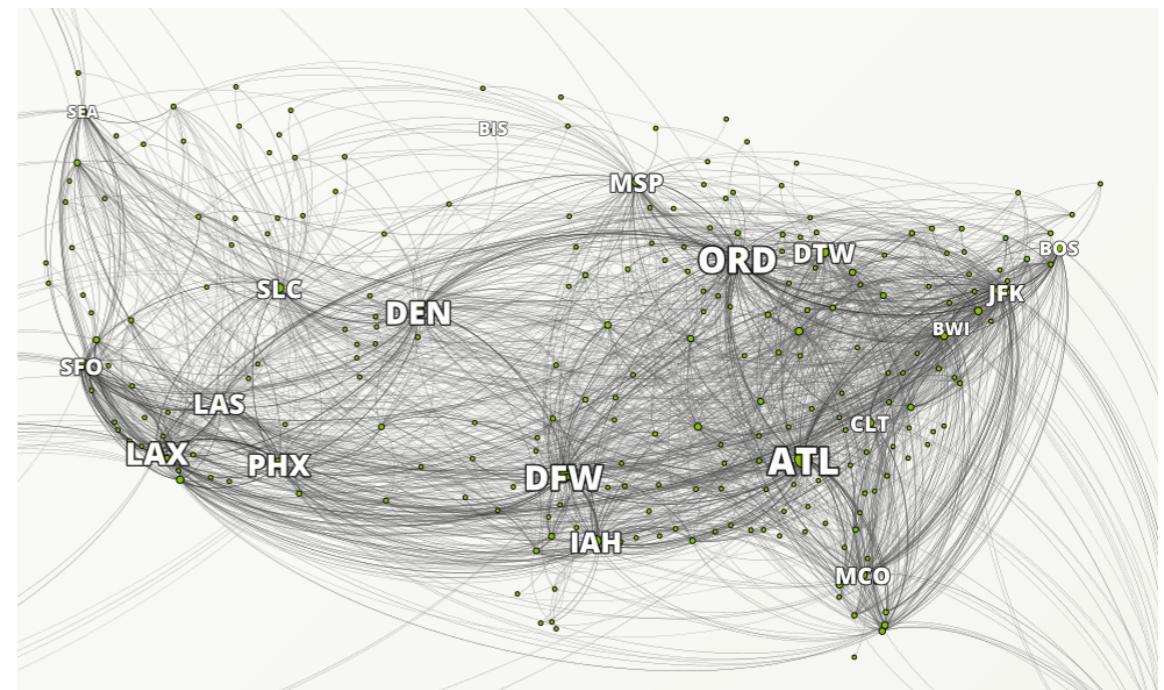


# Vuelos&Aeropuertos DB

## Documentos aeropuertos

- Atributos

- \_key : código abreviación
- \_id : nombre colección+"/"+\_key
- Lat y long : latitud y longitud
- Vip : ¿Aeropuerto con salón premium?



## Aeropuertos

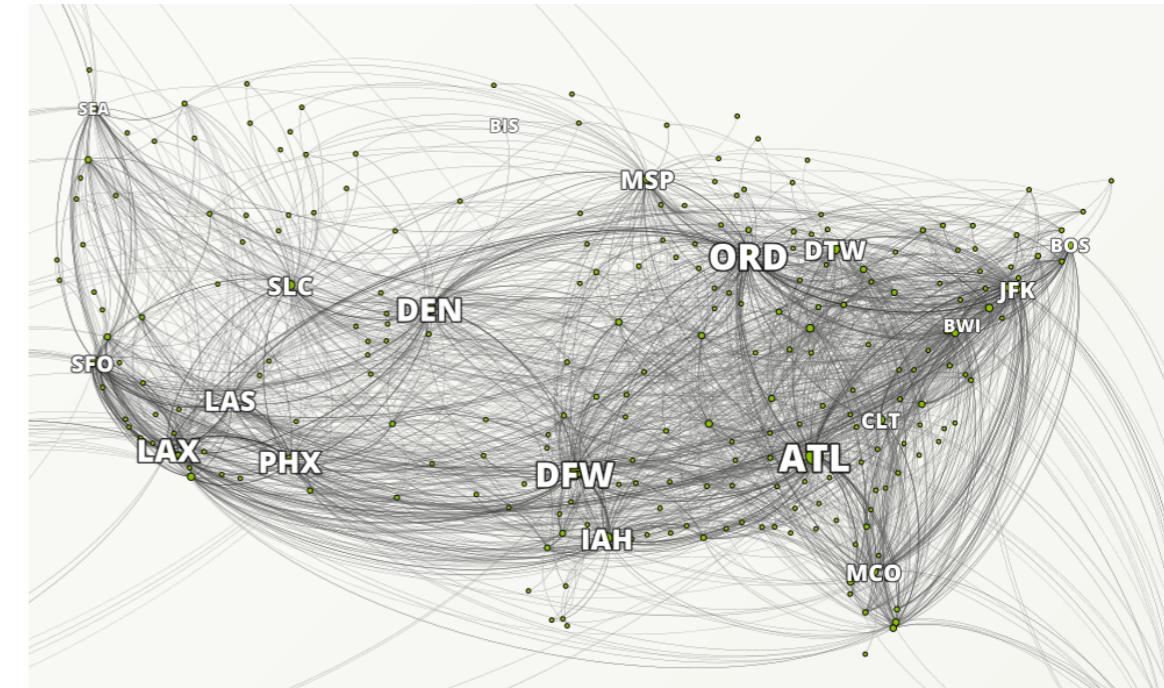
| key | name                 | city             | state | country | lat         | long         | vip   |
|-----|----------------------|------------------|-------|---------|-------------|--------------|-------|
| 00M | Thigpen              | Bay Springs      | MS    | USA     | 31.95376472 | -89.23450472 | FALSE |
| 00R | Livingston Municipal | Livingston       | TX    | USA     | 30.68586111 | -95.01792778 | FALSE |
| 00V | Meadow Lake          | Colorado Springs | CO    | USA     | 38.94574889 | -104.5698933 | FALSE |
| 01G | Perry-Warsaw         | Perry            | NY    | USA     | 42.74134667 | -78.05208056 | FALSE |
| 01J | Hilliard Airpark     | Hilliard         | FL    | USA     | 30.6880125  | -81.90594389 | FALSE |
| 01M | Tishomingo County    | Belmont          | MS    | USA     | 34.49166667 | -88.20111111 | FALSE |
| 02A | Gragg-Wade           | Clanton          | AL    | USA     | 32.85048667 | -86.61145333 | FALSE |
| 02C | Capitol              | Brookfield       | WI    | USA     | 43.08751    | -88.17786917 | FALSE |
| 02G | Columbiana County    | East Liverpool   | OH    | USA     | 40.67331278 | -80.64140639 | FALSE |

# Vuelos&Aeropuertos DB

## Documentos vuelos

- Atributos

- `_from` : aeropuerto de origen (id\_aeropuerto)
- `_to` : aeropuerto destino (id\_aeropuerto)
- Year, Month, Day, DayofWeek.
- DepTime : Hora de salida
- ArrTime : Hora de llegada
- FlightNum: Numero de vuelo
- TailNum : Numero de Avión.
- UniqueCarrier : Código de operador.
- Distance : Distancia de vuelo en millas.



## Vuelos

| <code>_from</code> | <code>_to</code> | Year | Month | Day | DayOfWeek | DepTime | ArrTime | UniqueCarrier | FlightNum | TailNum | Distance |
|--------------------|------------------|------|-------|-----|-----------|---------|---------|---------------|-----------|---------|----------|
| airports/ATL       | airports/CHS     | 2008 | 1     | 1   | 2         | 2       | 57      | FL            | 579       | N937AT  | 259      |
| airports/CLE       | airports/SAT     | 2008 | 1     | 1   | 2         | 3       | 230     | XE            | 2895      | N14158  | 1241     |
| airports/IAD       | airports/CLE     | 2008 | 1     | 1   | 2         | 5       | 132     | YV            | 7185      | N592ML  | 288      |
| airports/JFK       | airports/PBI     | 2008 | 1     | 1   | 2         | 8       | 332     | B6            | 859       | N505JB  | 1028     |
| airports/CVG       | airports/MHT     | 2008 | 1     | 1   | 2         | 9       | 215     | OH            | 5169      | N669CA  | 741      |
| airports/JFK       | airports/SFO     | 2008 | 1     | 1   | 2         | 11      | 327     | UA            | 9         | N555UA  | 2586     |
| airports/MIA       | airports/TPA     | 2008 | 1     | 1   | 2         | 14      | 105     | AA            | 1831      | N3CHAA  | 204      |
| airports/CVG       | airports/GSO     | 2008 | 1     | 1   | 2         | 25      | 148     | OH            | 5448      | N398CA  | 330      |
| airports/FLL       | airports/JFK     | 2008 | 1     | 1   | 2         | 26      | 250     | B6            | 878       | N656JB  | 1069     |

# Vuelos&Aeropuertos DB

## Ejemplos de Documentos en JSON

### Aeropuertos

```
{  
  "_key": "JFK",  
  "_id": "airports/JFK",  
  "_rev": "_Y0008KG-_T",  
  "name": "John F Kennedy Intl",  
  "city": "New York",  
  "state": "NY",  
  "country": "USA",  
  "lat": 40.63975111,  
  "long": -73.77892556,  
  "vip": true  
}
```

```
{  
  "_key": "BIS",  
  "_id": "airports/BIS",  
  "_rev": "_Y0SrLBe--r",  
  "name": "Bismarck Municipal",  
  "city": "Bismarck",  
  "state": "ND",  
  "country": "USA",  
  "lat": 46.77411111,  
  "long": -100.7467222,  
  "vip": false  
}
```

### Vuelos

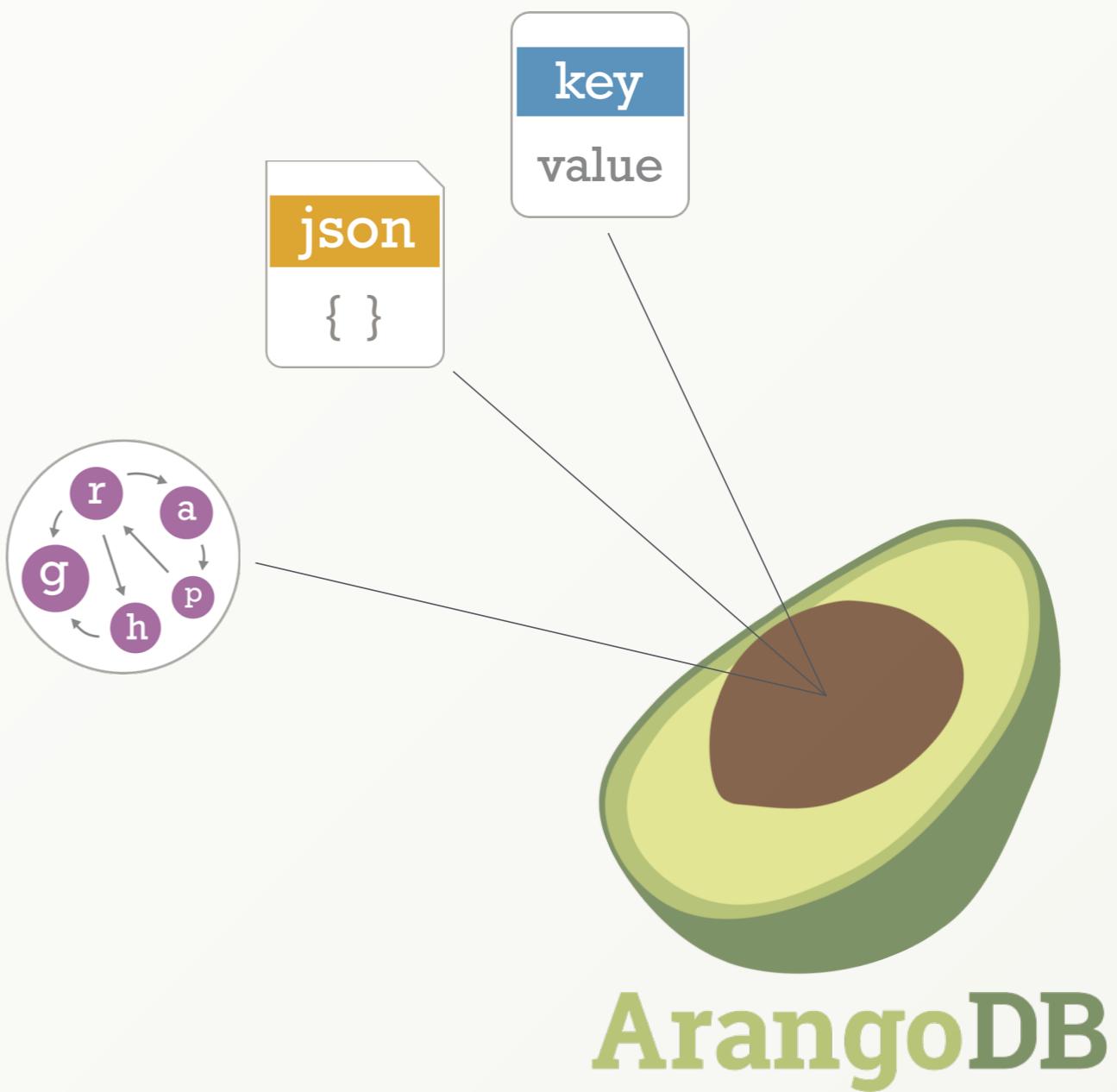
```
{  
  "_key": "25471",  
  "_id": "flights/25471",  
  "_from": "airports/BIS",  
  "_to": "airports/MSP",  
  "_rev": "_Y008JXG--f",  
  "Year": 2008,  
  "Month": 1,  
  "Day": 2,  
  "DayOfWeek": 3,  
  "DepTime": 1055,  
  "ArrTime": 1224,  
  "DepTimeUTC": "2008-01-02T16:55:00.000Z",  
  "ArrTimeUTC": "2008-01-02T18:24:00.000Z",  
  "UniqueCarrier": "9E",  
  "FlightNum": 5660,  
  "TailNum": "85069E",  
  "Distance": 386  
}
```

```
{  
  "_key": "71374",  
  "_id": "flights/71374",  
  "_from": "airports/JFK",  
  "_to": "airports/DCA",  
  "_rev": "_Y008LYG--N",  
  "Year": 2008,  
  "Month": 1,  
  "Day": 4,  
  "DayOfWeek": 5,  
  "DepTime": 1604,  
  "ArrTime": 1724,  
  "DepTimeUTC": "2008-01-04T21:04:00.000Z",  
  "ArrTimeUTC": "2008-01-04T22:24:00.000Z",  
  "UniqueCarrier": "MQ",  
  "FlightNum": 4755,  
  "TailNum": "N854AE",  
  "Distance": 213  
}
```

# ArangoDB

## Motor muti-modelo

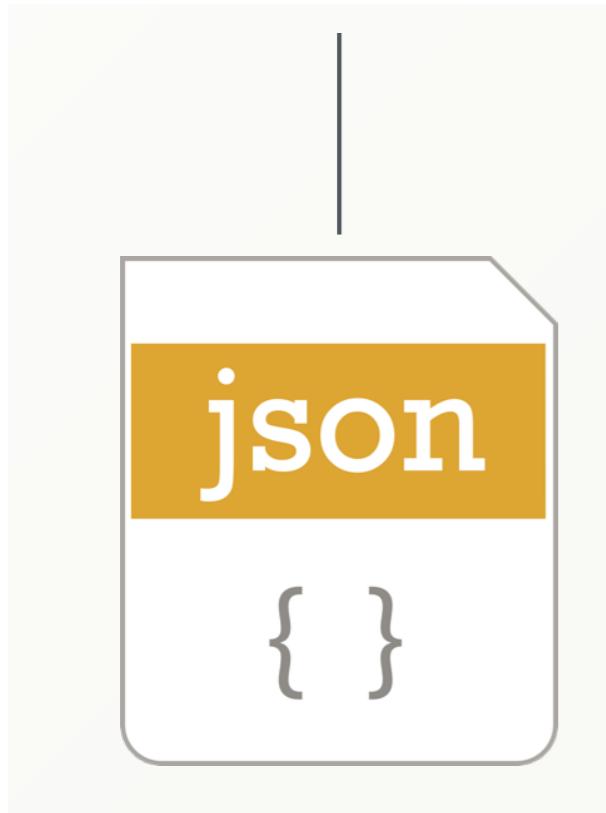
- Características únicas de AQL:
  - Posibilidad de combinar los **3 modelos de datos** en una sola consulta.
  - Podemos combinar uniones (joins), recorridos (traversal), filtros, operaciones geoespaciales y agregaciones en nuestras consultas



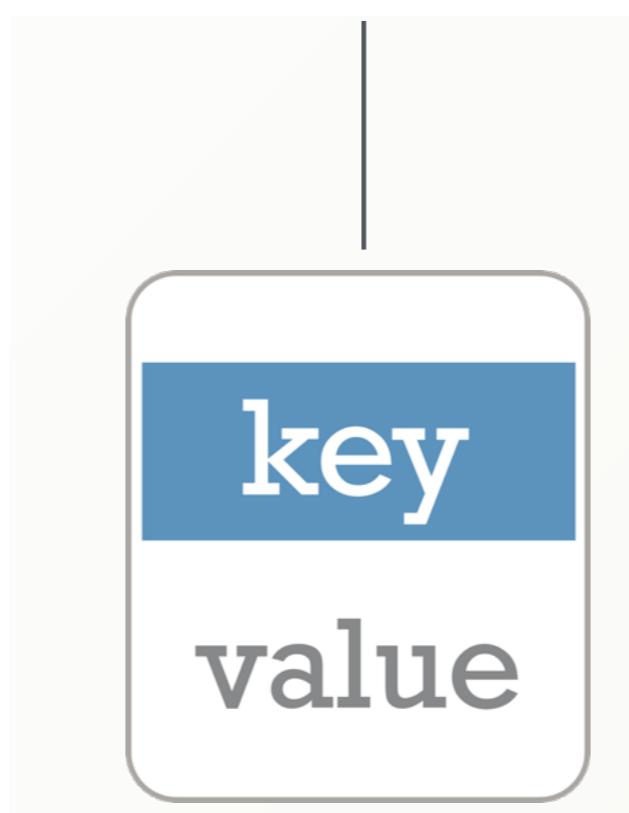
# Implementacion Multi-Modelo

## ¿Cómo es implementado el multi-modelo en Arango?

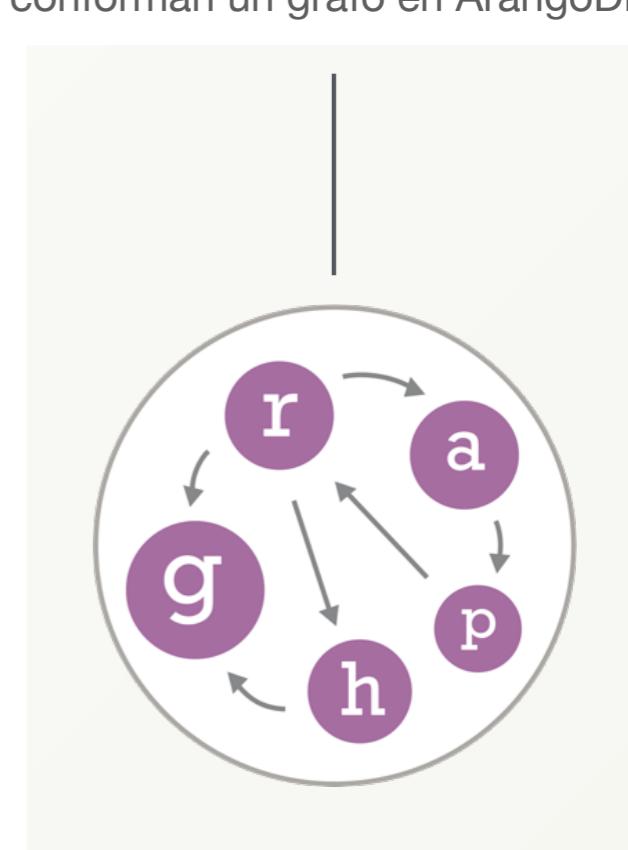
ArangoDB es un motor de datos orientado a documentos que utiliza **claves primarias**



Si almacenamos un **documento JSON** y lo tratamos como un valor de una clave primaria, entonces tenemos documento con clave/valor.

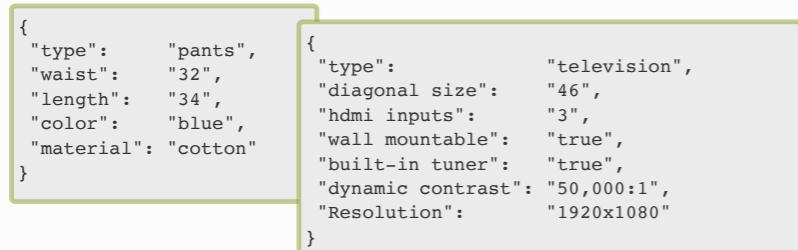


Los atributos especiales **\_from** y **\_to** en documentos de aristas que apuntan a otros documentos conforman un grafo en ArangoDB

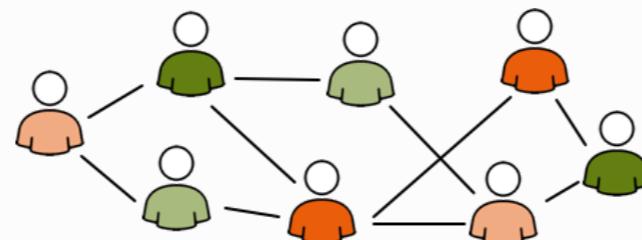


### MultiModelo Nativo de ArangoDB

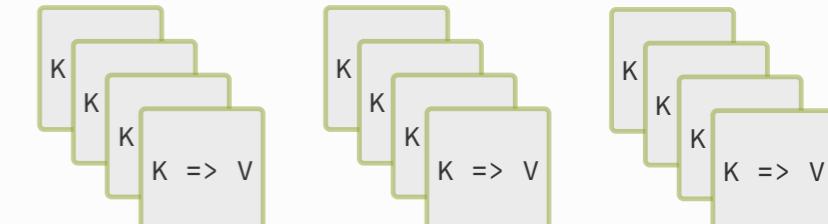
Documents - JSON



Graphs



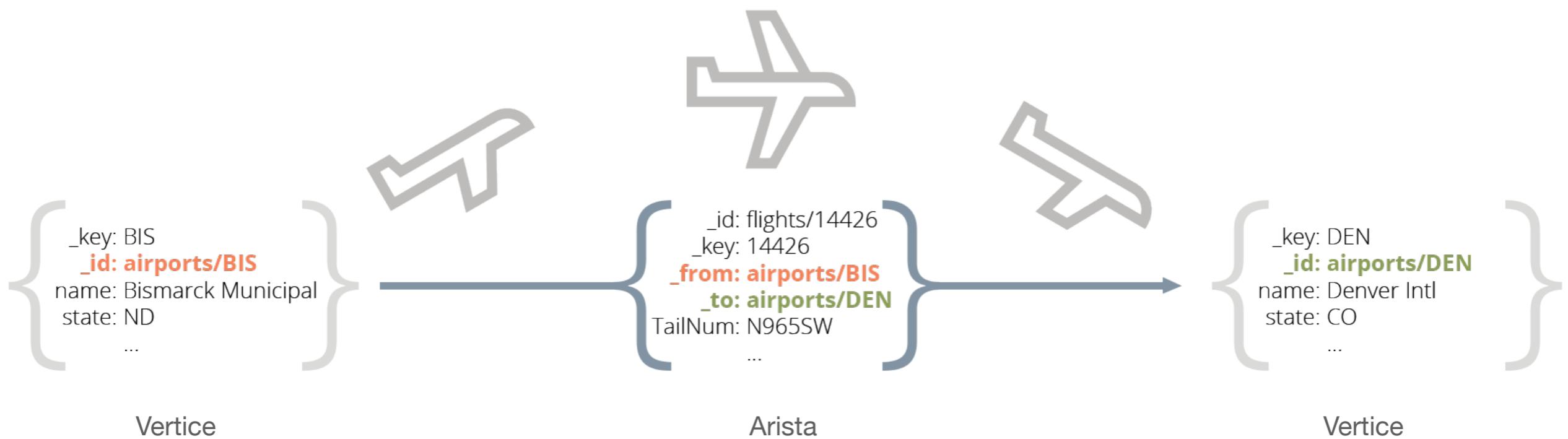
Key Values



# Implementacion Multi-Modelo

## ¿Cómo es implementado el multi-modelo en Arango?

- ArangoDB tiene una jerarquía de almacenamiento:
  - Puedemos crear diferentes bases de datos que pueden contener un número arbitrario de **colecciones**. Hay una base de datos predeterminada llamada `_system`.
  - Las **colecciones** pueden contener cantidades arbitrarias de documentos. Hay dos tipos de colección: **documentos** y **colecciones aristas**.
  - Los documentos se almacenan en formato JSON. Un documento es un objeto JSON en el nivel superior, cuyos nombres de atributo son strings y los valores pueden ser nulo, verdadero, falso, números, texto, matrices y objetos. También hay atributos del sistema (`_key`, `_id`, `_rev`, para aristas también `_from`, `_to`)

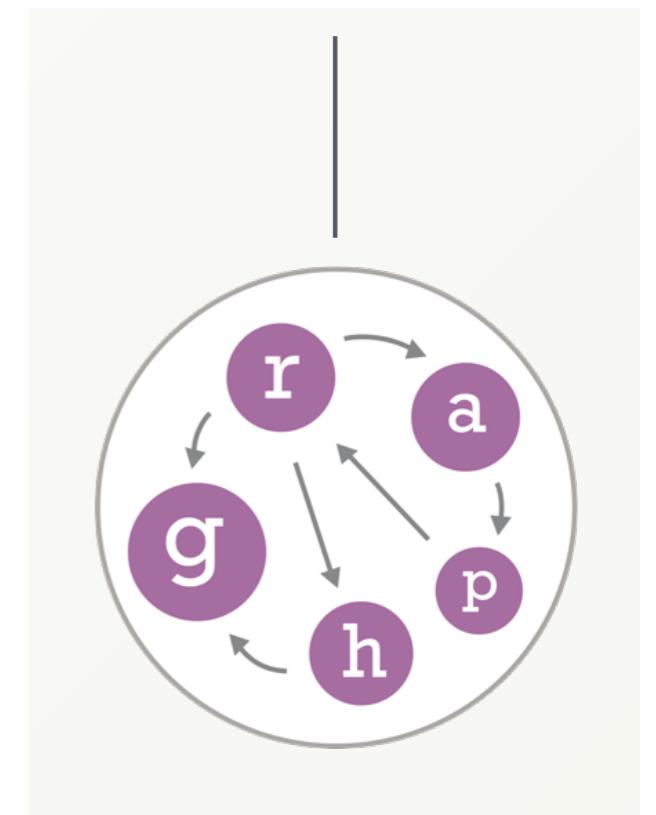


# ArangoDB

## Colecciones de aristas

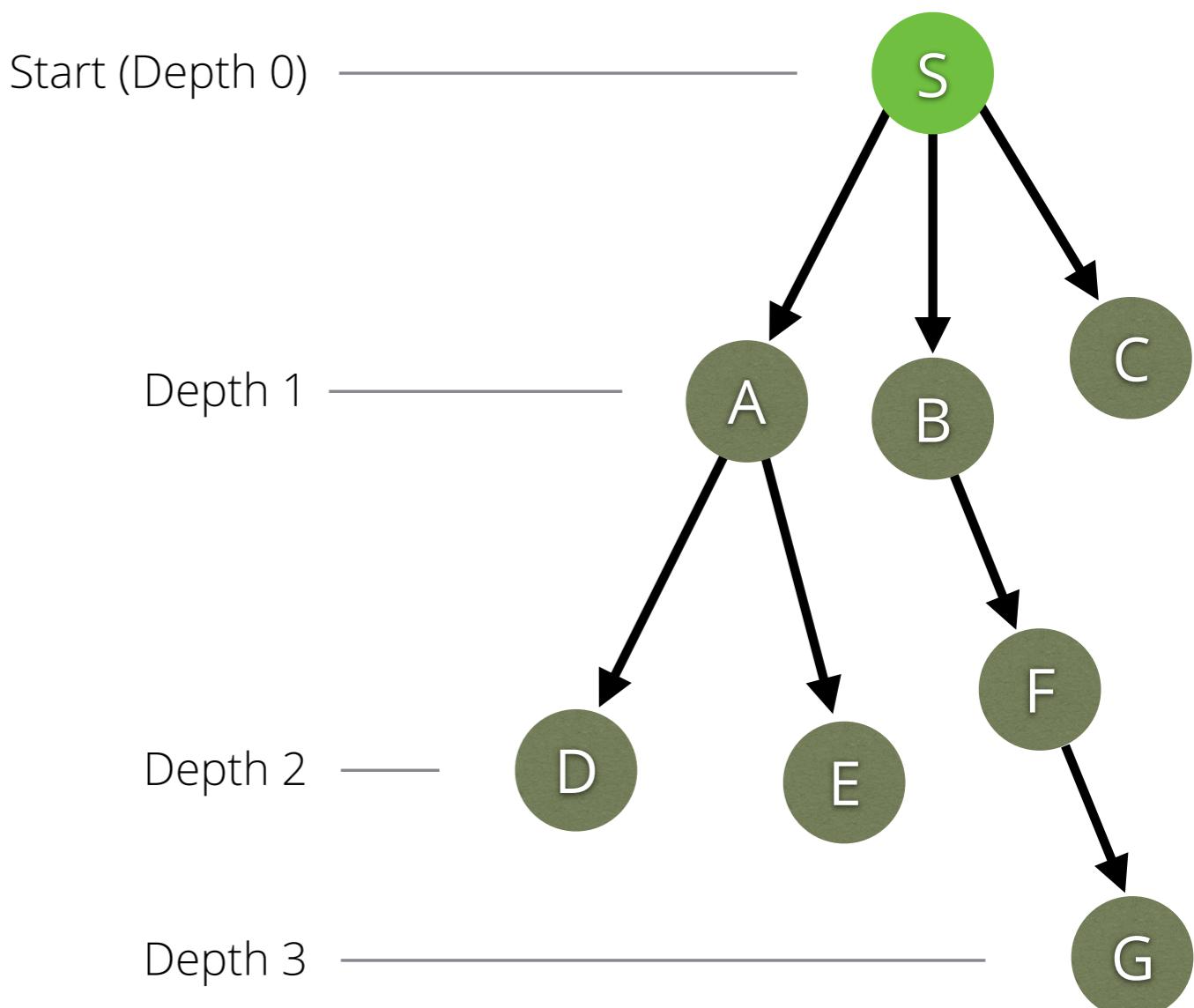
- En resumen:
- Lugar para representar/manipular relaciones
- Comparable con relaciones de N:M en sistemas SQL
- Documentos, pero con atributos especiales
  - **\_from**: \_id valor del vértice origen
  - **\_to**: \_id del vértice destino
- Índice de aristas incorporado para cada colección de aristas
- Elementos de construcción de grafos en ArangoDB.

Los atributos especiales **\_from** y **\_to** en documentos de aristas que apuntan a otros documentos conforman un grafo en ArangoDB



# Recorridos en ArangoDB

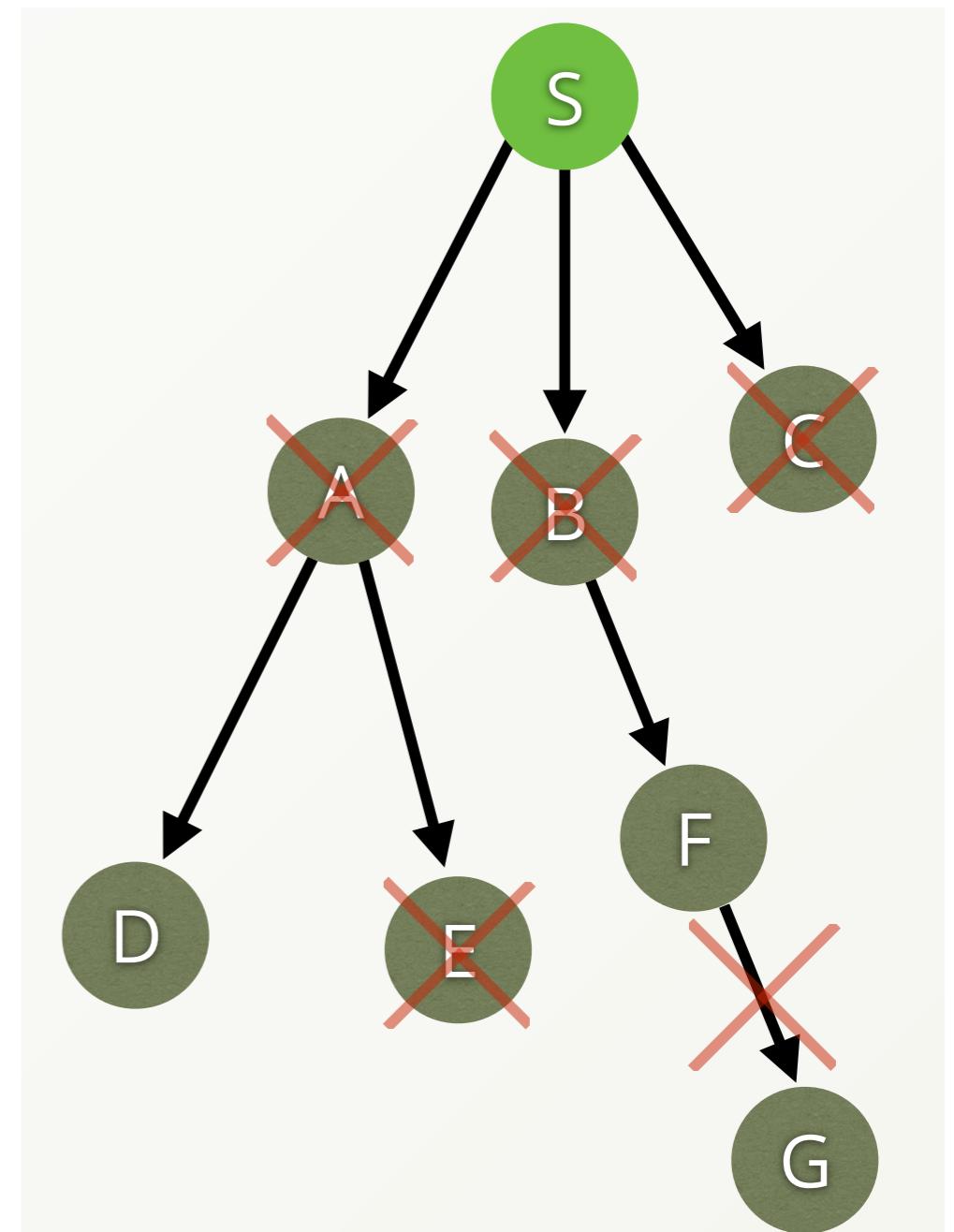
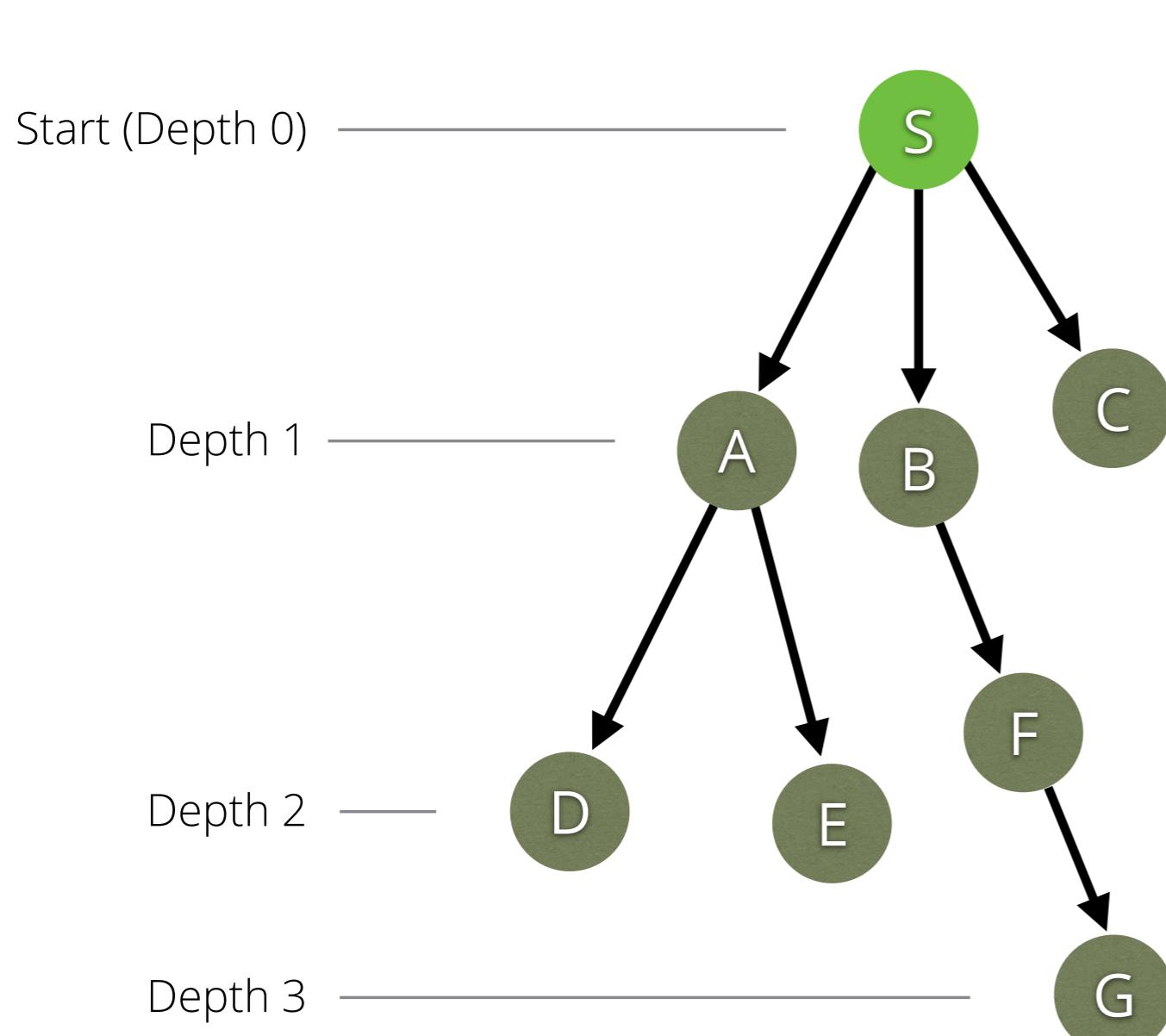
- Traversal significa **recorrer las aristas de un grafo** de ciertas maneras, opcionalmente con algunos filtros. En ArangoDB, esto se logra de manera muy eficiente mediante el índice de aristas.
- La cantidad de pasos que hay que dar en un recorrido se conoce como **profundidad de recorrido**.



# Recorridos en ArangoDB

## Ejemplos

- Un recorrido en dirección OUTBOUND con una profundidad mínima y máxima de 2.

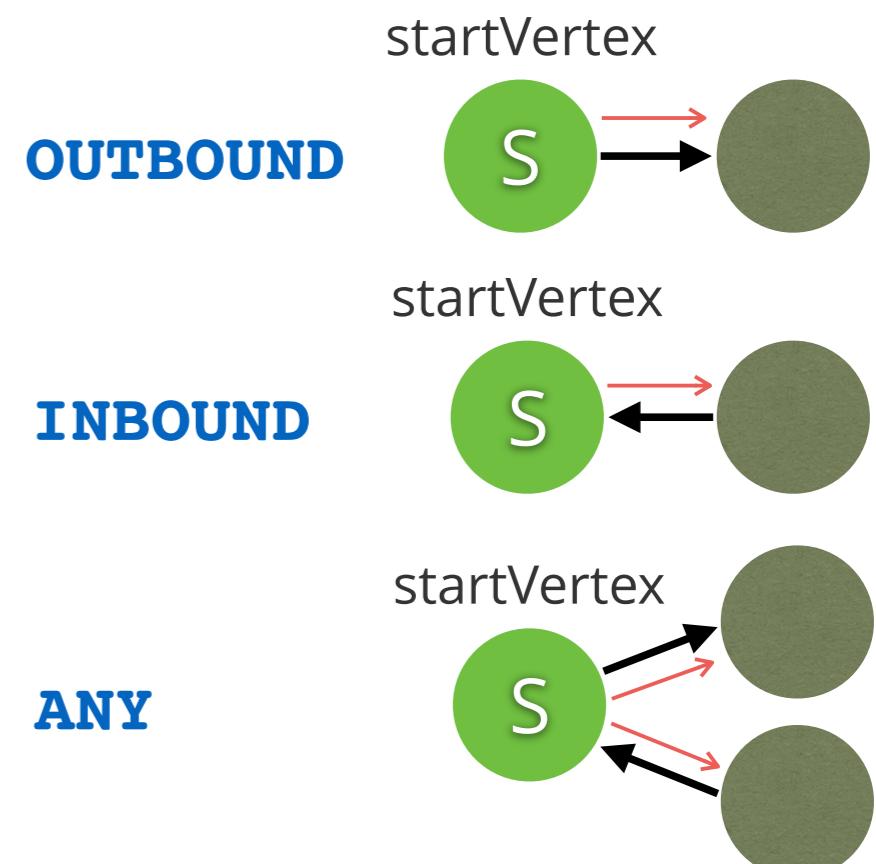


# Recorridos en ArangoDB

## AQL

```
FOR vertex[, edge[, path]]  
IN [min[..max]] OUTBOUND |  
INBOUND | ANY startVertex  
edgeCollection[, more...]
```

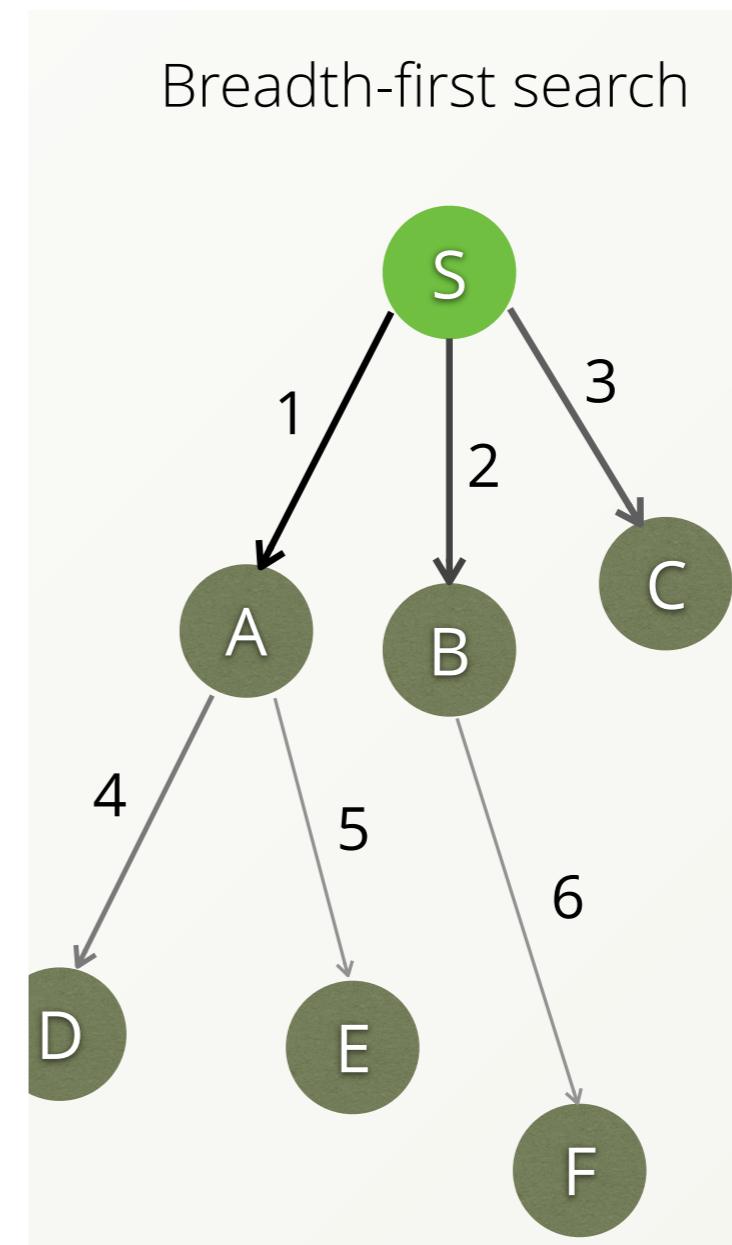
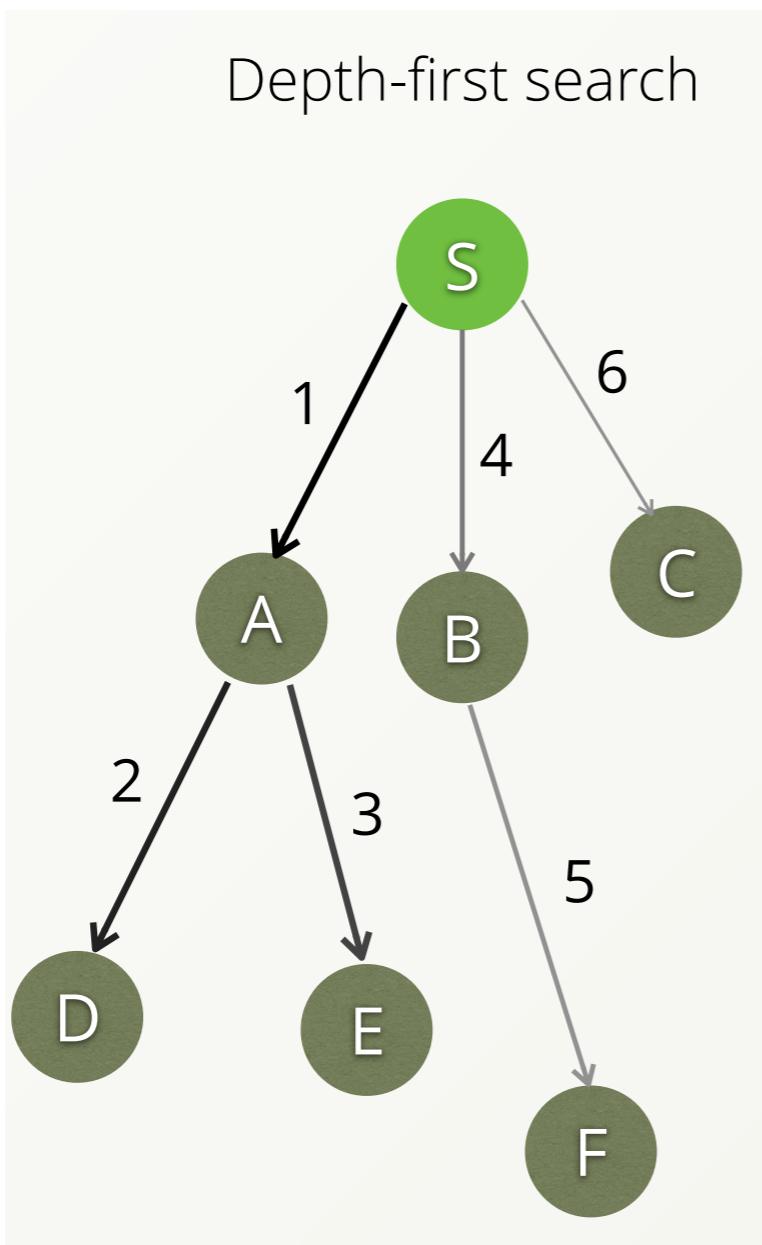
- **FOR** : vertices, aristas, caminos
- **IN** : define la profundidad minima y maxima.
- **OUTBOUND** : El recorrido sigue las aristas salientes
- **INBOUND** : El recorrido sigue las aristas salientes
- **ANY** : El recorrido sigue las aristas en cualquier dirección.



# Recorridos en ArangoDB

## Depth vs. Breadth-First Search

- **Depth** (defecto): Continúa hacia abajo por las aristas desde el vértice de inicio hasta el último vértice en ese camino o hasta alcanzar la profundidad de recorrido máxima, luego continua por los otros caminos.
- **Breadth** (opcional) : Sigue todos las aristas desde el vértice de inicio hasta el siguiente nivel, luego siga todos las aristas de sus vecinos por otro nivel y continúe este patrón hasta que no haya más aristas para seguir o se alcance la profundidad máxima.

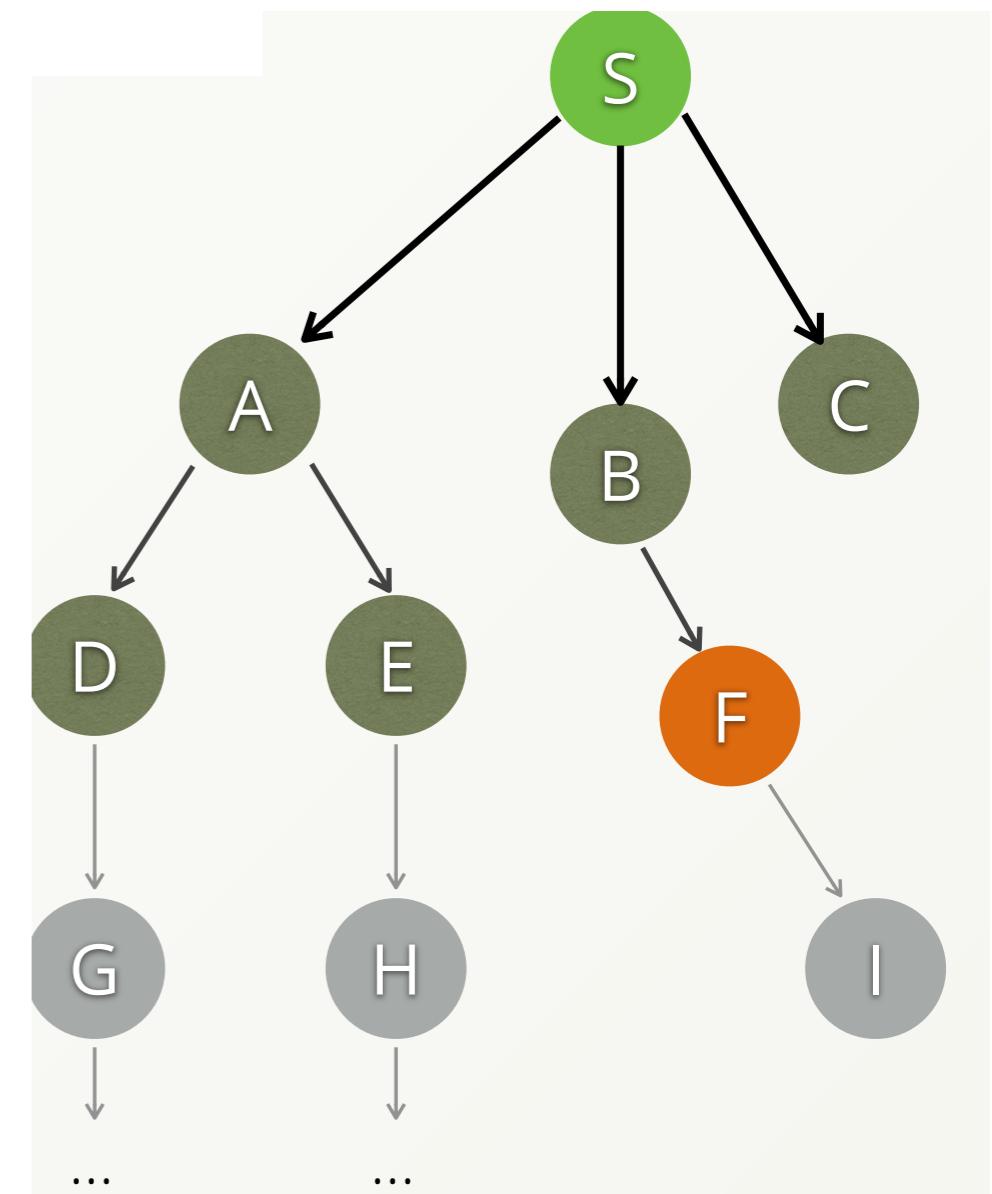


# Depth/Breath First search

## DFS o BFS?

- La búsquedas pueden ser significativamente más rápidas si se usan filtros y límites (profundidad máxima).
- Por ejemplo:
  - Recorrer un G desde S con profundidad de 1 .. 10
  - Encontrar un vértice (F) que cumpla algún criterio.
- DFS: visitaría primero A luego exploraría hasta profundidad 10 y volvería.
- BFS: encontraría F a profundidad 2 y no exploraría las demás profundidades.

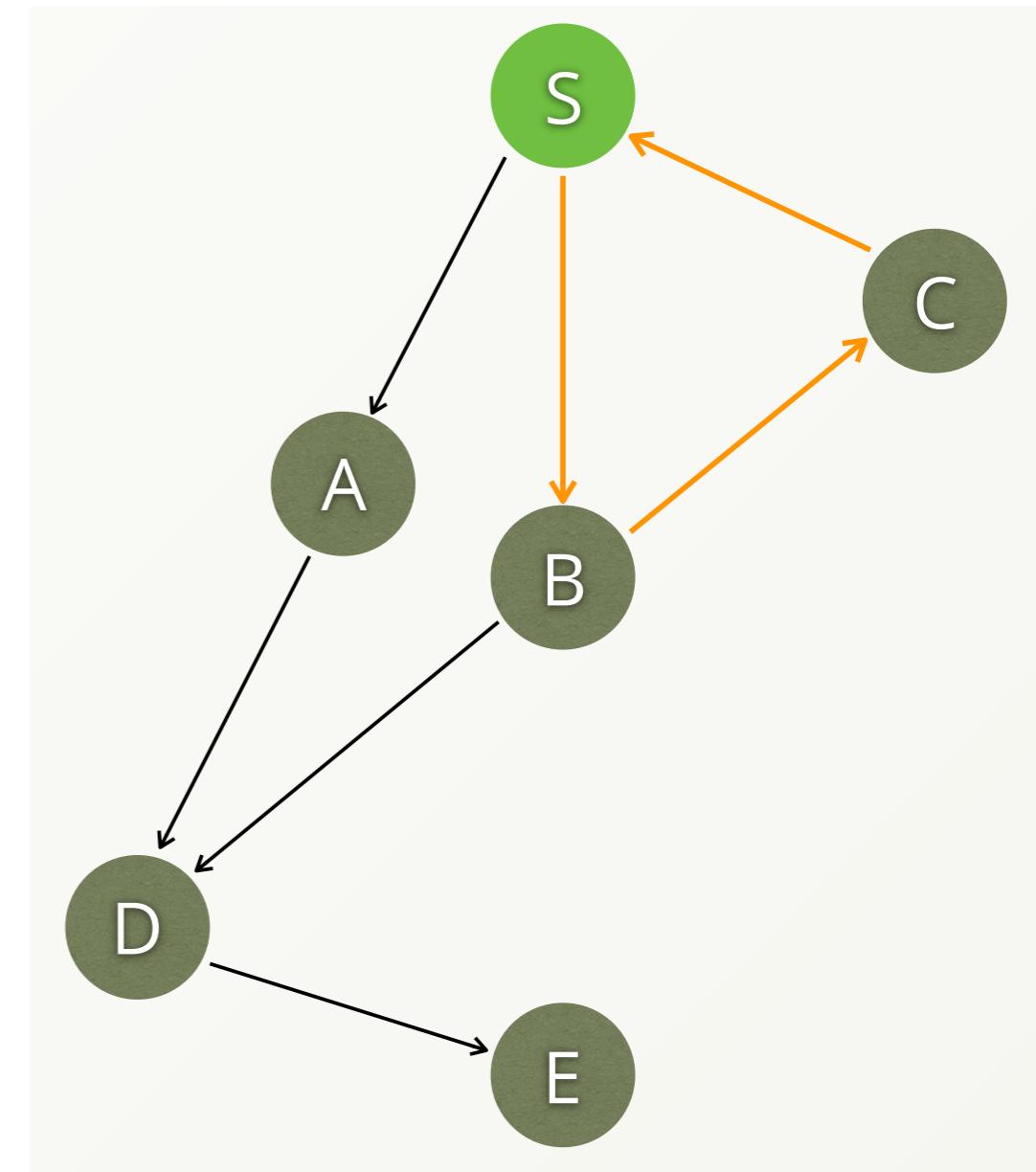
```
FOR v IN 1..10 OUTBOUND  
  'verts/S' edges OPTIONS  
  {bfs: true}  
  FILTER v._key == 'F'  
  LIMIT 1  
  RETURN v
```



# Depth/Breath First search

## Controlando recorridos

- Los Grafos pueden ser complejos.
  - Multiples rutas entre dos vértices
  - Ciclos
- Las aristas de un camino no pueden estar duplicadas.
- Se permiten vértices duplicados en una ruta a menos que el recorrido esté configurado de otra manera.



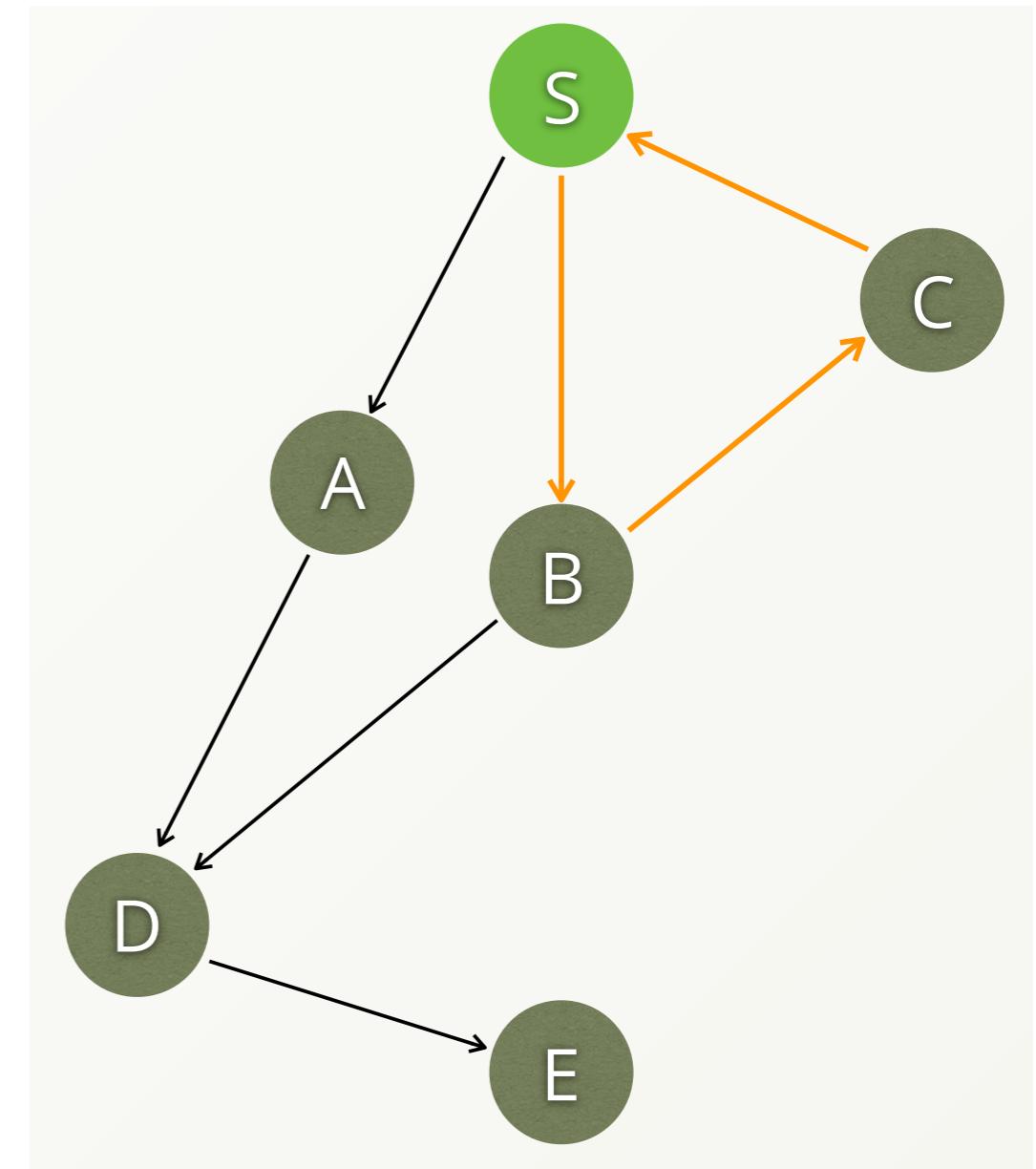
S → E

# Depth/Breath First search

## Controlando recorridos

```
FOR v, e, p IN 1..5 OUTBOUND  
'verts/S' edges OPTIONS {  
    uniqueVertices: 'none',  
    uniqueEdges: 'path'  
}  
RETURN CONCAT_SEPARATOR( '->',  
p.vertices[*]._key)  
  
“S->A->D->E”
```

- `uniqueVertices: 'path'` asegura que no existen vertices duplicados en un camino.
- `uniqueVertices: 'global'` asegura que todos los vertices alcanzables son visitados solo una vez (BFS:true).



# Depth/Breath First search

## Ejemplo

- Listar todos los aeropuertos accesibles desde un aeropuerto particular.

```
FOR airport IN OUTBOUND 'airports/LAX' flights  
RETURN DISTINCT airport
```

```
FOR airport IN OUTBOUND 'airports/LAX' flights  
OPTIONS { bfs: true, uniqueVertices: 'global' }  
RETURN airport
```

Cual es más eficiente?

# Combinando documentos y grafos en una misma consulta

## ArangoDB

```
FOR orig IN airports
  FILTER orig._key IN ["JFK", "PBI"] FOR
dest, flight IN
  OUTBOUND orig flights
  FILTER dest.FlightNum IN [859,860]
  RETURN { from: orig.name,
to: dest.name, number: f.FlightNum,
day: f.Day }
```

<https://www.arangodb.com/docs/stable/aql/fundamentals-syntax.html>

```
FOR u IN users
  FILTER u.age < 39
  RETURN u
```

```
FOR u IN users
  FILTER u.status == "not active"
  UPDATE u WITH { status:
  "inactive" } IN users
```

# AQL

## Syntax

- Una consulta AQL debe devolver un resultado (**RETURN**) o ejecutar una operación de modificación de datos (**INSERT**, **UPDATE**, **REPLACE**, **REMOVE**).
- **FOR**: Iterar sobre una colección o vista, todos los elementos de un arreglo, o recorrer un grafo.
- **RETURN**: retornar el resultado de una consulta.
- **FILTER**: Restringir los resultados a elementos que cumplan condiciones lógicas arbitrarias.
- **SEARCH**: Consultar una vista de arangosearch o search-alias.
- **SORT**: Ordenar un arreglo de resultados intermedios ya producidos.
- **LIMIT**: Reducir el número de elementos en el resultado a un máximo especificado.
- **LET**: Asignar un valor a una variable.
- **COLLECT**: Agrupar un arreglo por uno o varios criterios. También se puede contar y agregar.
- **WINDOW**: Realizar agregaciones sobre filas relacionadas.
- **REMOVE**: Eliminar documentos de una colección.
- **UPDATE**: Actualizar parcialmente documentos en una colección.
- **REPLACE**: Reemplazar completamente documentos en una colección.
- **INSERT**: Insertar nuevos documentos en una colección.
- **UPsert**: Actualizar/reemplazar un documento existente o crearlo en caso de que no exista.
- **WITH**: Especificar las colecciones utilizadas en una consulta (solo al comienzo de la consulta).

# AQL

## Ejemplos y tipos de datos

```
FOR u IN users
  FILTER u.type == "newbie" && u.active == true
  RETURN u.name

FOR u IN users
  FOR f IN friends
    FILTER u.active == true && f.active == true && u.id ==
f.userId
    RETURN u.name

FOR u IN users
  LET friends = u.friends
  RETURN { "name" : u.name, "friends" : friends }

LET users = []
FOR u IN users
RETURN u

FOR u IN users
  FILTER u.age < 39
  RETURN u

//subconsultas
FOR p IN persons
  LET recommendations = ( // subconsulta comienza
    FOR r IN recommendations
      FILTER p.id == r.personId
      SORT p.rank DESC
      LIMIT 10
      RETURN r
  ) // termina
RETURN { person : p, recommendations : recommendations }
```

| Data type         | Description  |
|-------------------|--|
| null              | Valor vacío  |
| boolean           | Falso o verdadero                                  |
| number            | Numero con signo (real)                            |
| string            | Texto codificado                                   |
| array / list      | Secuencia de valores referidos por sus posiciones. |
| object / document | Secuencia de valores referidos por sus nombres.    |

# AQL

## Operadores

| Operadores            | Descripción   |
|-----------------------|---|
| <code>==</code>       | Igualdad  |
| <code>!=</code>       | Distinto  |
| <code>&lt;</code>     | Menor   |
| <code>&lt;=</code>    | Menor igual   |
| <code>&gt;</code>     | Mayor   |
| <code>&gt;=</code>    | Mayor igual   |
| <code>IN</code>       | Prueba si un valor esta contenido en un arreglo/lista             |
| <code>NOT IN</code>   | Prueba si un valor no esta contenido en un arreglo/lista          |
| <code>LIKE</code>     | prueba si un valor de texto coincide con un patrón                |
| <code>NOT LIKE</code> | prueba si un valor de text no coincide con un patrón              |
| <code>=~</code>       | prueba si un valor de text coincide con una expresión regular     |
| <code>!~</code>       | prueba si un valor de texto no coincide con una expresión regular |

```

0 == null          // falso
1 > 0             // verdadero
true != null      // verdadero
45 <= "yikes!"    // verdadero
65 != "65"         // verdadero
65 == 65           // verdadero
1.23 > 1.32       // falso
1.5 IN [ 2, 3, 1.5 ] // verdadero
"foo" IN null      // falso
42 NOT IN [ 17, 40, 50 ] // verdadero
"abc" == "abc"      // verdadero
"abc" == "ABC"       // falso
"foo" LIKE "f%"      // verdadero
"foo" NOT LIKE "f%"     // falso

25 > 1 && 42 != 7          // verdadero
22 IN [ 23, 42 ] || 23 NOT IN [ 22, 7 ] //v
25 != 25           //falso

```

- + suma
- - resta
- \* multiplicacion
- / division
- % modulo

# AQL

## Consultas (DAQ y DMQ)

- DAQ (Data access Query)

```
FOR doc IN users
  FILTER doc.status == "active"
  SORT doc.name
  LIMIT 10
```

- DMQ (Data Modification Query)

```
INSERT {
  firstName: "Alex",
  name: "Perez",
  profession: "artista"
} INTO users
```

```
UPDATE "Alex" WITH {
  status: "activo",
  location: "Rancagua"
} IN users
```

```
REPLACE {
  _key: "user1", // no se puede modificar
  firstName: "Alex",
  name: "Peréz",
  status: "activo",
  level: "vip"
} IN users
```

```
REMOVE "Alex" IN users
```

- DMQ (Data Modification Query)

```
FOR i IN 1..10000
  INSERT {
    id: 100000 + i,
    edad: 18 + FLOOR(RAND() * 25),
    nombre: CONCAT('test', TO_STRING(i)),
    estado: i % 2 == 0 ? "active" : "not active",
    sexo: i % 3 == 0 ? "hombre" : i % 3 == 1 ?
      "mujer" : "no-informado"
  } IN users
```

```
FOR u IN users
  INSERT u IN backup
```

```
LET r1 = (FOR u IN users REMOVE u IN users)
LET r2 = (FOR u IN backup REMOVE u IN backup)
RETURN true
```

# AQL

## Operaciones de alto Nivel

```
FOR u IN users
  LET subquery = (FOR l IN locations RETURN
l.location)
  RETURN { "user": u, "locations": subquery }
```

```
FOR value IN ["foo", "bar", "bar", "baz", "foo"]
  RETURN DISTINCT value
```

```
FOR u IN users
  FILTER u.active == true
  SORT u.age ASC
  LIMIT 5
  FILTER u.sexo == "mujer"
  RETURN u
```

```
FOR u IN users
  SORT u.lastName, u.firstName, u.id DESC
  RETURN u
```

```
FOR u IN users
  LIMIT 5
  RETURN u
```

```
FOR u IN users
  LET numRecom = LENGTH(u.recomendaciones)
  RETURN {
    "usuario" : u,
    "numRecommendaciones" : numRecom,
    "esUsuarioPopular" : numRecom >= 10
  }
```

- La operación **COLLECT** se puede utilizar para agrupar datos por uno o varios criterios de grupo.

```
FOR u IN users
  COLLECT AGGREGATE minAge = MIN(u.age), maxAge =
MAX(u.age)
```

```
  RETURN {
    minAge,
    maxAge
  }
```

- LENGTH() / COUNT()
- MIN()
- MAX()
- SUM()
- AVERAGE() / AVG()
- STDDEV\_POPULATION() / STDDEV()
- STDDEV\_SAMPLE()
- VARIANCE\_POPULATION() / VARIANCE()
- VARIANCE\_SAMPLE()
- UNIQUE()
- SORTED\_UNIQUE()
- COUNT\_DISTINCT() / COUNT\_UNIQUE()
- BIT\_AND()
- BIT\_OR()
- BIT\_XOR()

```
WITH collection1 [, collection2 [, ... collectionN ] ]
```

- FOR ... IN collection
- INSERT ... INTO collection
- UPDATE ... IN collection
- GRAPH "graph-name" (via the graph definition)

# 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] 1 #capture
2 git clone -b oasis_connector --single-branch https://github.com/arangodb/interactive_tutorials.git
3 !sync -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: TUUyldgmpzoz9vsgn79708
Password: TUTe56x1xmuxchib9t1jdrv
Database: TUUy29y9plwkarpe9mkn2c628

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

At the bottom, there is a file list:

- mod5\_1.ppt
- mod5\_1.ppt
- comprobante\_trabajo.pdf
- Formato Retribución.docx
- DB18F.png

<https://github.com/adigenova/uohpmd/blob/main/code/Arango%20GraphDB.ipynb>

<https://github.com/adigenova/uohpmd/blob/main/code/NoSQL%20ArangoDB%20ejemplo.ipynb>

<https://www.arangodb.com/docs/stable/aql/tutorial.html>

# Consultas?

Consultas o comentarios?

Muchas gracias