Programa de formación MLDS

UNIVERSIDAD NACIONAL DE COLOMBIA

Ben Chams - Fotolia

Módulo
BIG DATA
Bases de Datos NoSQL

Por
Ing. Elizabeth León Guzmán, Ph.D.

Agenda

- NoSQL
- Clave valor (Key Value)
- Documentos (Document)
- Grafos (Graph)

Not Only SQL

2009

Dynamo de Amazon
Bigtable de Google

NoSQL

- Se usan cuando BD relacionales tienen problemas de escalabilidad y rendimiento (miles de usuarios concurrentes y millones de consultas diarias).
- No son E/R
- No usan estructura de datos “Tabla”
- Formatos usados: clave-valor, grafos o mapeo de columnas.

No SQL

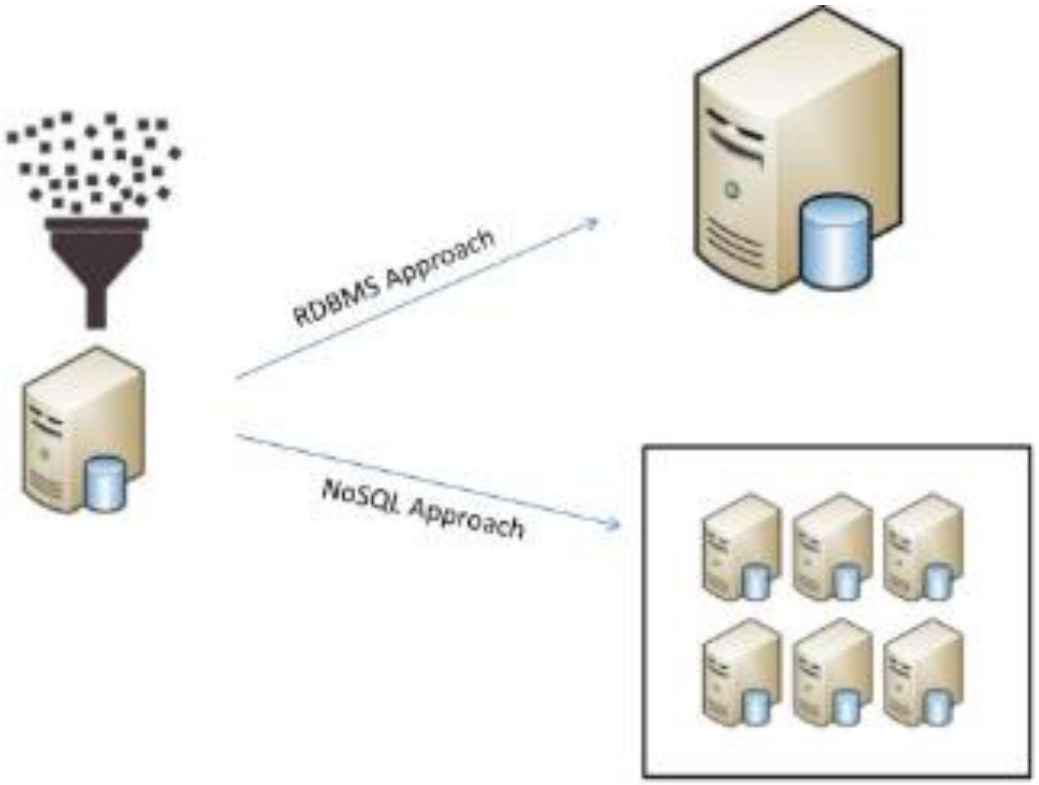


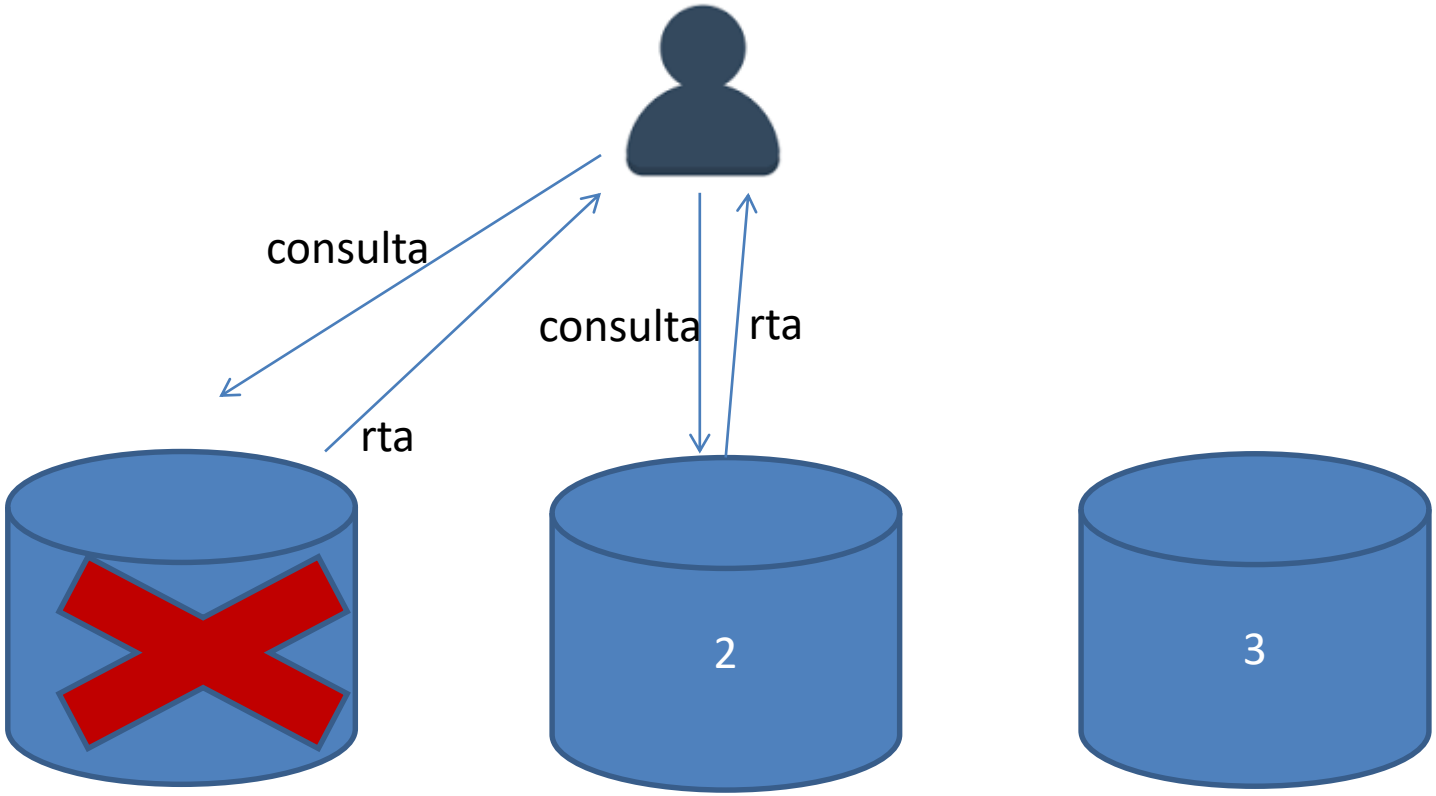
Figura tomada de Grisha Weintraub 2014

Distribución de los datos a través de **fragmentación/ partición** de los datos

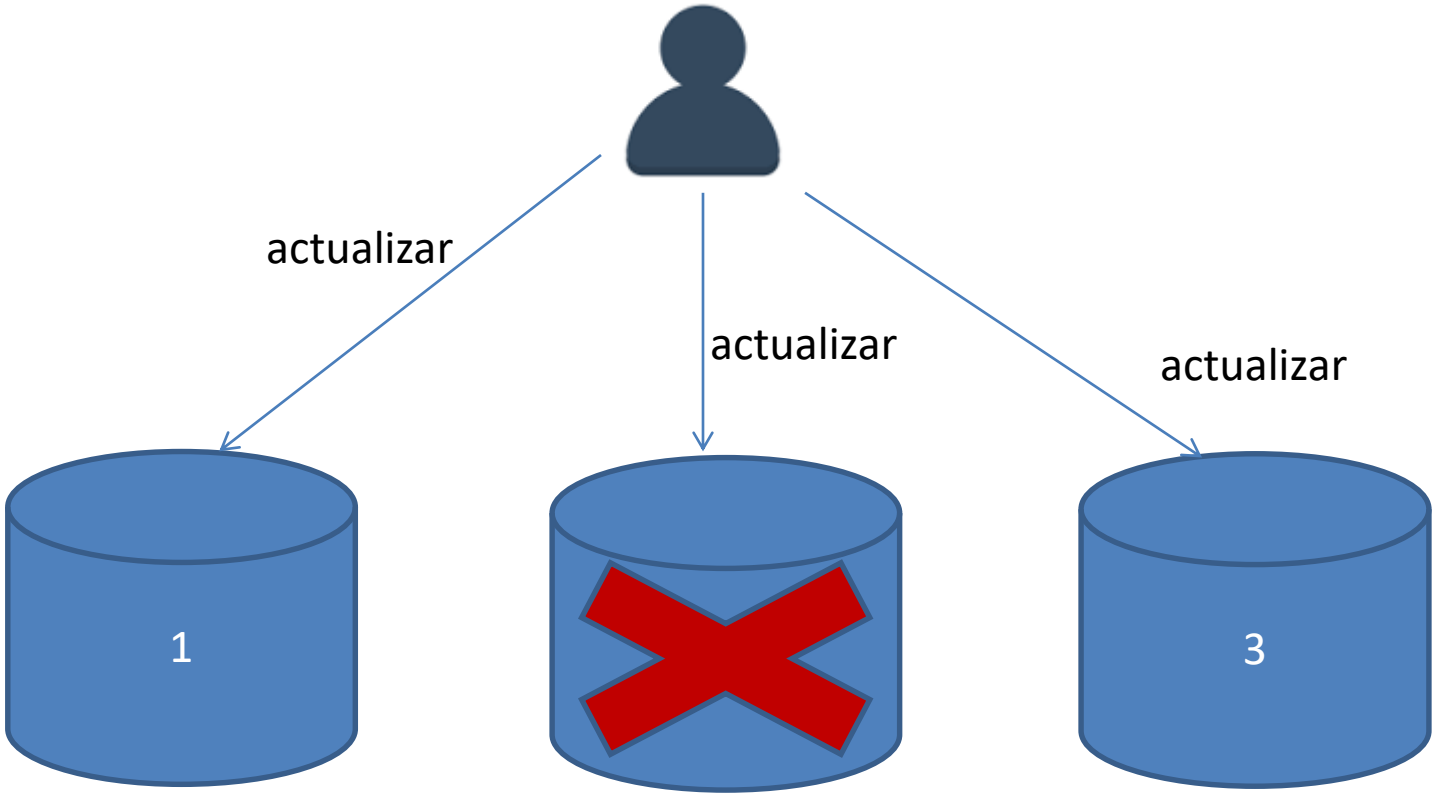
Disponibilidad a través de **Replicación** (los datos se copian en varios nodos/máquinas

Replicación vs Actualización

Replicar asegura disponibilidad



Actualizar

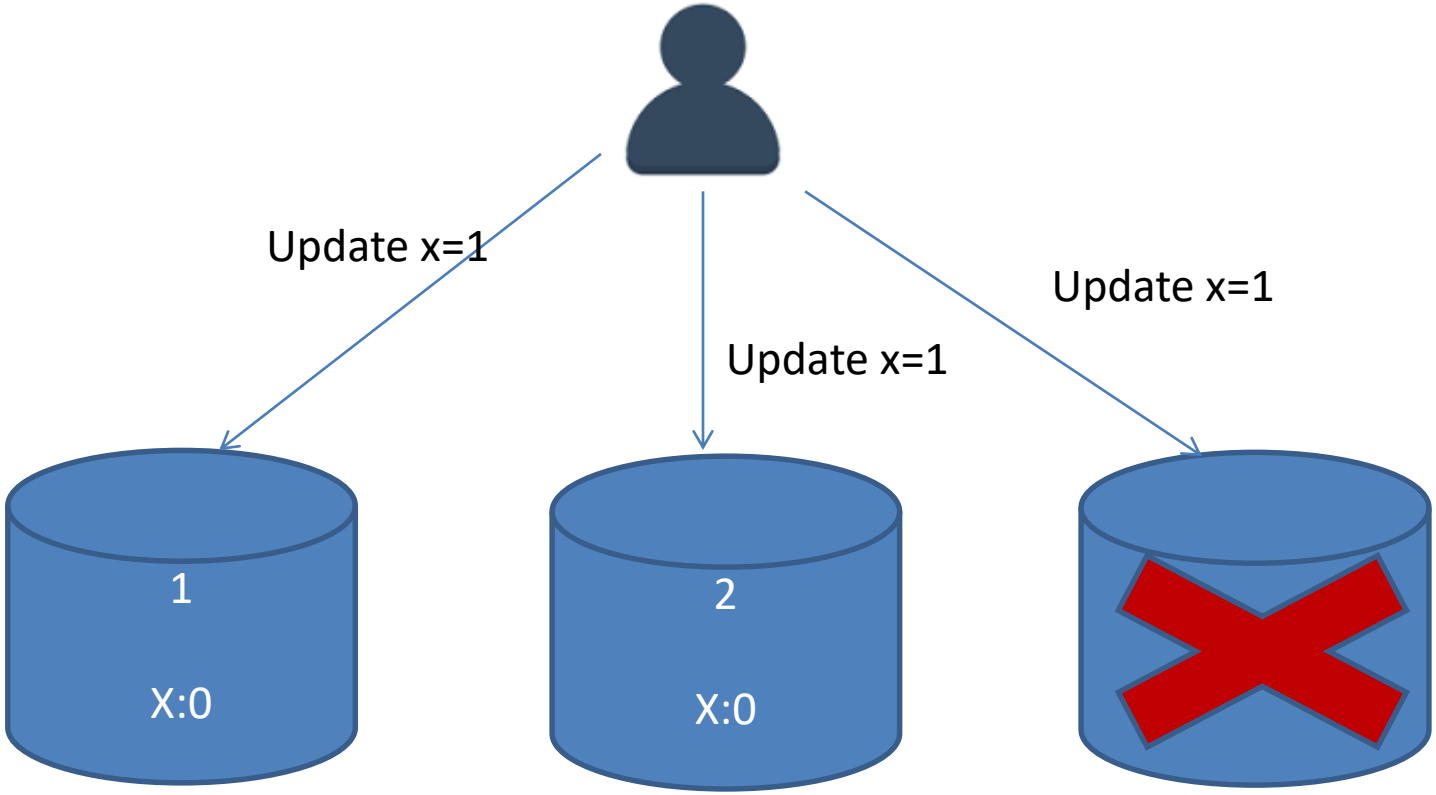


Actualizaciones sin replicas disponibles Problemas

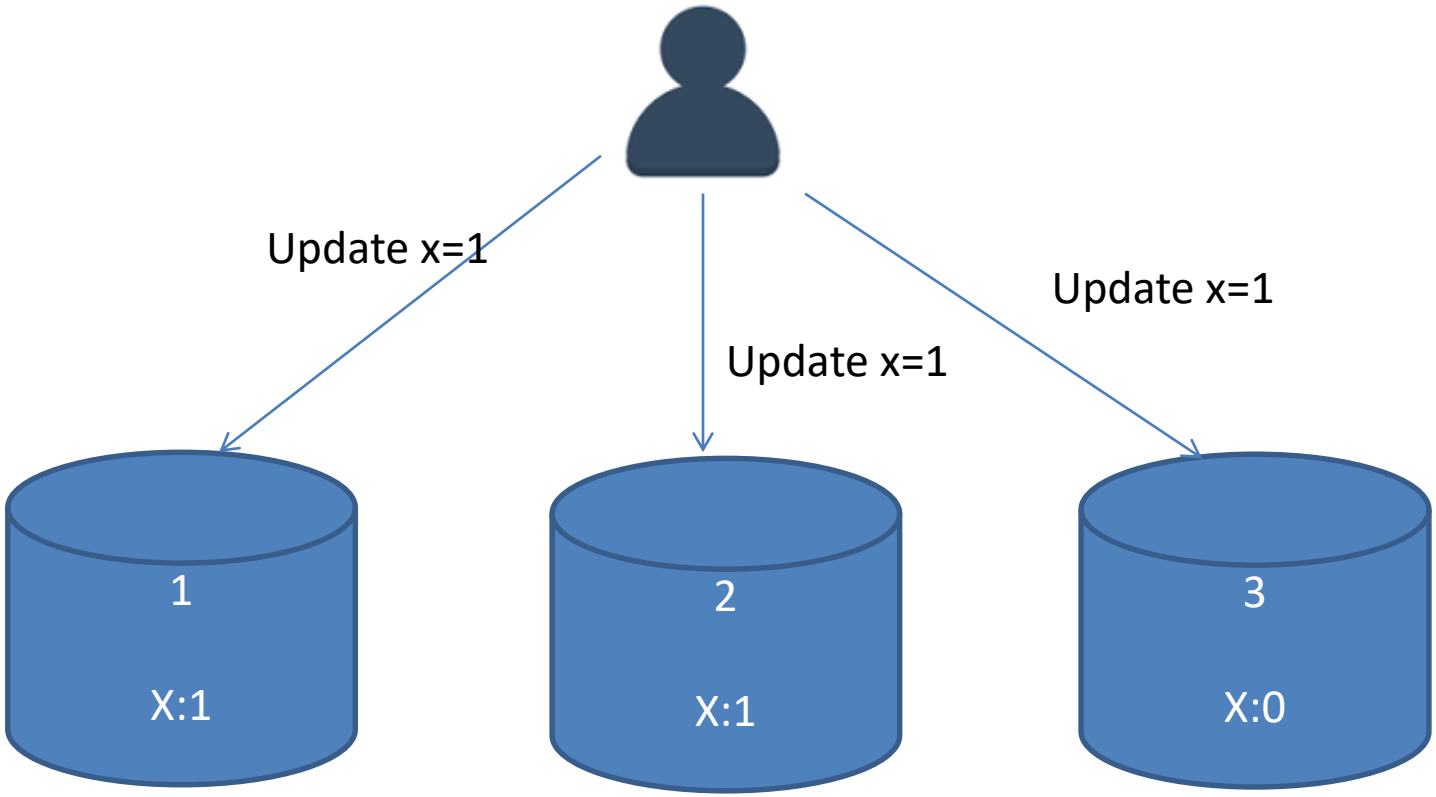
- Opción 1 : Esperar hasta que el nodo replica que falla se recupere - Problema: No se puede actualizar el sistema ->"no disponible"
- Opción 2 : Seguir adelante (" Eventual consistencia") - Problema: ¿Qué hacer cuando el nodo que falló se recupera? - ¿Qué pasa si se han enviado lecturas al nodo que ha fallado?
- Opción 3 : Mayoría escribe/mayoría lee

Mayoría escribe/Mayoría Lee

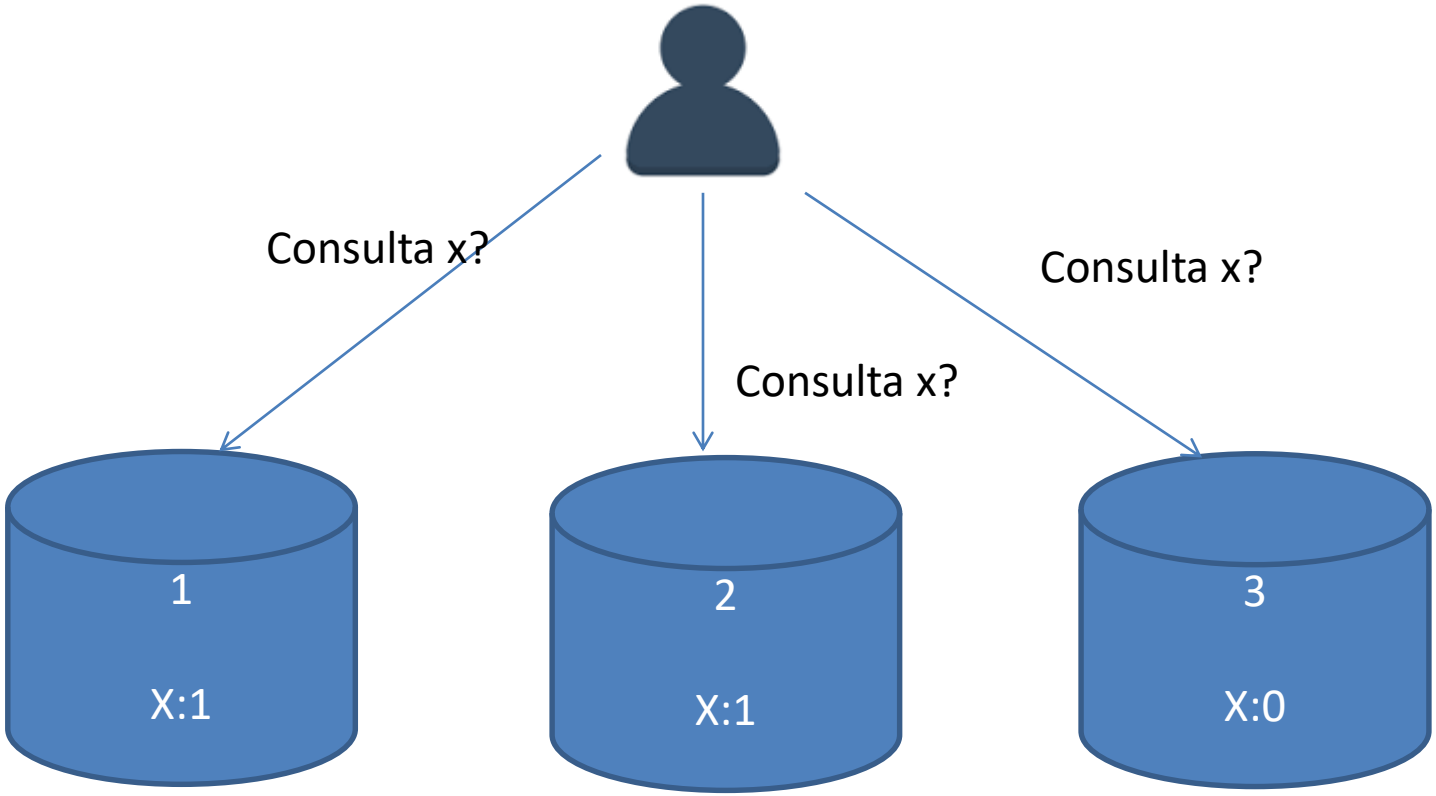
- Si la mayoría de replicas están disponibles se actualizan los datos



Mayoría escribe/Mayoría Lee



Mayoría escribe/Mayoría Lee



Actualizaciones
Resumen

Opción	Ventaja	Desventaja
Esperar hasta que el nodo replica que falla se recupere	• Solo lee una réplica • Plenamente consistente	No disponible mientras la replica este fallando
Seguir adelante "Eventual consistencia"	• Solo lee una réplica. • Tolera múltiples fallas de replicas	Lee pero no obtiene la versión más reciente de los datos o son inconsistentes.
Mayoría escribe/mayoría lee	• Consistente • Capaz de tolerar algunas fallas en replicas	Lectura de muchas réplicas (tiempo) No disponible cuando más de la mitad de las réplicas falla.

Enfoques:

- Mantener un registro de cambios (log) en cada réplica

  Copiar y reproducir a la recuperación de réplica
- Periódicamente comparar réplicas
- Copiar todo el estado de una réplica existente a una nueva réplica
- …

Clave – Valor
(key-value)

- Más popular y sencilla
- Estructura de "diccionario"
- Datos son asociados por claves a valores. Los valores no conforman estructuras particulares.

El valor puede ser un entero, una cadena de caracteres, un arreglo. Pueden ser documentos: JSON

Clave – Valor
(key-value)

Key	Value
8.01	La Ciudad y los Perros de Mario Vargas Llosa
8.02	Cien Años de Soledad de Gabriel Garcia Marquez
8.12	La Divina Comedia de Dante Alighieri
8.50	Don Quijote de La Mancha de Miguel de Cervantes

Lenguaje de programación es *get* y *put*

get (“8.02”) -> “Cien Años de Soledad de Gabriel Garcia Marquez”

put (“9.02”, “La Iliada de Homero”)

Clave – Valor
(key-value)

- Simple de programar e implementar
- Más rápido que SQL
- Fácil de distribuir
- Falta de consistencia (coherencia, integridad)
- Se pierde atomicidad, el almacenamiento de key-values no son transacciones.

Clave – Valor
(key-value)

- Una máquina-> No garantía de “Atomicidad” (todo o nada)

- Múltiples máquinas-> Consistencia eventual significa que las replicas se sincronizarán, pero mientras no lo hagan los datos estaran *inconsistentes*.

Clave – Valor
(key-value)

- Cada elemento identificado con **llave única** , lo que permite recuperación de la información rápida.
- Información almacenada como un objeto binario.
- Eficiente para escritura y lectura

  Hbase (Hadoop)

  BigTable (Google)

  Dynamo(Amazon)

  Cassandra (desarrollada por Facebook, la usa twitter)

Clave – Documento

- Los datos referencian claves a ***documentos*** (JSON/XML)

Key	Value
8.01	{Titulo: “La Ciudad y los Perros”, Autor:“ Mario Vargas Llosa”}
8.02	{Titulo: “Cien Años de Soledad”, Año: “1967”}
8.12	{Titulo: “La Divina Comedia”, Año: “1304”, Autor:”Dante Alighieri”}
8.50	{Titulo: ”Don Quijote de La Mancha”, Autor: “Miguel de Cervantes”}

- Buscar documentos por clave
- Cierta capacidad para buscar contenido de los documentos. Por lo general, no JOINS ni actualizaciones de múltiples documentos.

BD Documento

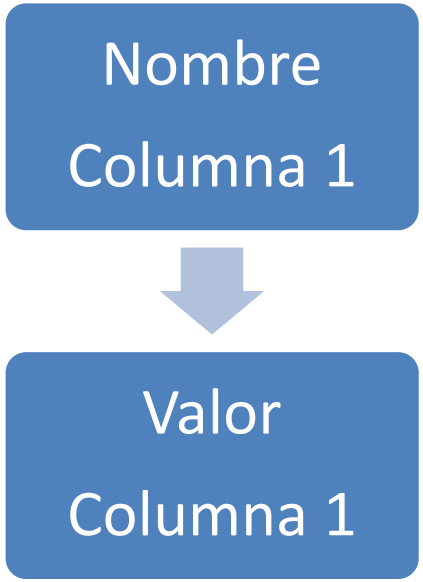
- Sub clase de BD clave-valor
- Extienden el concepto de clave-valor
- clave- conjunto de valores dentro de un documento
- Organiza grupos llamados **colecciones** (análogas a tablas relacionales)

BD Documento

- Permiten alto nivel de organización, por ejemplo para un sitio de comercio electrónico se podrían definir **colecciones** de:
  - Clientes (nombre, dirección)
  - Ordenes (producto, dirección de envío)
  - Productos (departamento, precio)

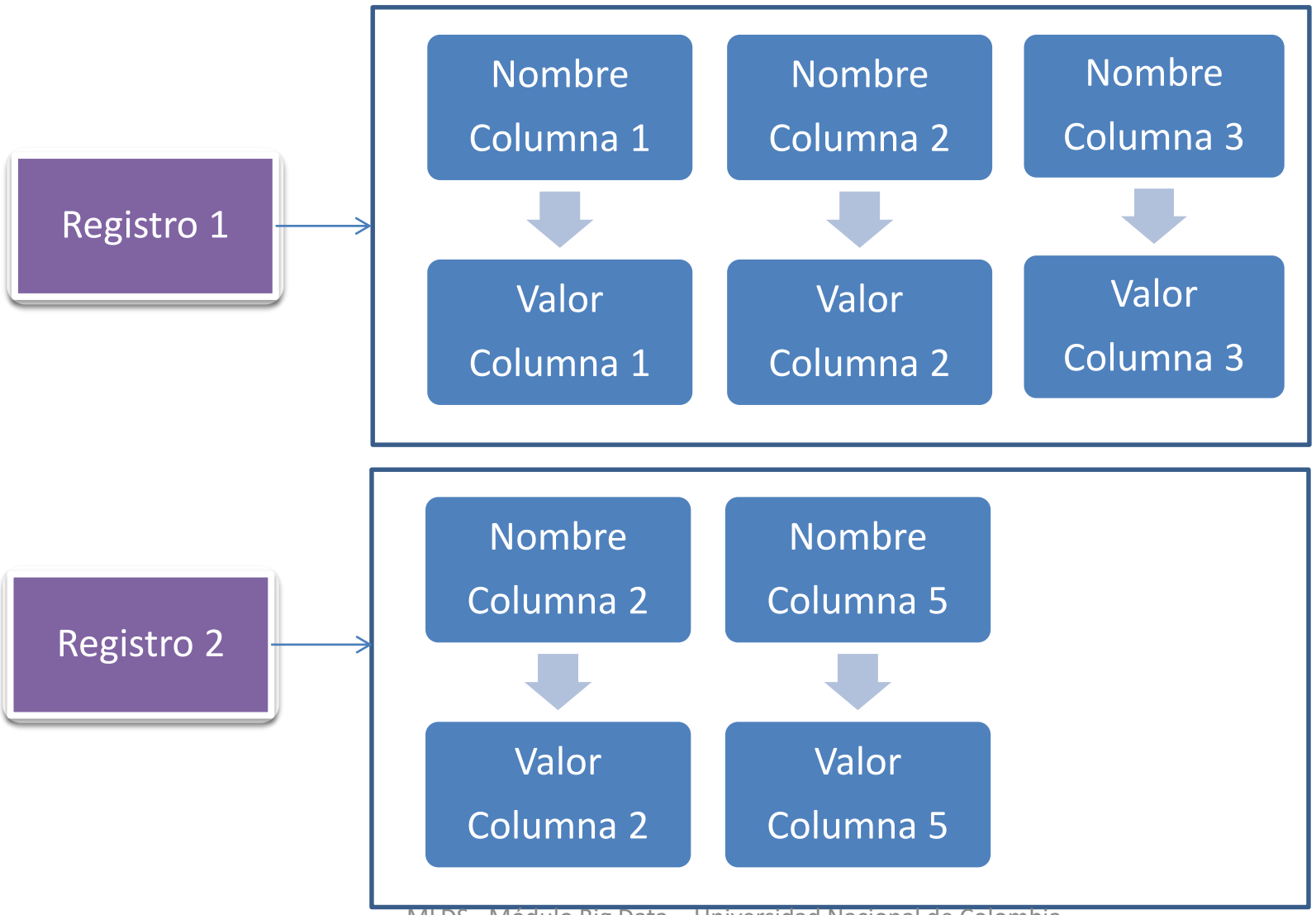
CouchDB
Mongo (10 gen Inc Company)

Clave - valor

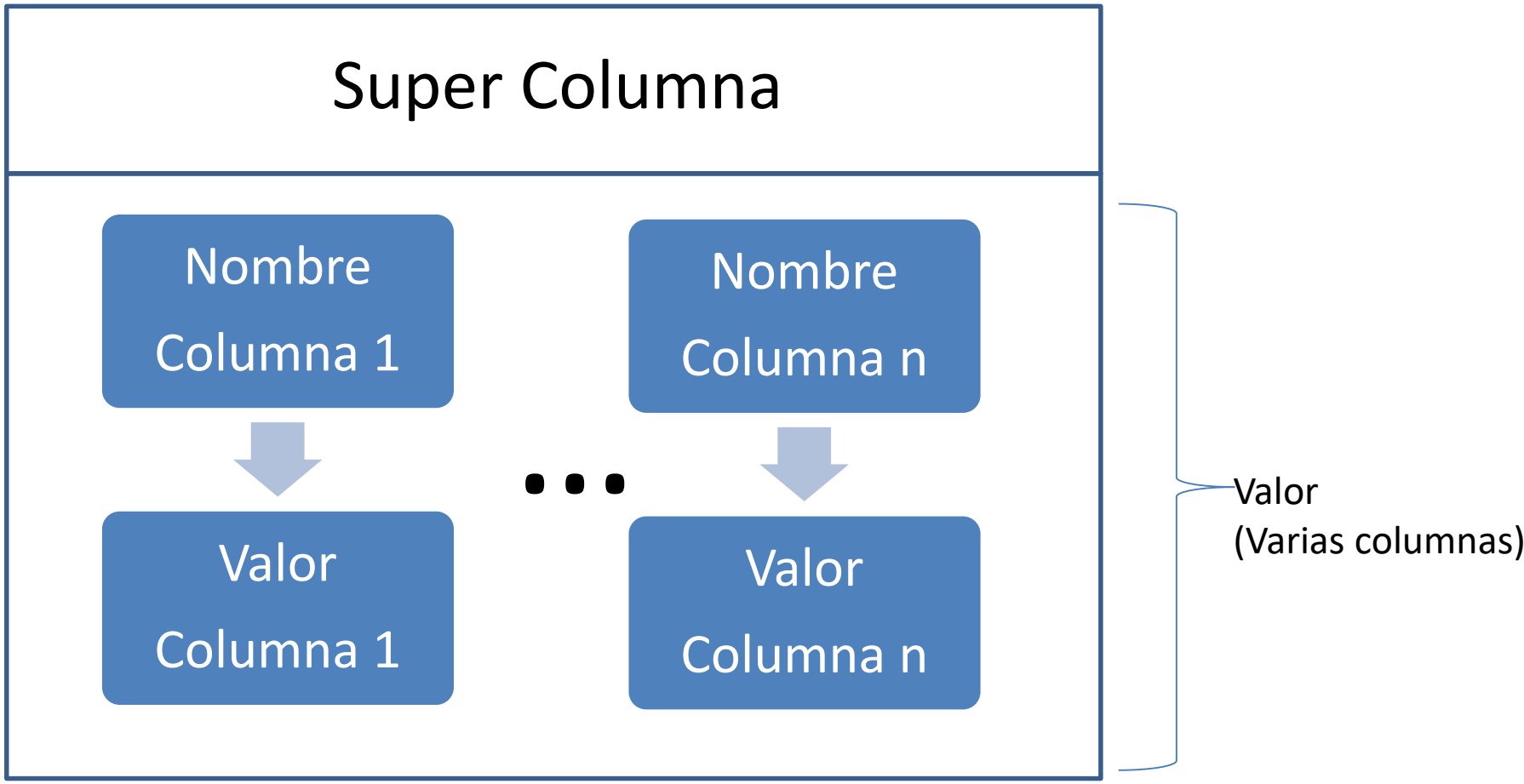


columna

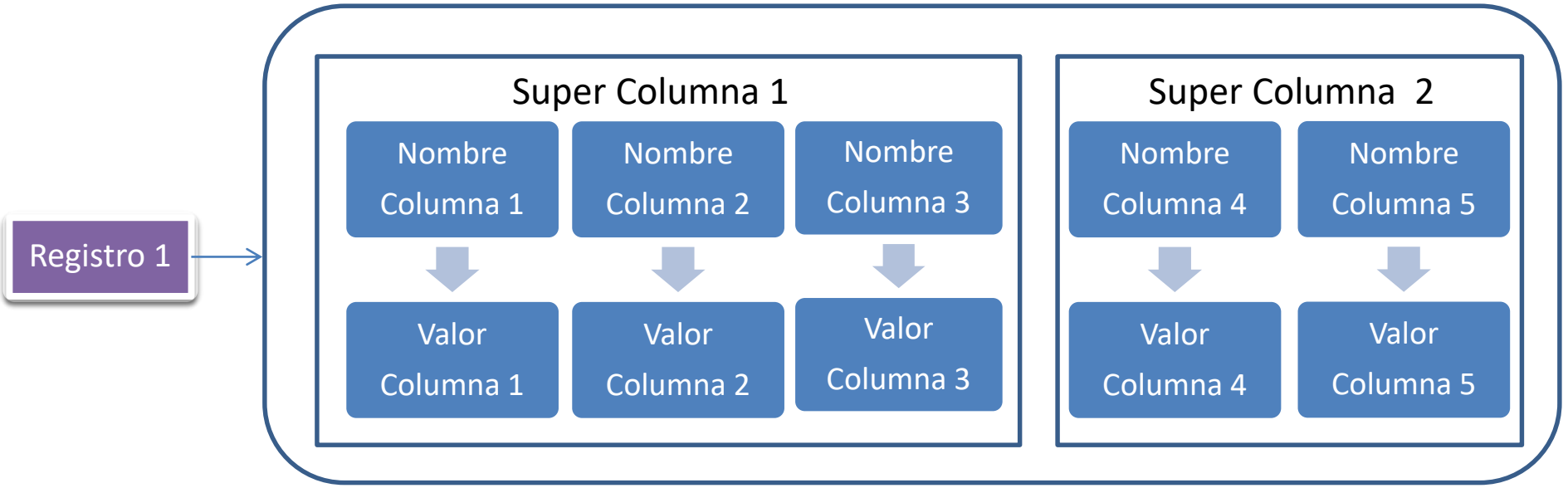
Clave – Valor
Familia de Columnas



Clave - Valor


Super Columna
Nombre Columna 1
Nombre Columna n
Valor Columna 1
Valor Columna n
Valor
(Varias columnas)

Clave – Valor
Familia de Super Columnas



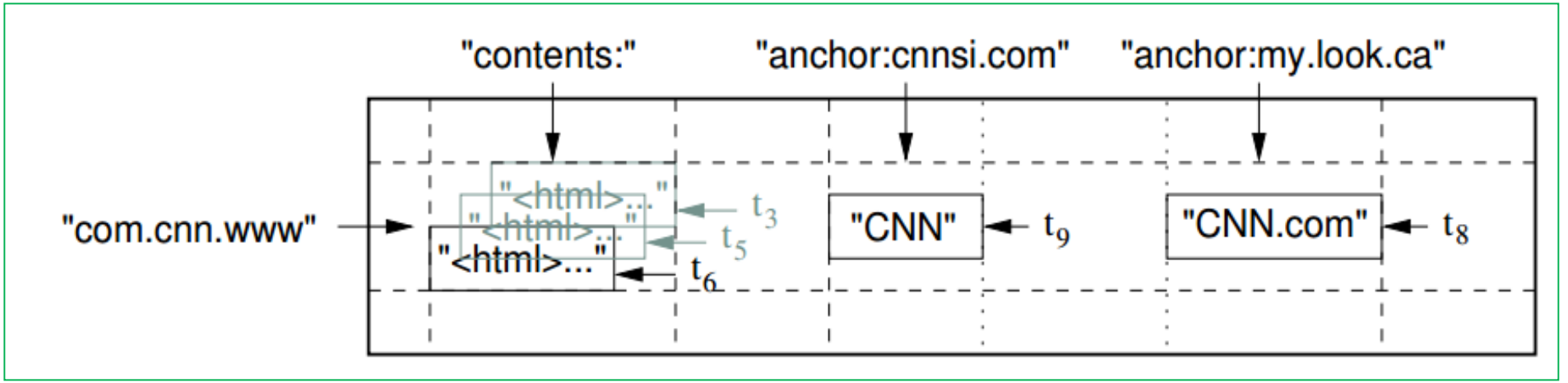
BigTable

- Creada por Google (2004)
- Mapa disperso , distribuido y persistente
- Construido sobre GFS (Google File System)
  - Tablas multidimensionales dispersas
  - Tiempo (manejo de versiones)
  - Tablas se fragmentan por columnas (cada división de 200MB)
  - Las divisiones son distribuidas en diferentes máquinas (balanceo de carga)

Big table

Los datos son indexados usando nombres de filas , columnas, y tiempo.

(row:string, column:string, time:int64) → string



Ejemplo: copia de páginas web e información relacionada de ellas

Urls como claves de filas, aspectos de la url como como claves de las columnas: contenido y anclas

Figura tomada de [1]

Big Table

- Busquedas por fila:
  - Nombres de las url ordenadas
  - Páginas del mismo dominio son almacenadas en la misma partición (contiguas).
- Claves de columnas agrupadas por familias
  - Datos por flia del mismo tipo
- Número pequeño de flias
- Muchas columnas

Familia: calificador

Ejemplo: columna clave -> lenguaje
familia de columnas -> ancla (el calificador es el nombre de la url que referencia) y el contenido de la celda es el enlace.

Big Table
escribir

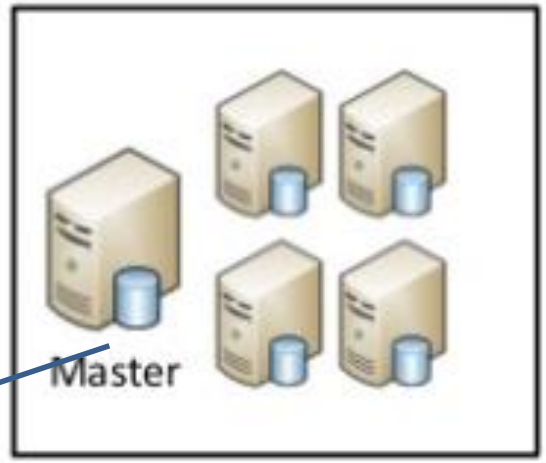
```
// Open the table
Table *T = OpenOrDie("/bigtable/web/webtable");

// Write a new anchor and delete an old anchor
RowMutation r1(T, "com.cnn.www");
r1.Set("anchor:www.c-span.org", "CNN");
r1.Delete("anchor:www.abc.com");
Operation op;
Apply(&op, &r1);
```

Figure 2: Writing to Bigtable.

Big Table
leer

```
Scanner scanner(T);
ScanStream *stream;
stream = scanner.FetchColumnFamily("anchor");
stream->SetReturnAllVersions();
scanner.Lookup("com.cnn.www");
for (; !stream->Done(); stream->Next()) {
  printf("%s %s %lld %s\n",
         scanner.RowName(),
         stream->ColumnName(),
         stream->MicroTimestamp(),
         stream->Value());
}
```

Big Table



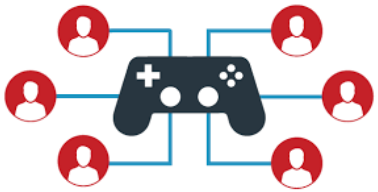
Mantiene los metadatos del sistema

Cloud Big table

- Gran scala (cientos de Petabytes)
- Millones de operaciones por segundo
- Baja latencia

Aplicaciones

- Aplicaciones web que manejan directamente : individuos y consumidores:



  - juegos “on line” multijugador
    - Información de cada una de las sesiones de los individuos

  - Administrar “shopping car” de los compradores “on line”.

Ejercicio

Mencionar un *ejemplo de datos de su empresa* que podrían guardar en “BigTable”

¿Cuáles serían las columnas y las claves?

Cassandra

Desarrollada originalmente en Facebook para facilitar búsquedas en Inbox, *open-sourced* en 2008 y aceptada como proyecto apache en 2010.

- Código abierto
- Escrito en Java
- Multiplataforma

- **Distribución** de datos a través de múltiples nodos de datos.
- Escala de forma **horizontal**. El sistema escala añadiendo nuevos nodos basados en *hardware commodity* de bajo costo.

Características

- Cassandra ofrece el *AID* del *ACID test*:
  - Atomicidad
  - Aislamiento
  - Durabilidad
- Ofrece **alta disponibilidad** a cambio de ser **eventualmente consistente.** Si alguno de los nodos se cae el servicio no se degradará. La Consistencia es eventual, pero tratable.
- No existe el concepto de integridad referencial o llaves foráneas.

Características

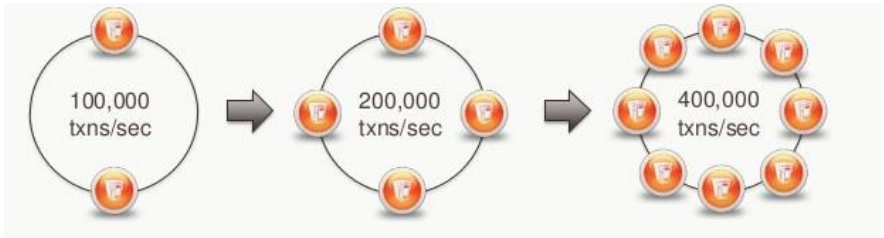
- Eficiencia **linealmente escalable**. Rendimiento de forma lineal respecto al número de nodos que se añaden.

  Ej: si 2 nodos soportan 100.000 operaciones por segundo,

  4 nodos soportan 200.000 operaciones por segundo

predictibilidad del sistema

Tomado de datastack.com



- Detección y recuperación de fallos transparente.
- Modelo de datos dinámico y flexible.
- CQL (Cassandra Query Language).

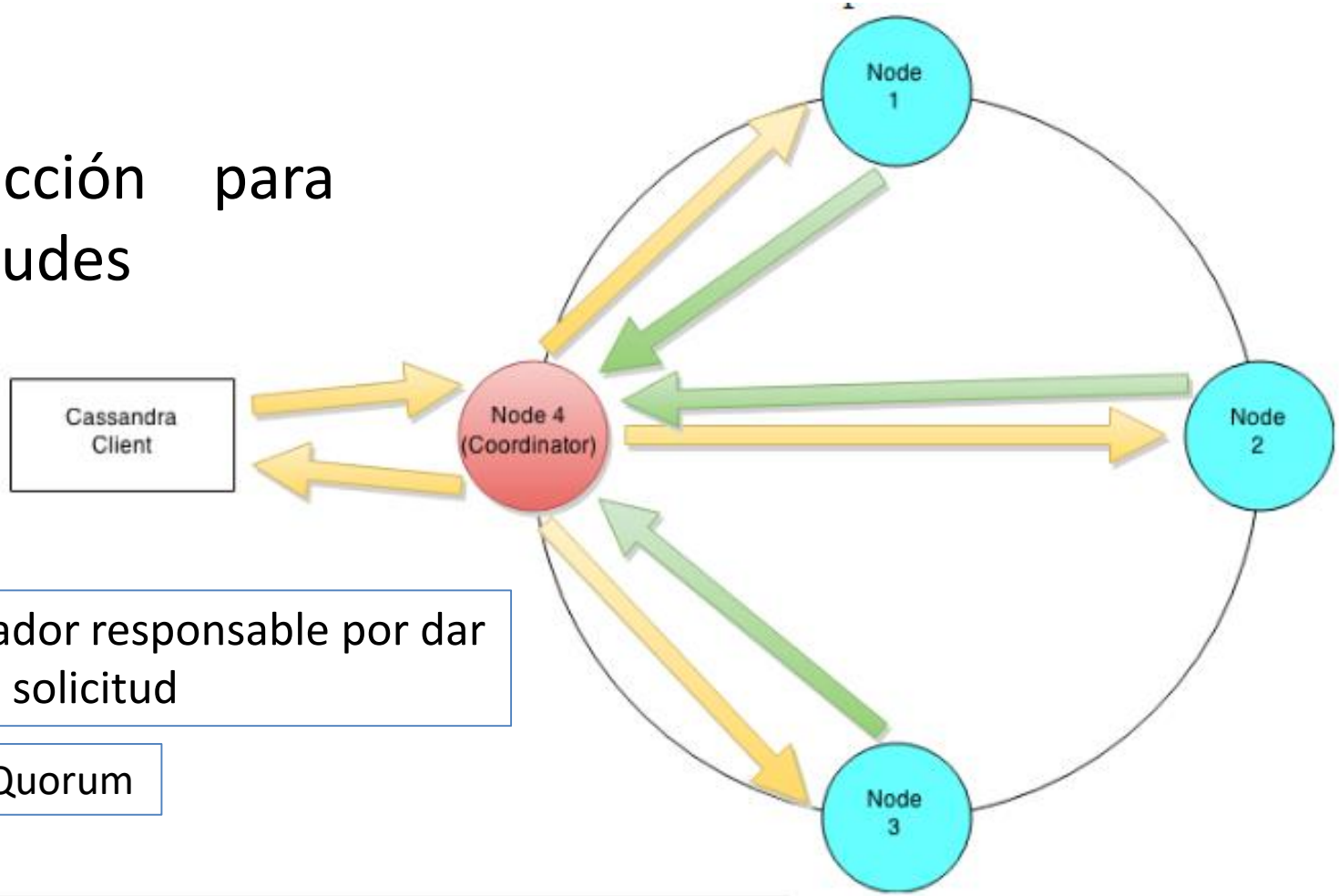
Arquitectura

Arquitectura de anillo, **peer to peer**

- Elimina los puntos de fallo único
- Carencia de nodo Maestro y de la relación Maestro-Esclavo.
- Todo los nodos juegan el mismo rol.
- Cualquiera de los nodos puede tomar *el rol de coordinador* de una *consulta*.

Arquitectura

- Interacción para solicitudes



Nodo coordinador responsable por dar Respuesta a la solicitud

Quorum

Modelo de Datos – Conceptos Básicos

- Keyspace:
  - Contenedor de tablas e índices. Similar a una base de datos o esquema en RDBMS.

- Table:
  - Similar a una tabla en RDBMS.

- Primary-key:
  - Se utiliza para identificar únicamente una fila en una tabla.
  - Permite realizar la distribución de las filas en un cluster.

Modelo de Datos – Conceptos Básicos

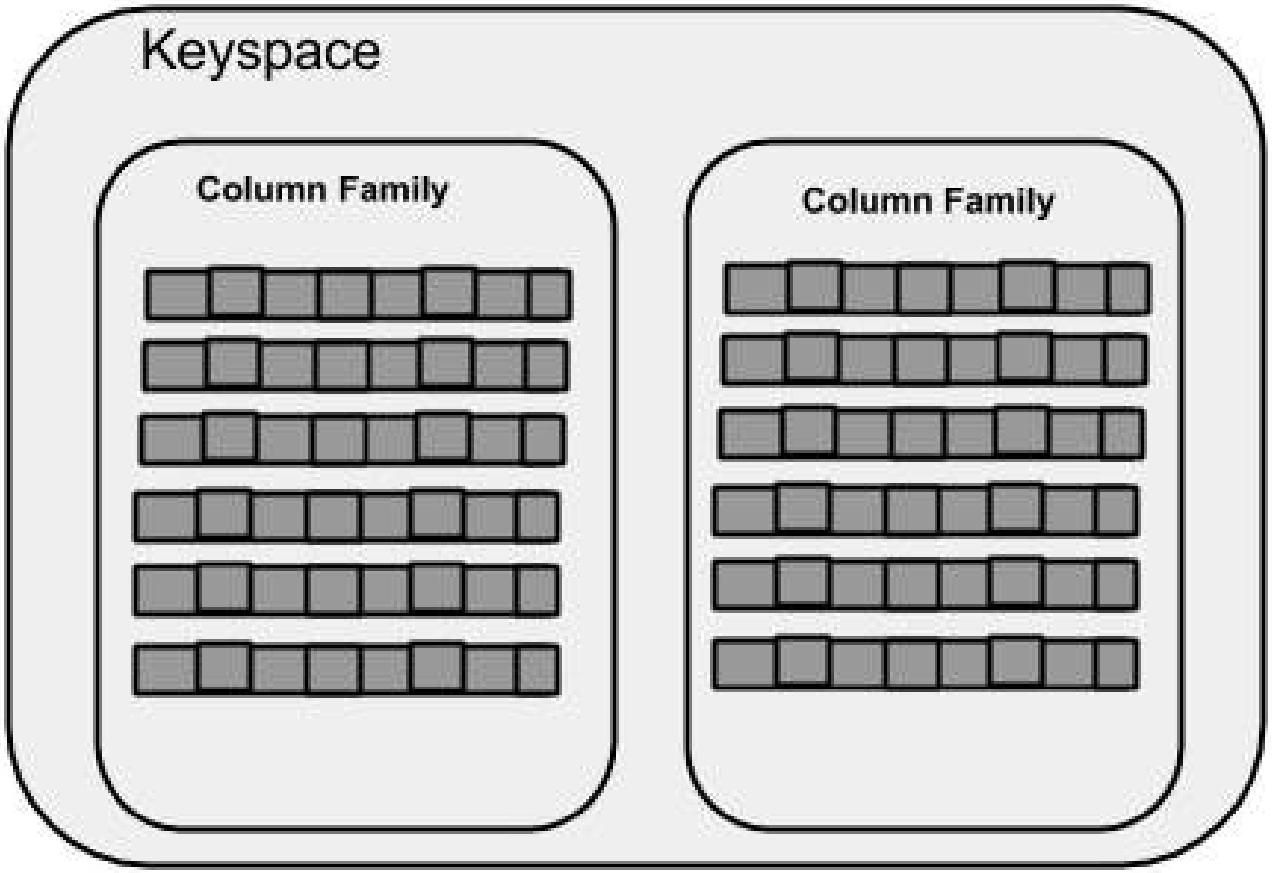
- Index:
  - Similar al indice en RDBMS pues acelera operaciones de lectura.

- Column Family (CF):
  - Es un contenedor de columnas.
  - Es un contenedor de filas ordenadas.
  - Cada fila posee las mismas CFs, pero en cada CF no se fuerzan las misma columnas.

- Column:
  - Es la estructura básica de almacenamiento en Cassandra.
  - Posee tres valores: name, value, timestamp.

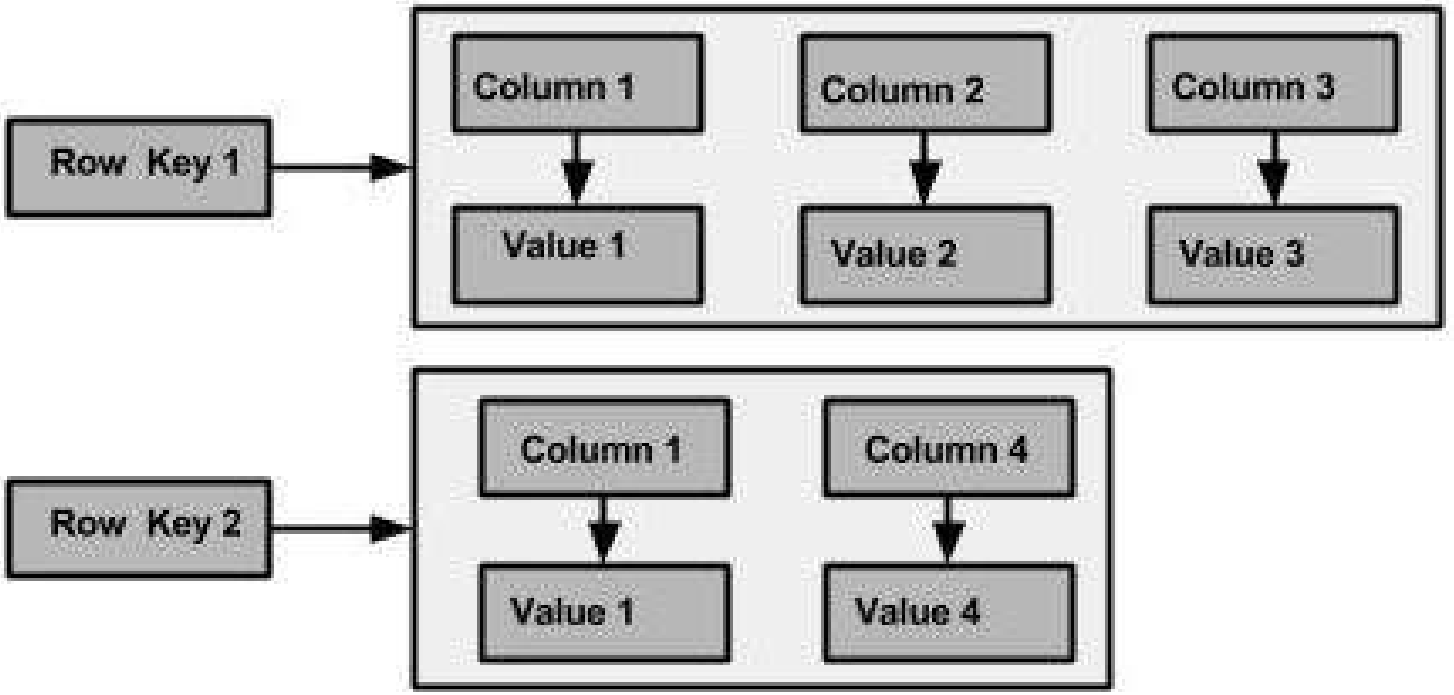
Modelo de Datos

- Keyspace:



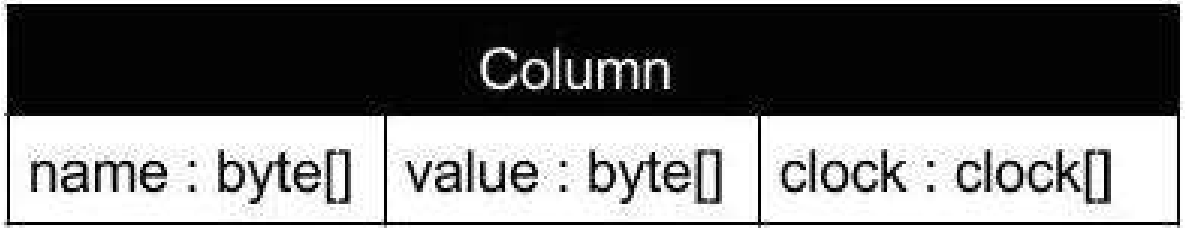
Modelo de Datos

- Column Family:



Modelo de Datos

- Column:



Cassandra Query Language - CQL

- CQL es muy similar al SQL, por ello la curva de aprendizaje es reducida.

- DDL:
  - CREATE
  - ALTER
  - DROP
- DML:
  - INSERT
  - UPDATE
  - DELETE
  - TRUNCATE
- QUERY:
  - SELECT

- DATA TYPES:
  - INT
  - BIGINT
  - DECIMAL
  - ASCII
  - VARCHAR
  - DATE (TIMESTAMP)
- Nuevos DATA TYPES:
  - MAP
  - LIST
  - SET

Aplicaciones en la Industria

- Aplicaciones IoT: Perfecta para consumir grandes cantidades de información rápida de sensores, dispositivos y mecanismos similares.
- Mensajería: Provee base de datos para múltiples plataformas de mensajería en telefonía móvil.
- Seguimiento y monitoreo de usuarios: Muchas compañías de entretenimiento utiliza Cassandra para seguir y monitorear las actividades de sus usuarios, interacciones con sus películas, música, aplicaciones web.
- Algunas compañías que utilizan Cassandra: CERN, eBay, Netflix, Reddit, GitHub, Instagram.

EJEMPLO CQL 3.0

- Listar keyspaces existentes:

```
DESC KEYSPACES;
```

- Crear nuevo Keyspace:

```
CREATE KEYSPACE ejemplo WITH REPLICATION
= { 'class' :'SimpleStrategy', 'replication_factor' : 1 };
```

- Seleccionar keyspace:

```
USE ejemplo;
```

- Crear Column Family (Table):

```
CREATE TABLE users (user_name varchar, password varchar,
                    gender varchar, session_token varchar,
                    state varchar, birth_year bigint,
                    PRIMARY KEY (user_name));
```

EJEMPLO CQL 3.0

- Insertar filas:

```
INSERT INTO users (user_name, password, gender, last_name)
VALUES ('cbrown', 'ch@ngem4a','male', 'chivas');
```

- Crear índice:

```
CREATE INDEX ON users(last_name);
```

- Alterar tabla:

```
ALTER TABLE users ADD name varchar;
```

- Actualizar valor:

```
UPDATE users SET name = ’pedro' WHERE user_name = 'santiago';
```

- Seleccionar:

```
SELECT * FROM users;
select * from users where last_name = 'Sanchez';
SELECT * FROM userslimit 1;
```

EJEMPLO CQL 3.0

- Colección SET:

```
ALTER TABLE users ADD emails set<text>;

INSERT INTO users(user_name, password, gender, last_name, emails)
VALUES ('Laura', '5m8svvd', 'female', 'Paez', {'lau@uno.com'});

UPDATE users SET emails = emails+{'lua1234@uno.com'}
WHERE user_name = 'Laura’;
```

EJEMPLO CQL 3.0

- Colección LIST:

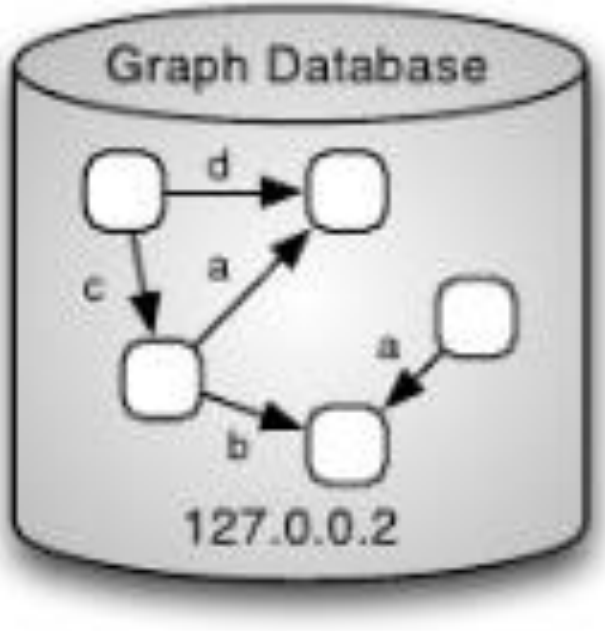
```
ALTER TABLE users ADD top_places list<text>;

UPDATE users SET top_places = ['rivendell', 'rohan’]
WHERE user_name = 'Laura’;

UPDATE users SET top_places = top_places+[ 'the shire' ]
WHERE user_name = 'Laura';

UPDATE users SET top_places[2] = 'riddermark’
WHERE user_name ='Laura';

DELETE top_places[3] FROM users WHERE user_name = 'Laura’;
```

EJEMPLO CQL 3.0

- Colección MAP:

```
ALTER TABLE users ADD telefonos map<text, text>;

UPDATE users SET telefonos = {'casa':'123456', 'oficina':'1236'}
WHERE user_name = 'Laura';

UPDATE users SET telefonos ['celular'] = '3012356’
WHERE user_name = 'Laura’;
```

Bases de Datos basada en grafos

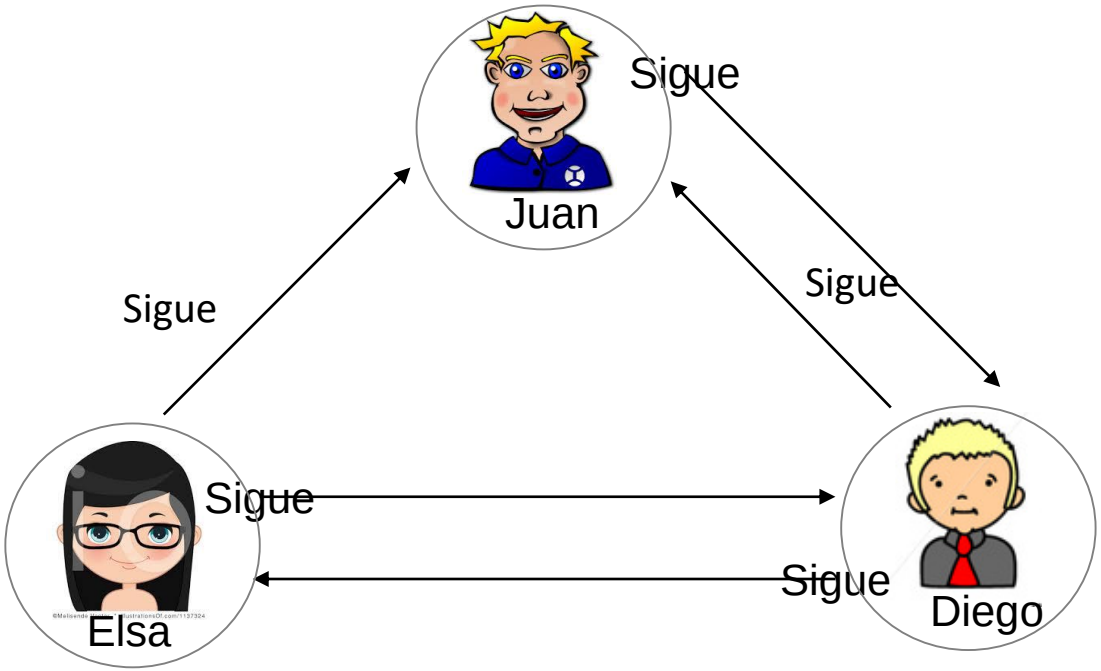


Grafo multirelacional

Grafo

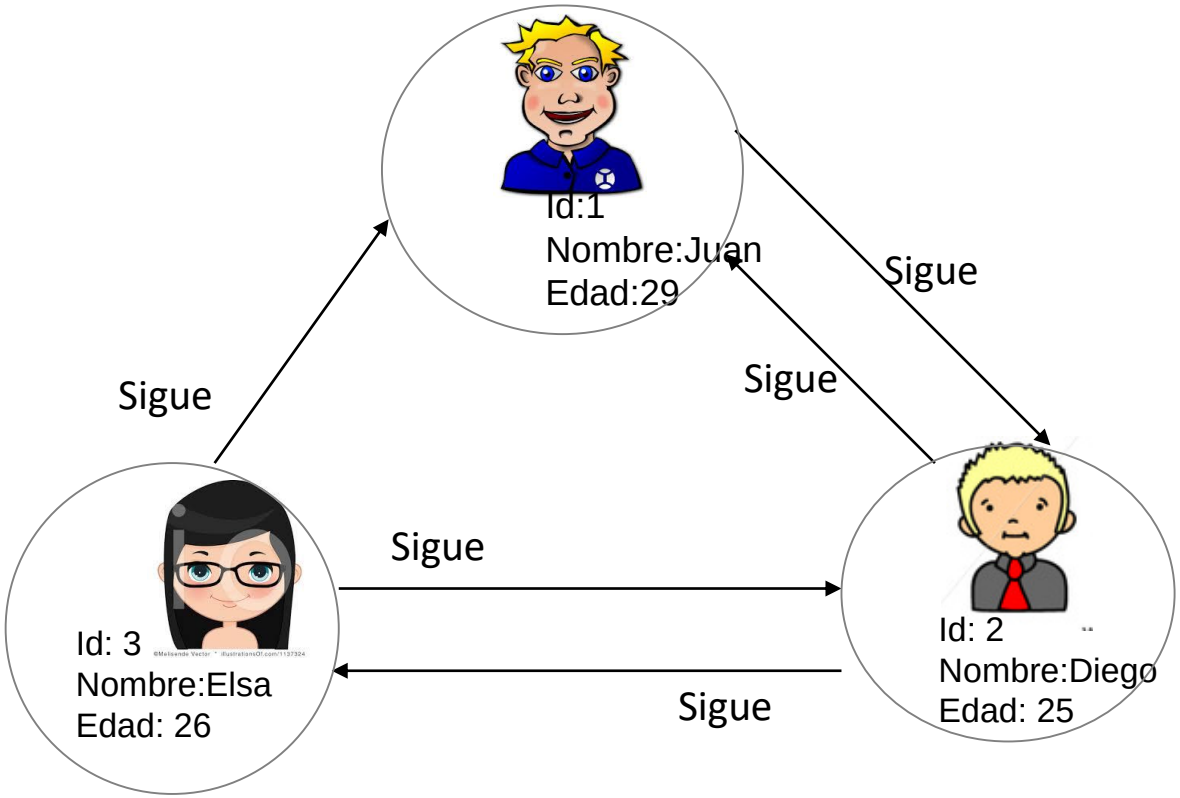
Estructura de datos, que se compone de:

- Nodos → Entes (personas, lugares, cosas, etc)
- Propiedades → Información de los nodos y enlaces
- Enlace → Relaciones



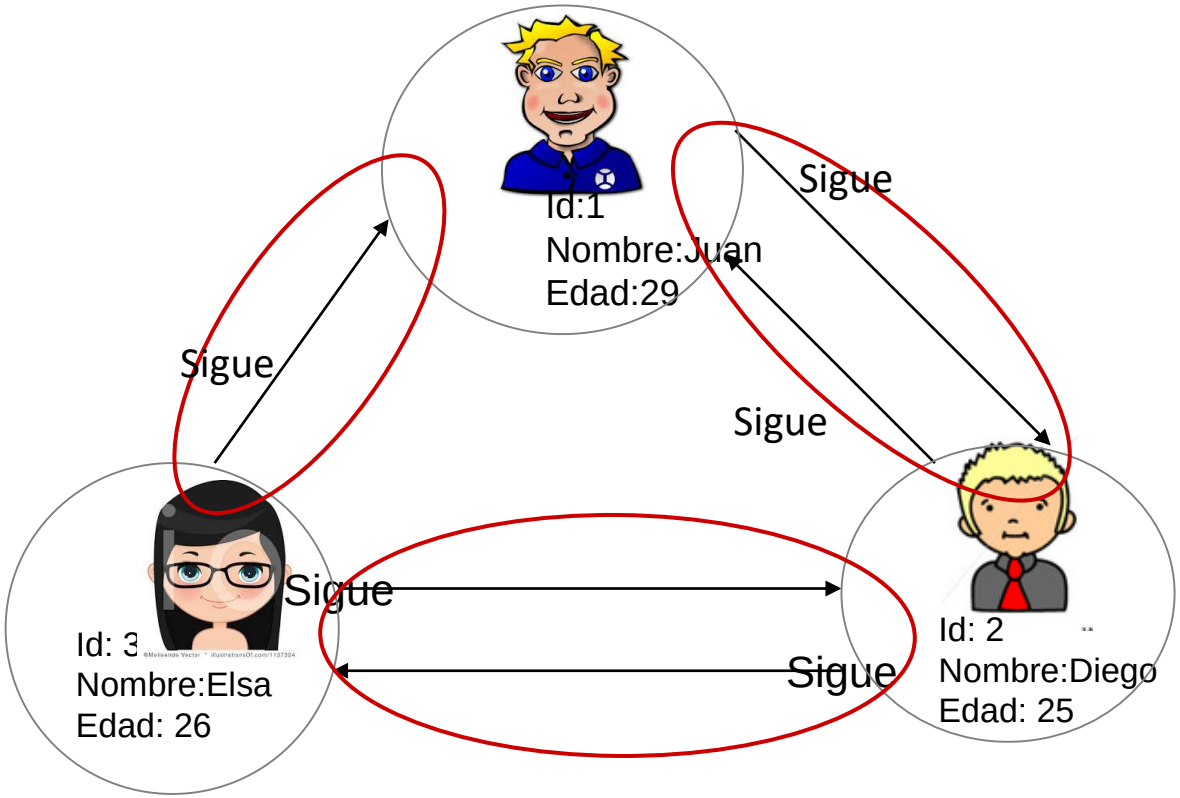
Grafo

Nodos tienen propiedades: (key,value)



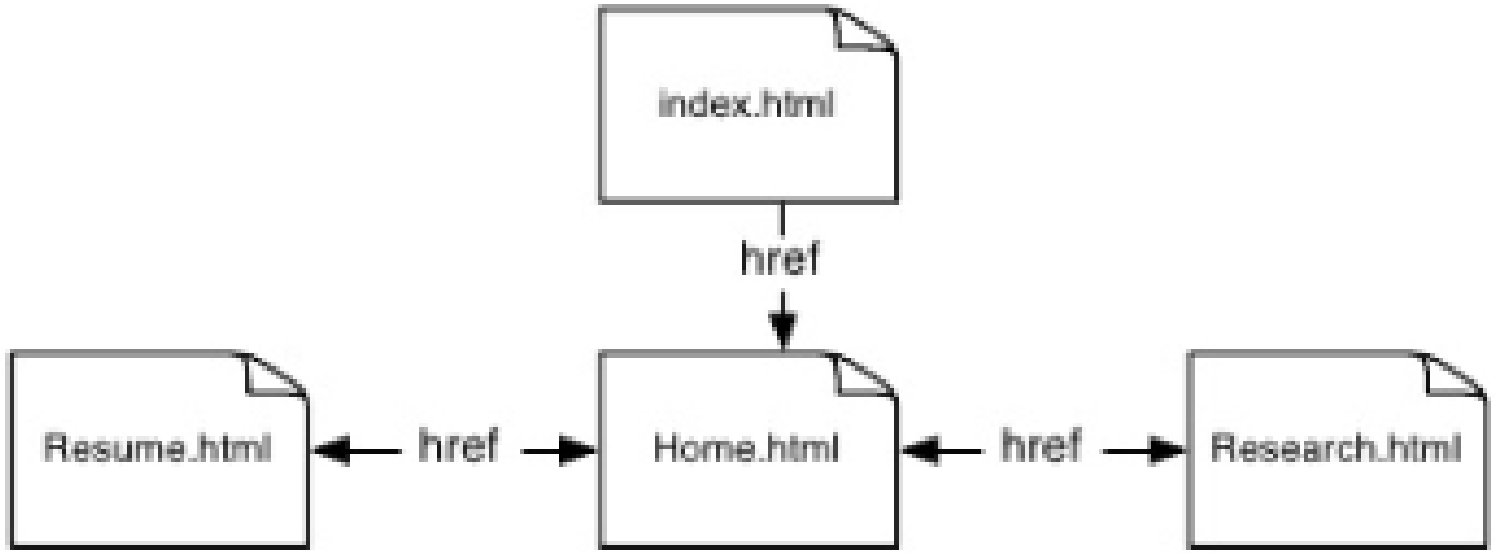
Grafo

Enlaces tienen nombre y dirección.
Siempre tiene un nodo inicial y final.

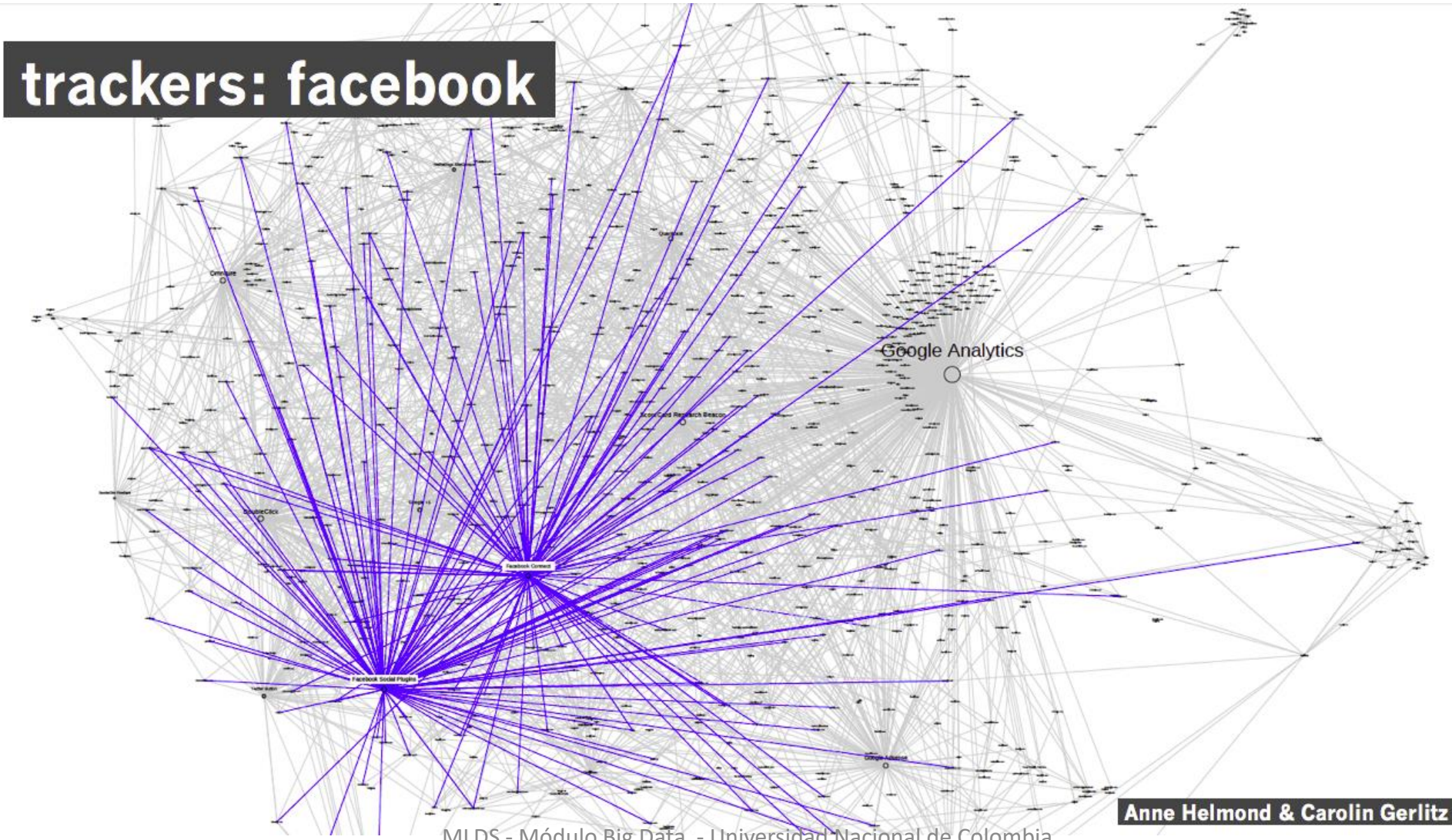


Ejemplo: La Web

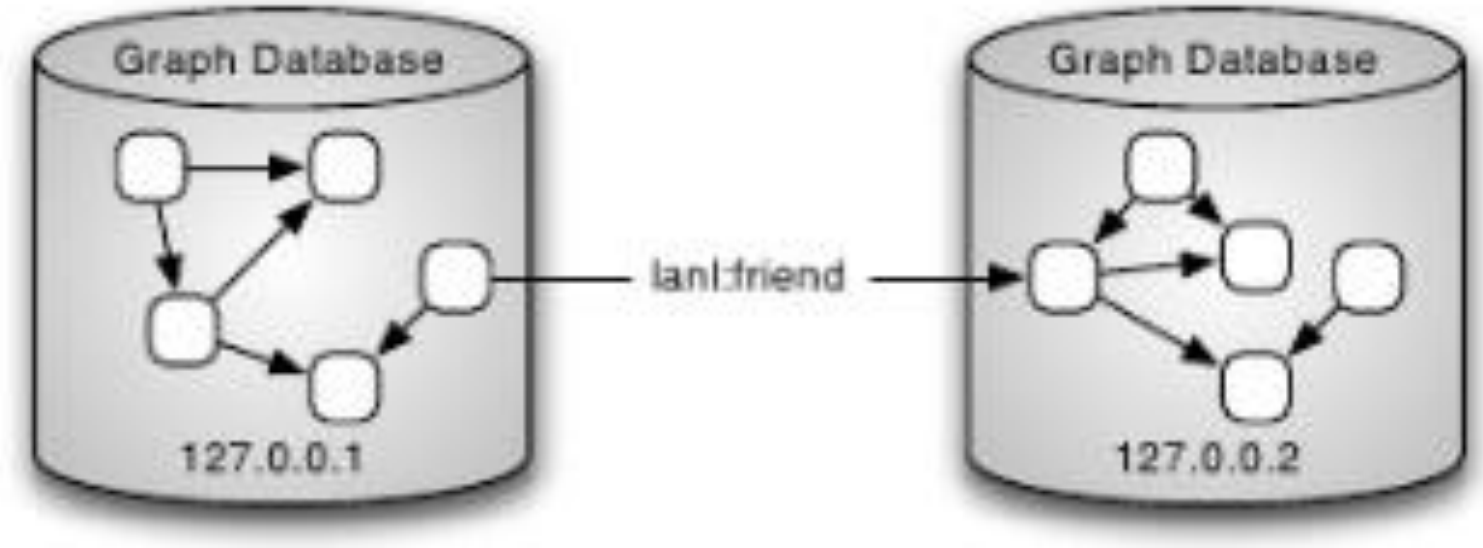
- Documentos html referencian urls.
- Recursos: documentos html, imagenes, audio, etc.



Ejemplo de grafo de la web



La Web



RDBMS a Graphs

obra

Id_obra	Nom_obra	Tipo_obra	Costo	Id_expo
111	Mona lisa	Pintura	$1,000.00	1003
112	Ultima cena	Pintura	$800.00	1003
113	Hombre vitruvio	Boceto	$400.00	1003
114	Planos	Planos	$200.00	1003
200	Fornarina	Pintura	$400.00	1004
201	David	Escultura	$700.00	1004
202	Nacimiento venus	Pintura	$250.00	1004
300	Violin and candlestick	Pintura	$300.00	1005
301	Les demoiselles d'avignon	Pintura	$350.00	1005
302	Cabeza de mujer	Escultura	$300.00	1005
400	Autorretrato	Pintura	$100.00	1006
401	La parade	Pintura	$300.00	1006
402	Solei levant	Pintura	$300.00	1006

exposición

Id_expo	Nom_expo
1003	Da Vinci
1004	Renacimiento
1005	Cubismo
1006	Impresionismo

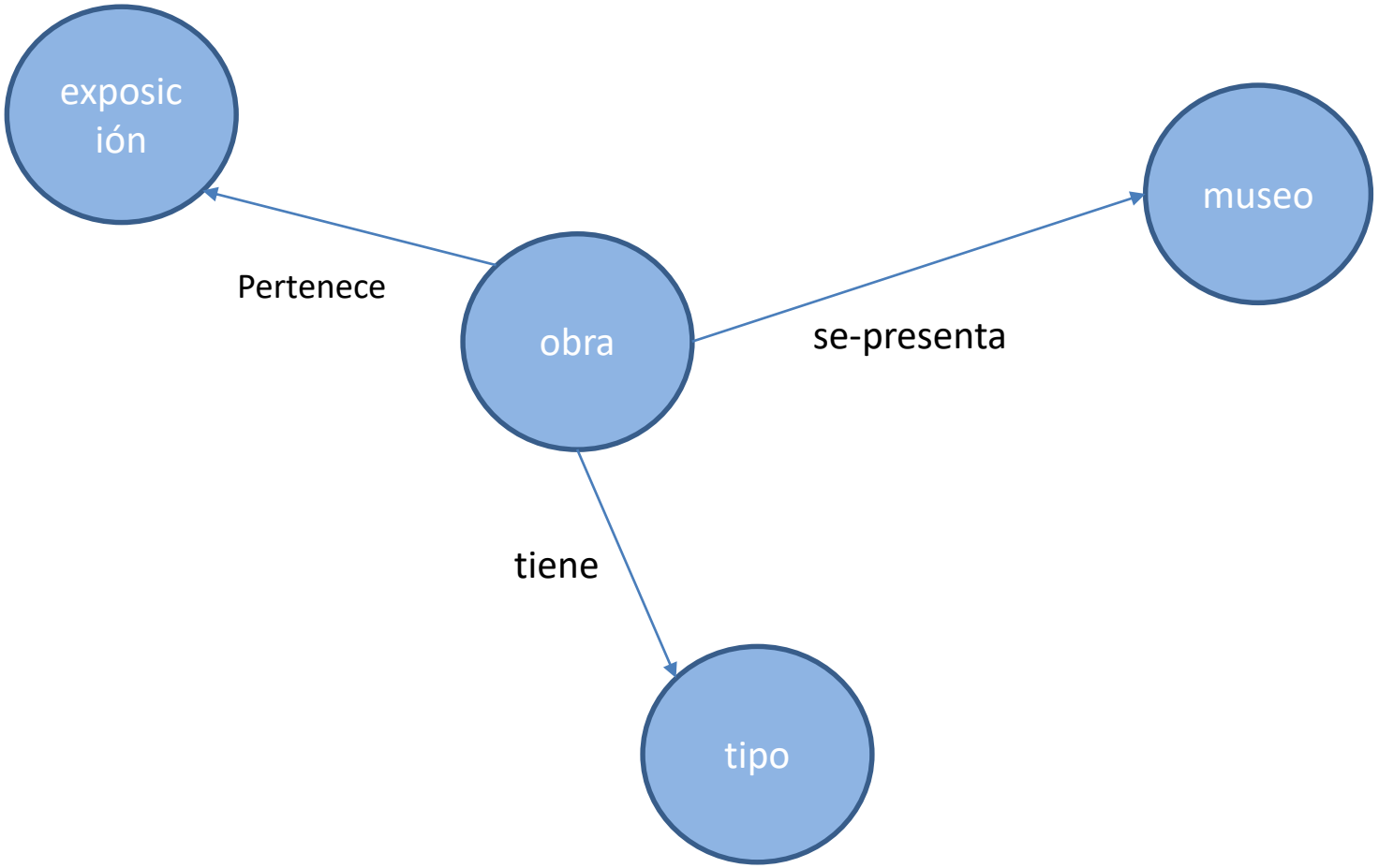
museo

Id_museo	Nom_museo
101	Louvre
102	Met
205	Shangai
304	Británico

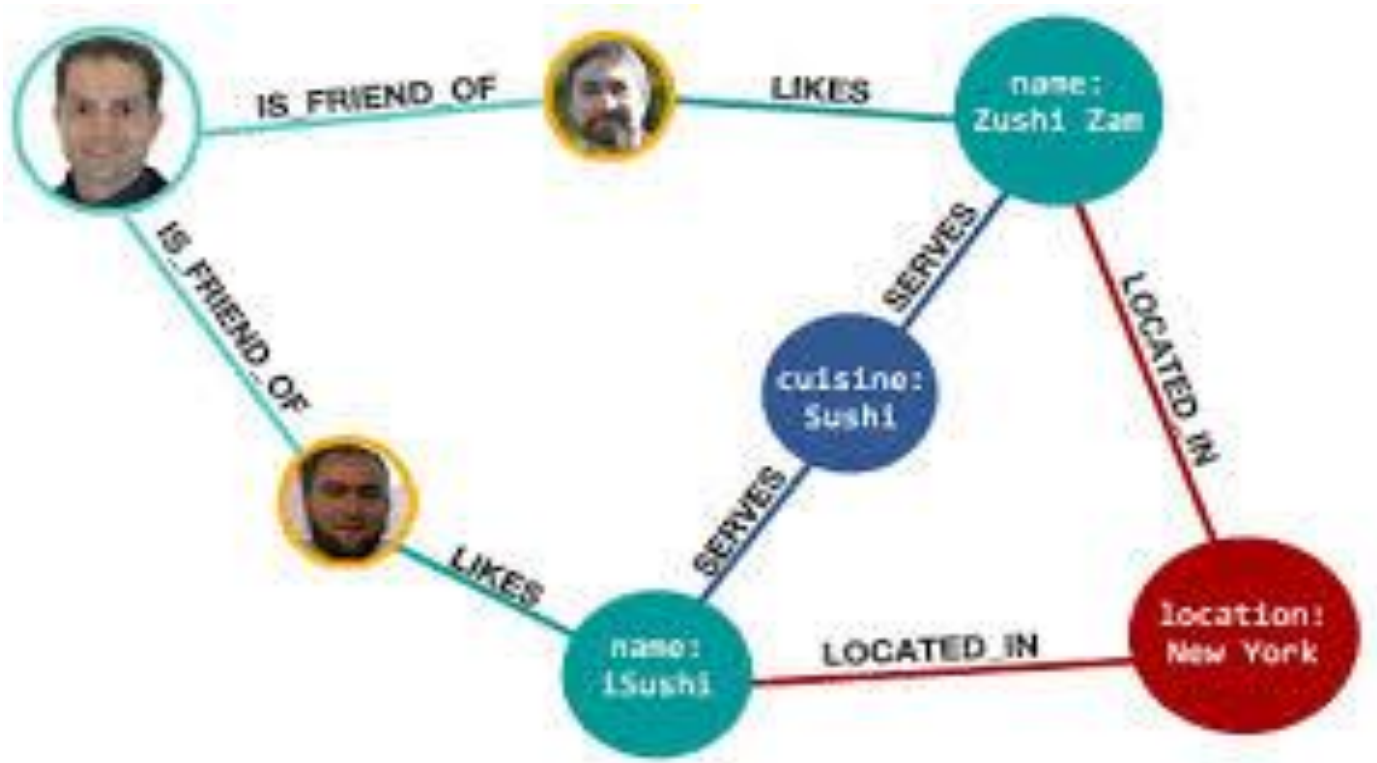
presentación

id	Id_obra	Id_museo
1	111	101
2	111	101
3	112	101
4	200	101
5	201	101
6	113	101
7	300	102
8	301	102
9	113	102
10	113	102
11	112	102
12	112	102
13	202	102
14	201	102
15	113	205
16	114	205
17	401	205
18	402	205

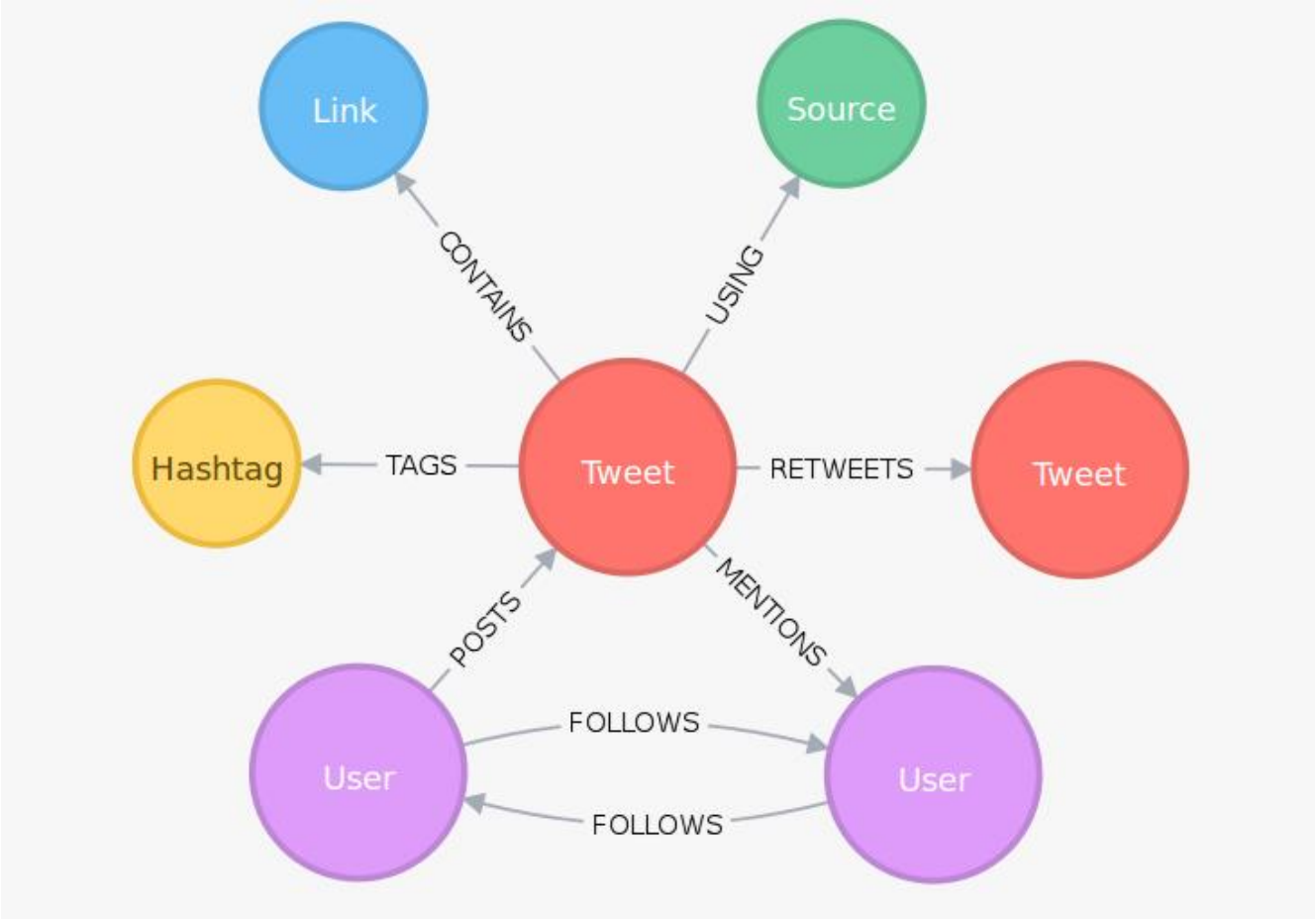
RDBMS a Graphs



Ejemplo de modelamiento basado en grafos

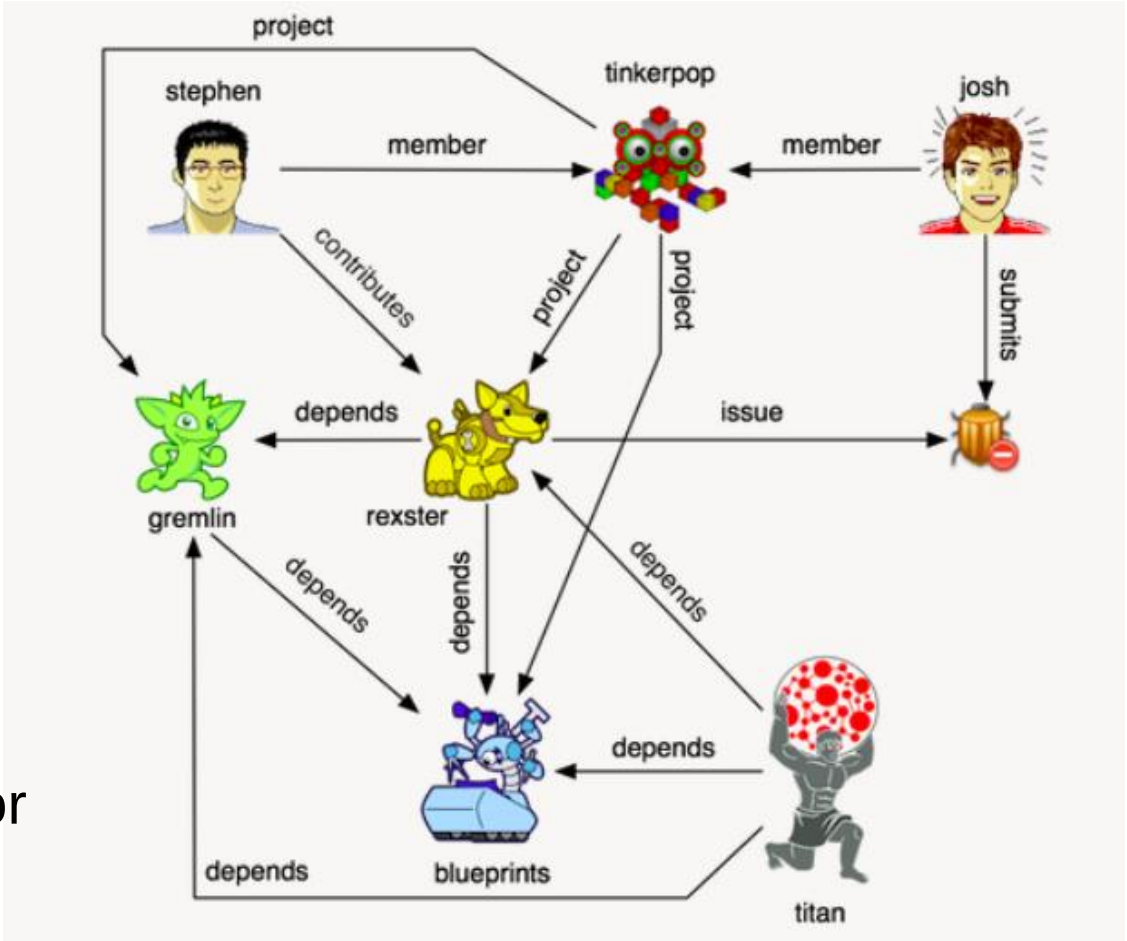


Ejemplos de modelamiento basado en grafos



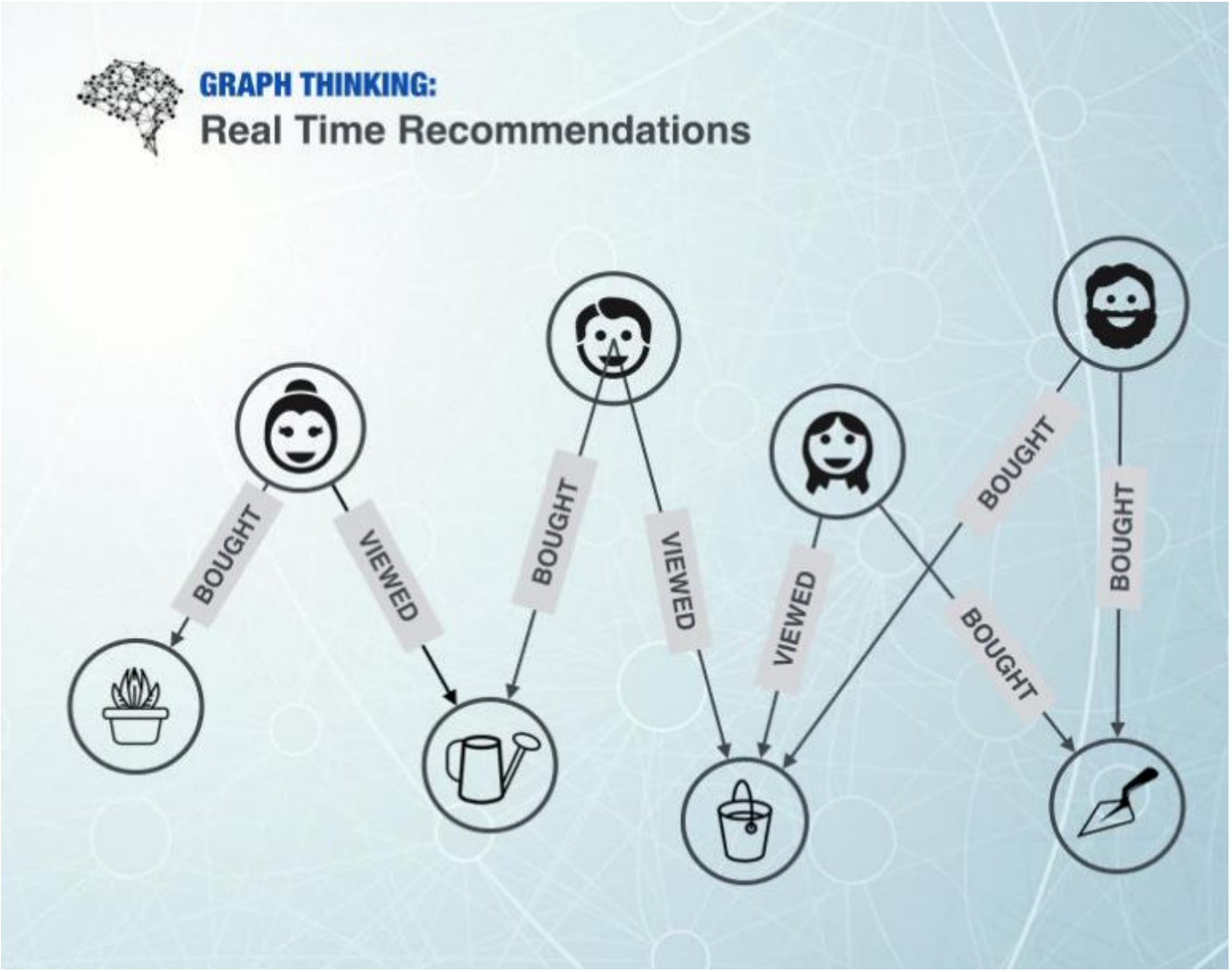
Ejemplos de modelamiento basado en grafos

Grafo de desarrollo de software:
Ambiente
de programación colaborativa.
Nodos son personas, grupos, proyectos y “tickets. Las relaciones son dependencias, pertenencia, y generador



https://markorodriguez.com

Aplicaciones



Aplicaciones

CISCO

Neo4j is the heart of Cisco's Helpdesk Solution too.

MENU

FAST COMPANY

TECHNOLOGY | LEADERSHIP | MAGAZINE | MOST INNOVATIVE COMPANIES | MOST CREATIVE PEOP

3 MINUTE READ | MIND AND MACHINE

NASA Is Harnessing Graph Databases To Organize Lessons Learned From Past Projects

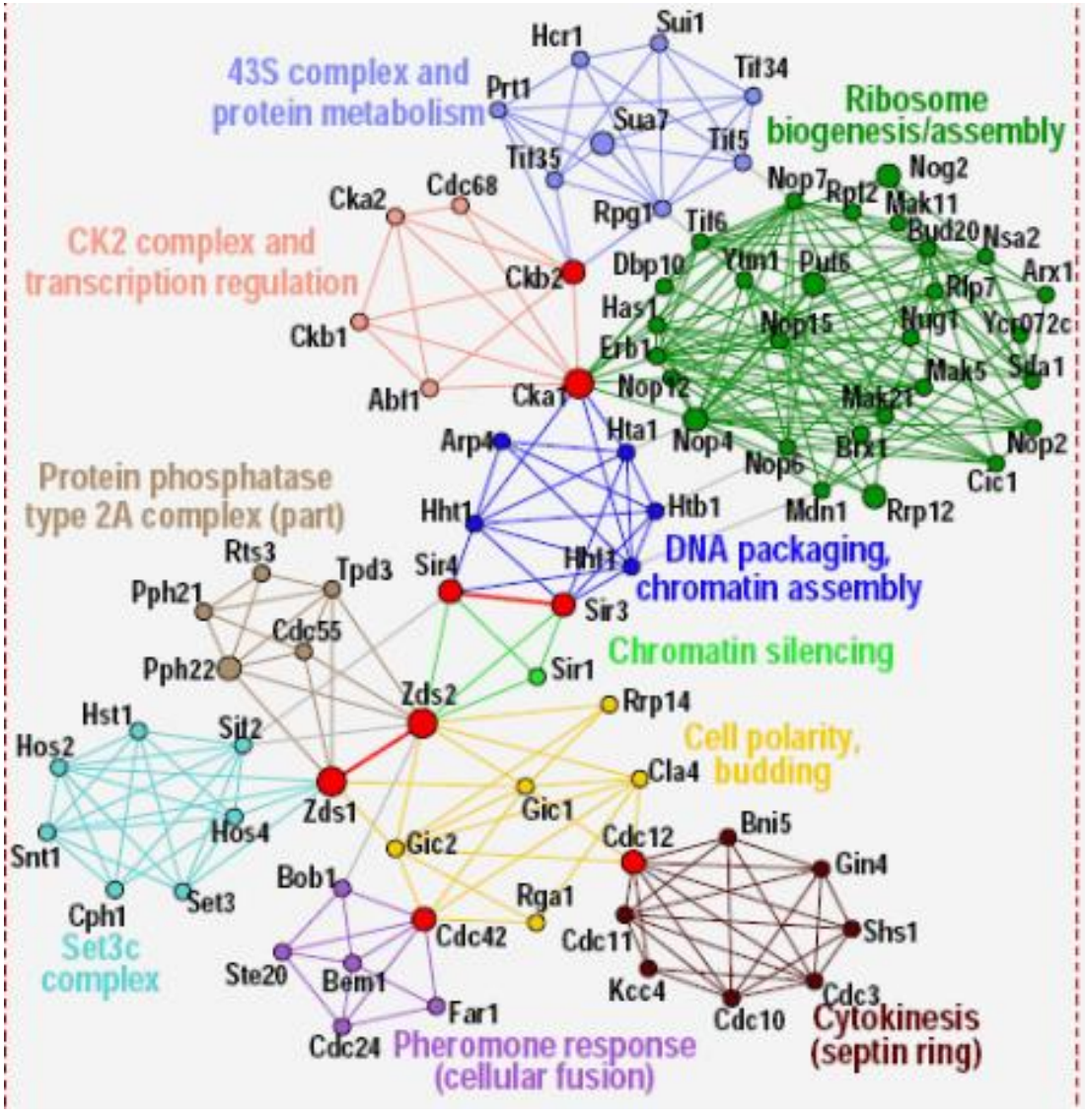
The space agency has a new tool to discover unexpected patterns in past projects.

[PHOTO: FLICKR ARCHIVE SAN DIEGO AIR AND SPACE MUSEUM]

BY STEVEN MELENDEZ 11.02.16 | 7:30 AM

https://www.fastcompany.com/3065044/nasa-is-harnessing-graph-databases-to-organize-lessons-learned-from-past-pr

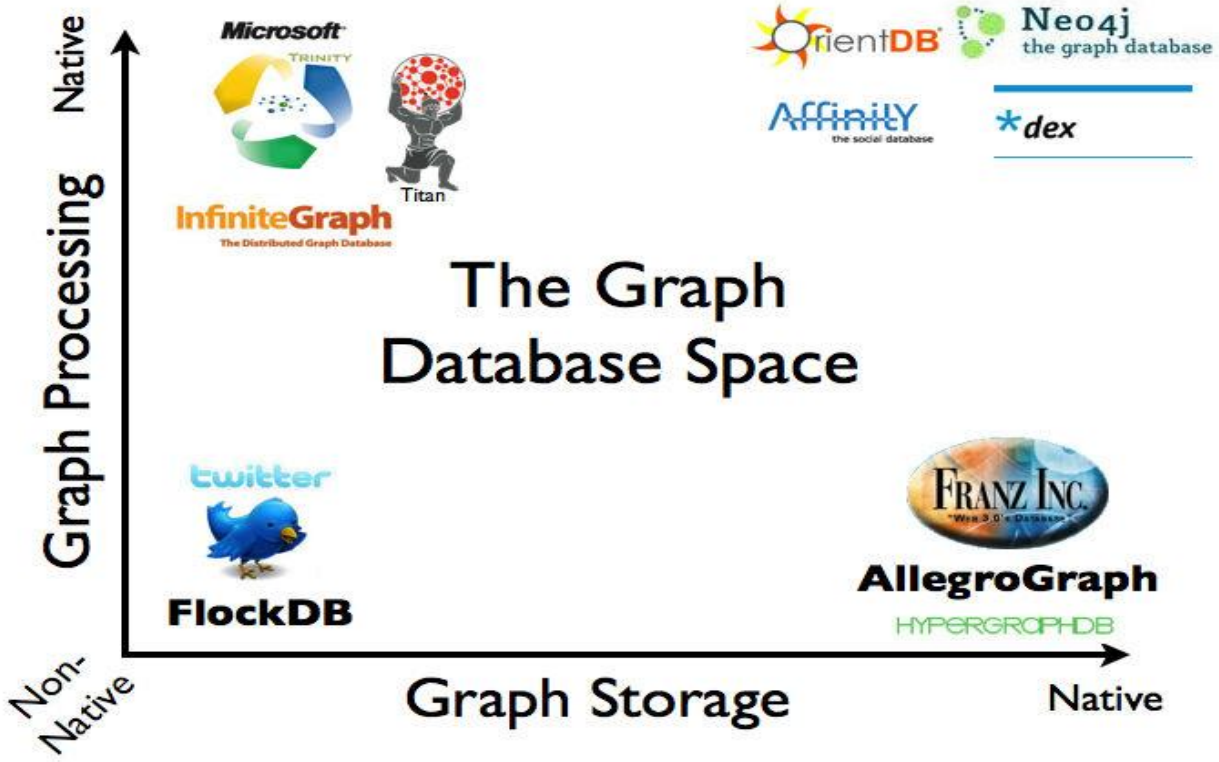
Aplicaciones



Aplicaciones

- Grafos Grandes
  - Internet
  - Detección de Comunidades
  - Herramientas de visualización y análisis de grafos:
    - Gephi (https://vimeo.com/9726202)
    - JUNG (Java University Network/Graph)
- Grafos pequeños
  - Química
    - CFinder

Software



https://towardsdatascience.com/the-king-that-graph-theory-discovered-8cce31a3cd26

Almacenamiento y Búsqueda

- Mecanismos de almacenamiento:
  1. Dependen de un modelo relacional. Se almacena en tablas (tablas lógicas).
  2. Key-value. Orientado a documentos
  3. Adicional el concepto de Tags (propiedades)
- Buscar en una bd basada en grafos requiere de un lenguaje de consulta:
  - Propio del producto de software
  - Estandarización:
    - SPARQL
    - Gremlin
    - Cypher

Referencias

[1] Fay Chang y otros. Bigtable: A Distributed Storage System for Structured Data. Google. 2006.

[2] Sitio Web Cassandra, http://cassandra.apache.org/doc/latest/

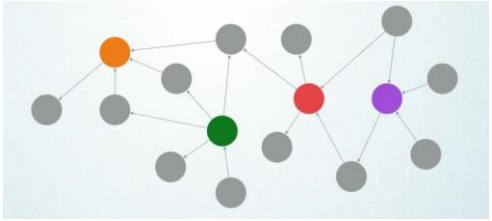
[3] J. Carpenter y Ebe Hewitt: Cassandra: The Definitive Book. O'Reilly Media (2016)

[4] DBA guide to NoSQL: http://www.datastax.com/dbas-guide-to-nosql (ebookgratuito)

[5] Getting started and user information (web page): http://docs.datastax.com/en/landing_page/doc/landing_page/current.html

[6]https://academy.datastax.com/resources/brief-introduction-apache-cassandra

References



[7] Graph Databases and the Future of Large Scale Knowledge Management. Marko A. Rodriguez. 2009

[8] Bollobas, B., 1998, Modern Graph Theory (Springer Verlag, New York, USA).

[9] J. Adibi, H. Chalupsky, M. Grobelnik, N. Milic- Frayling, and D. Mladenic. KDD Workshop on Link Analysis and Group Detection. 2004.

[10] Webinar- Neo4j - RDBMS to Graphs