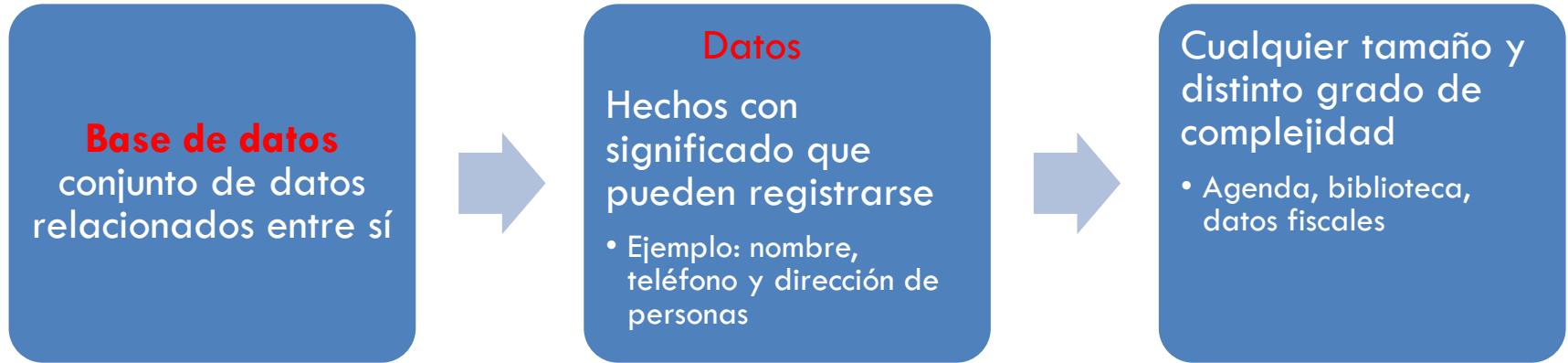


Bases de datos NoSQL

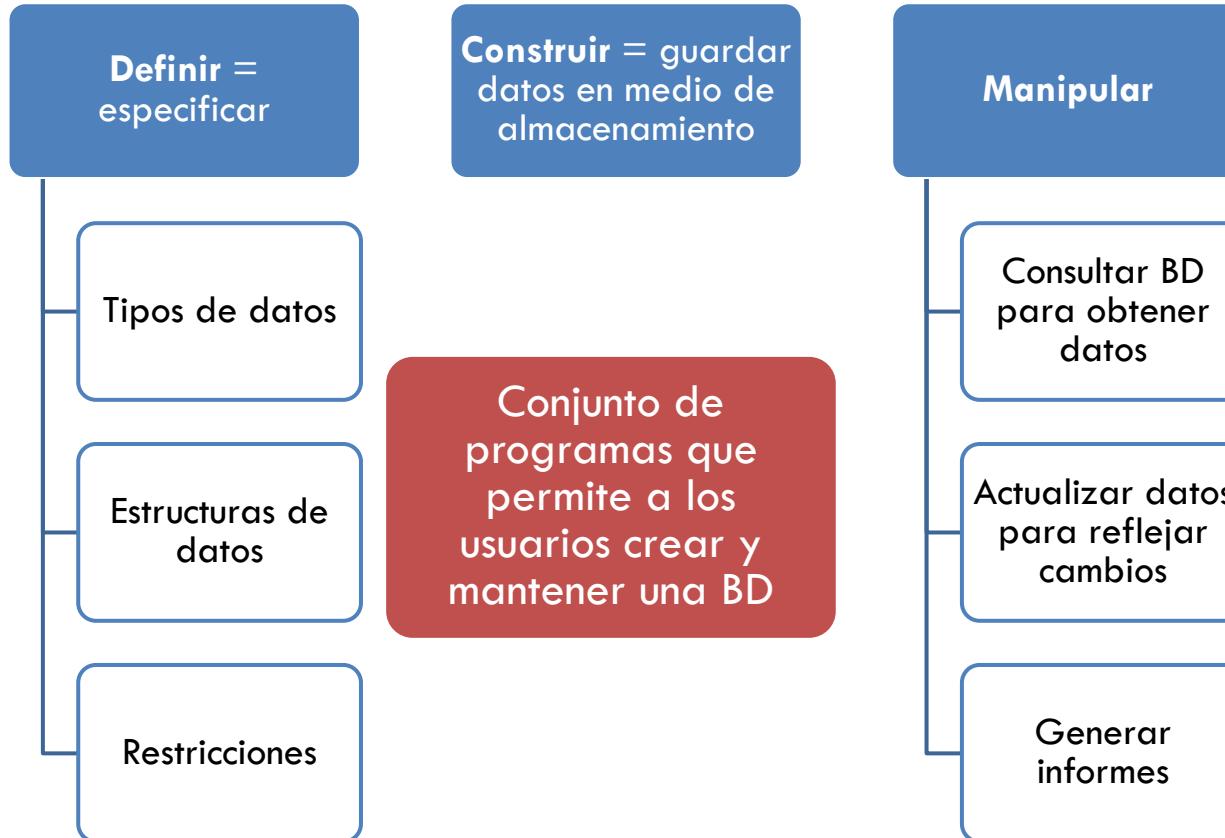
Contenidos

- ¿Qué es una base de datos?
- Motivación para BDs NoSQL
- Modelos ACID y BASE
- Consistencia eventual
- Características BDs NoSQL
- Tipos de BDs NoSQL

¿Qué es una base de datos?



Sistema de Gestión de Base de Datos



Ejemplo – BD Universidad

Definir

Especificar estructura de registros de cada archivo

Tipos de datos de cada campo

Construir

Almacenar datos que representan a cada estudiante, asignatura, etc

Cinco archivos

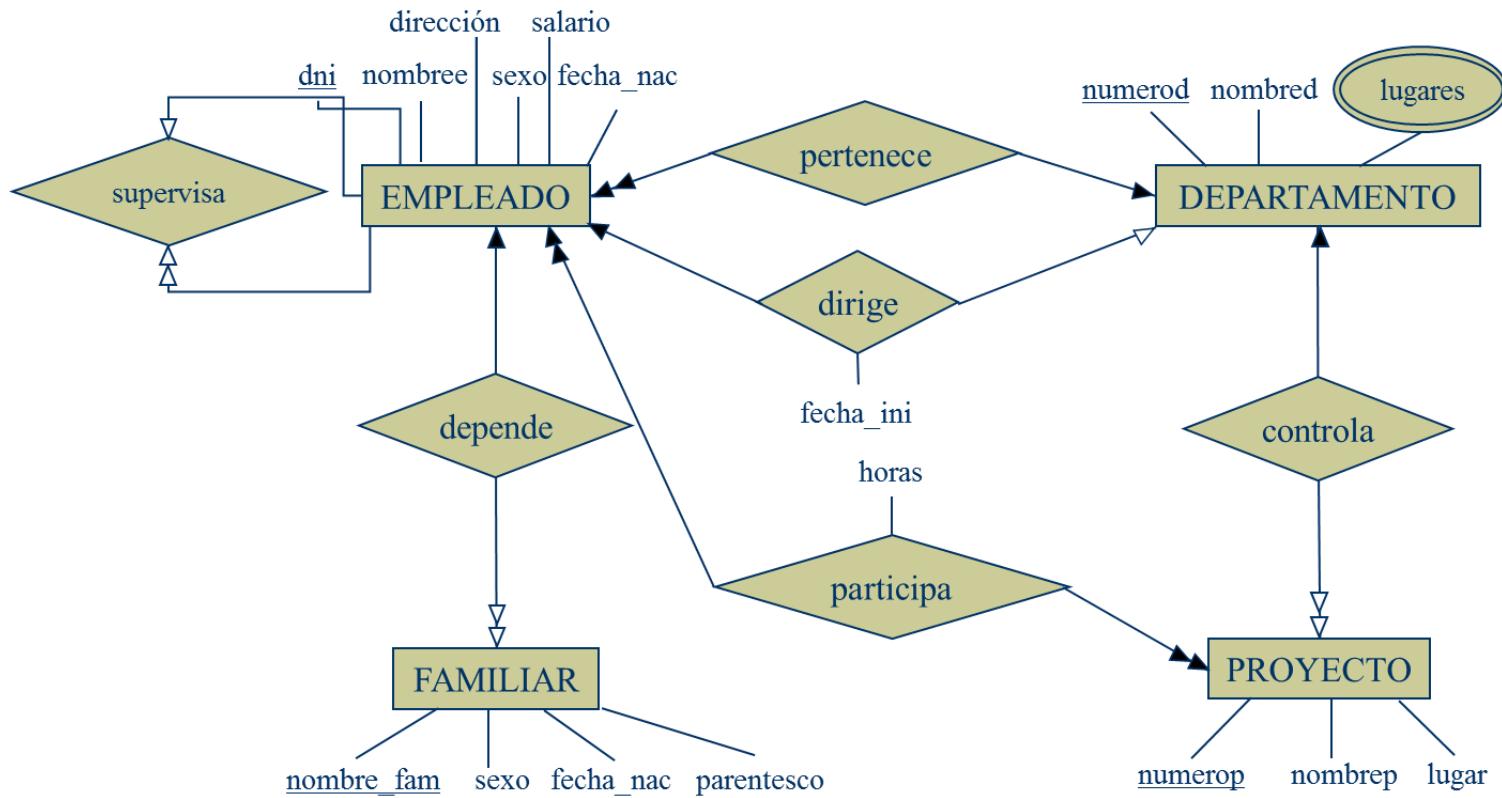
- ESTUDIANTES
- ASIGNATURAS
- NOTAS
- REQUISITOS
- GRUPOS

Manipular

Consultas: notas de Pérez, lista de estudiantes asignatura BD en 1992

Actualizaciones: cambiar curso de Pérez a 2º, crear nuevo grupo de BD

Modelo Entidad-Relación



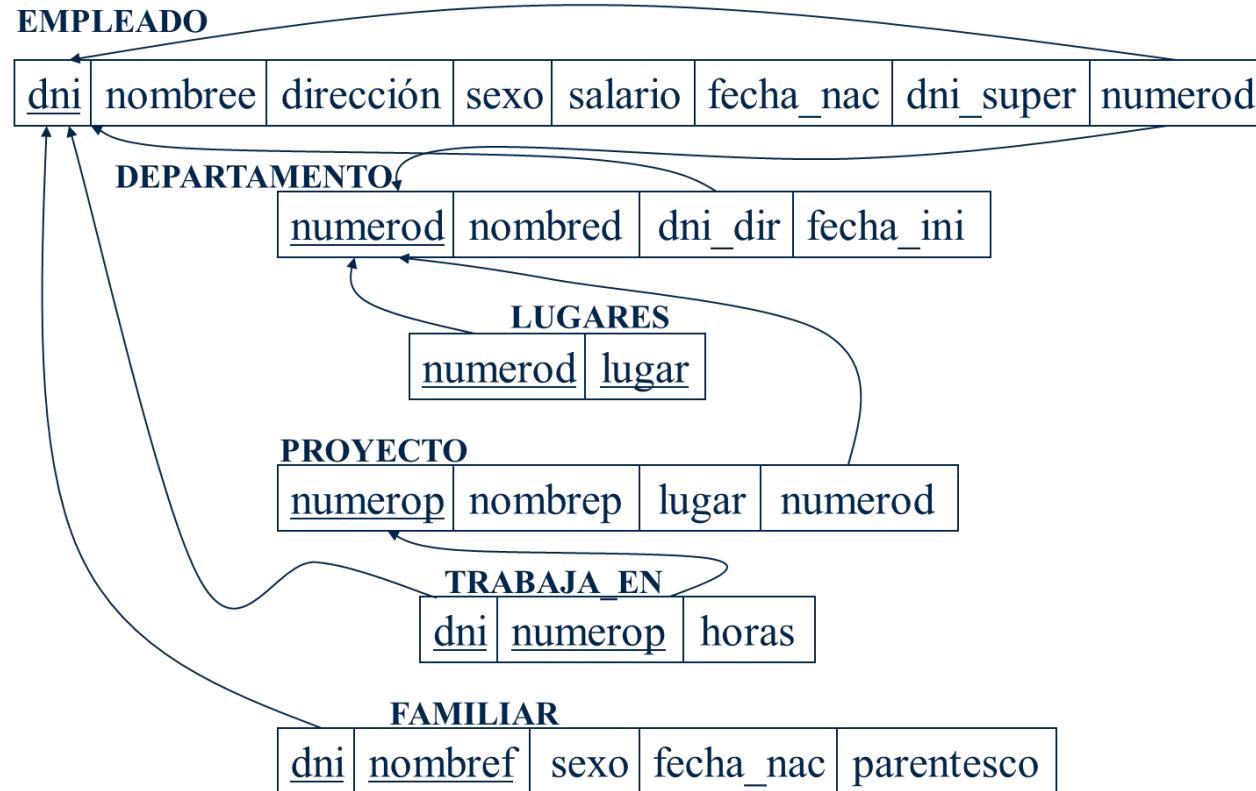
Modelo Relacional

- # Colección de tablas
- # Cada tabla tiene varias columnas

Nombre	Ciudad	N.Cuenta
Pepe	Madrid	14
Juan	Madrid	18
Paco	Lugo	14

N.Cuenta	Saldo
14	1500
18	2000

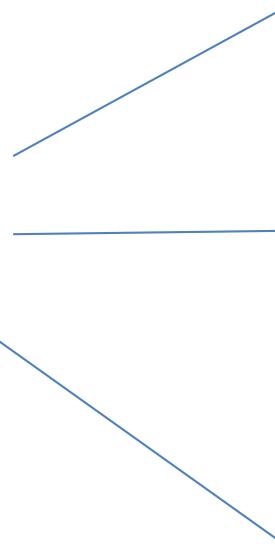
Ejemplo



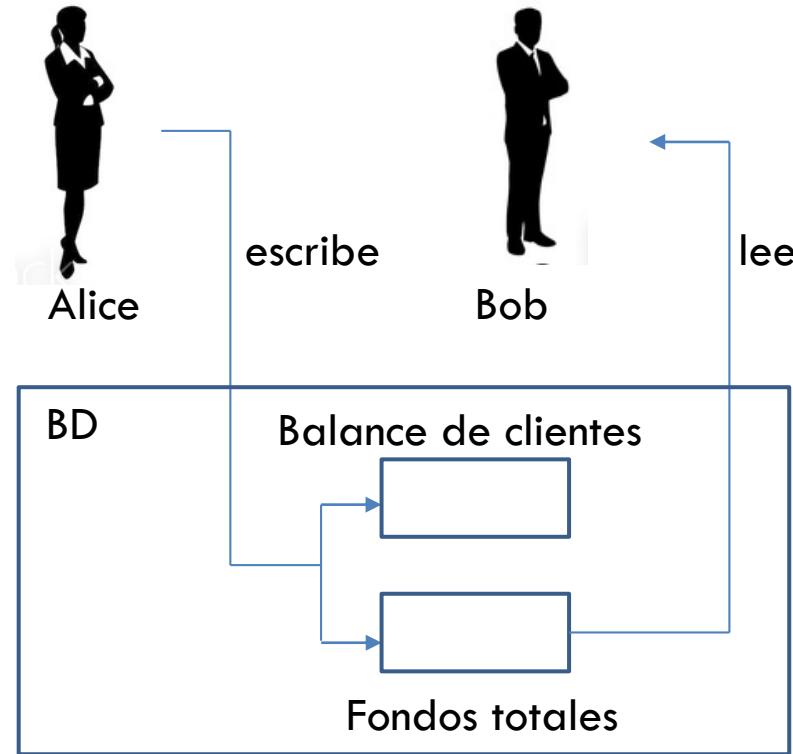
Un SGBD debe hacer tres cosas



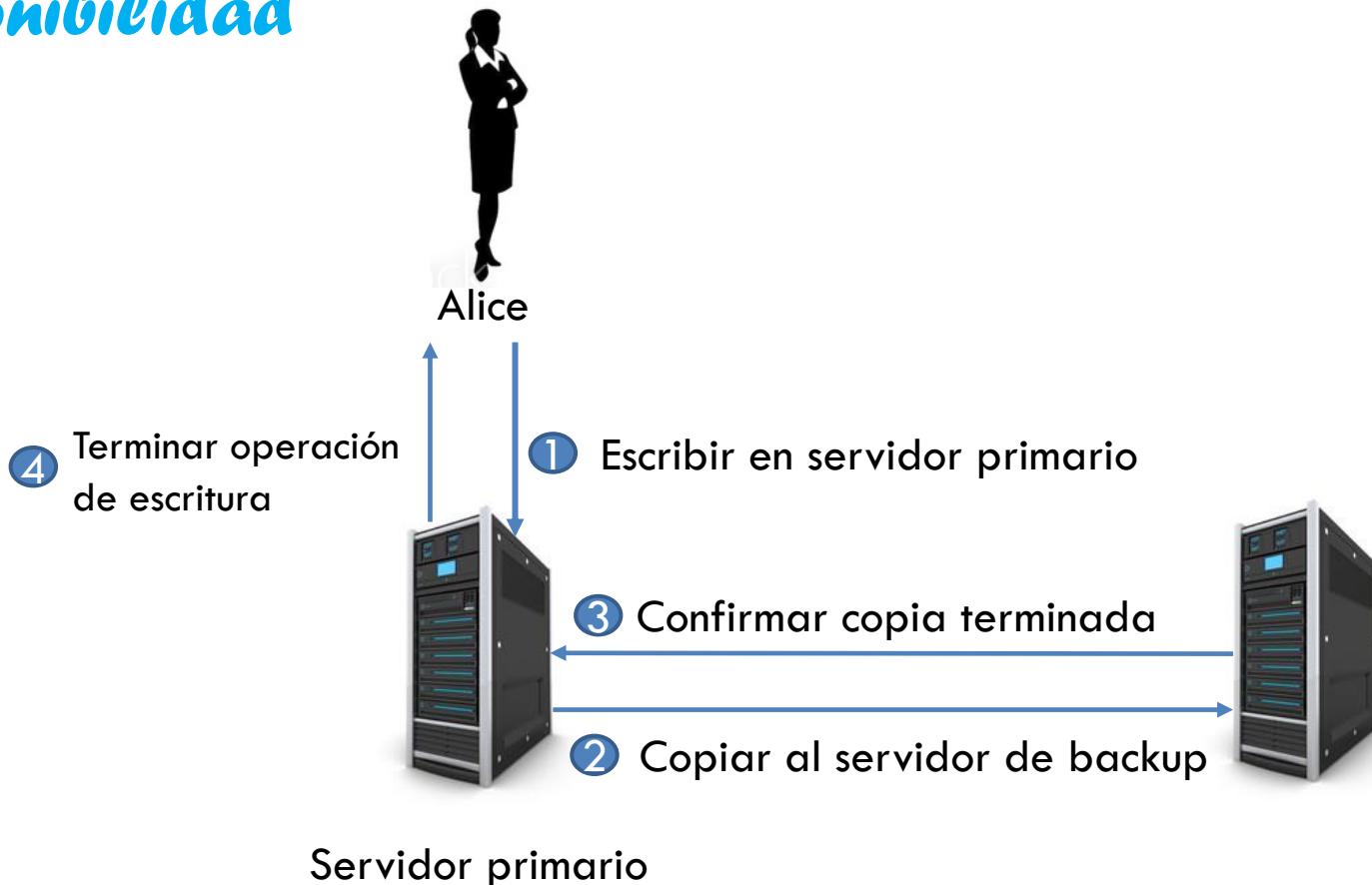
Persistencia



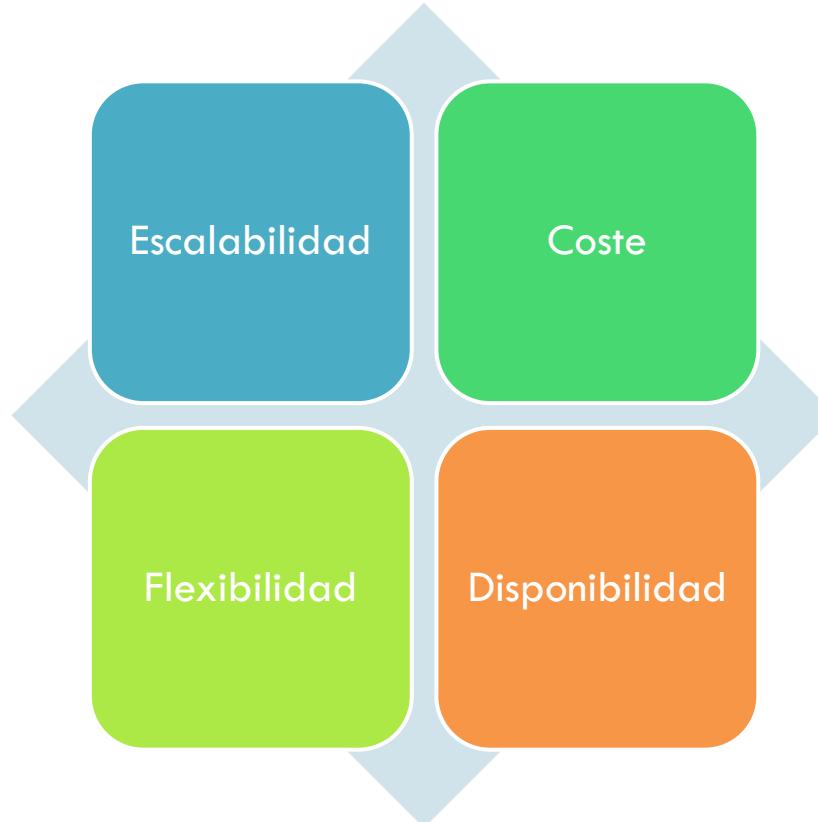
Consistencia



Disponibilidad



Motivación para BDs NoSQL



Escalabilidad



Scale up

Scale out

Coste

- Licencias en función del número de usuarios
- Aplicaciones web => ¿cuántos usuarios?
- NoSQL => open source

Flexibilidad

- En BD relacionales
 - los diseñadores saben desde el principio tablas y columnas
 - La mayoría de las columnas aparecen en todas las filas
- Aplicación de comercio electrónico con BD para productos
 - Ordenadores - tipo de CPU, memoria, tamaño de disco, ...
 - Microondas - tamaño, potencia,...
 - Una tabla por producto o NoSQL

Disponibilidad



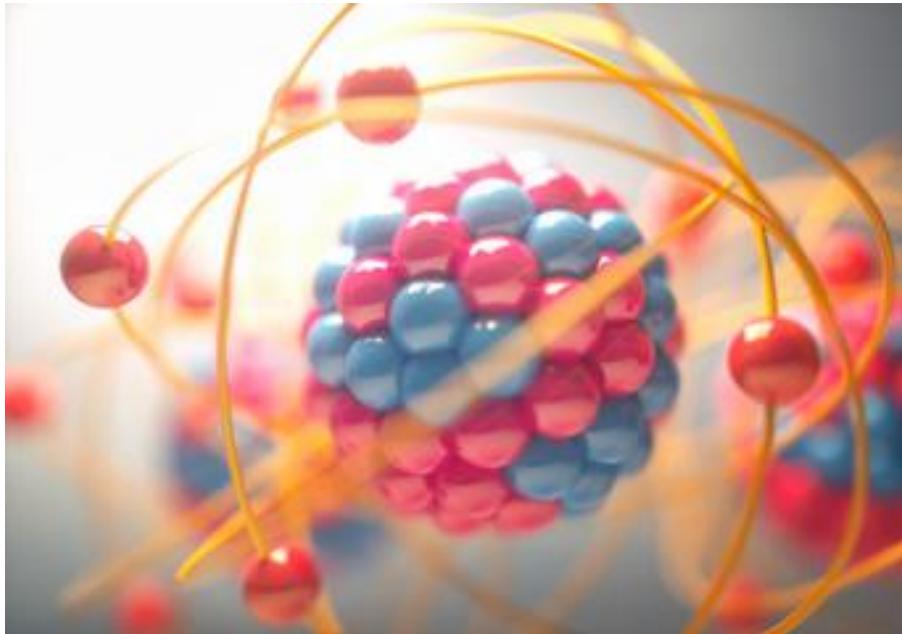
ACID

- Atomicity
- Consistency
- Isolation
- Durability

BASE

- Basically Available
- Soft state
- Eventually consistent

Atomicidad



- En una transacción es necesario completar todos los pasos o ninguno
- El conjunto de operaciones que la forman es indivisible

Consistencia

- Una transacción no deja una base de datos en un estado que viola la integridad de los datos



Aislamiento



- Las transacciones no son visibles a otros usuarios hasta que se completa su ejecución

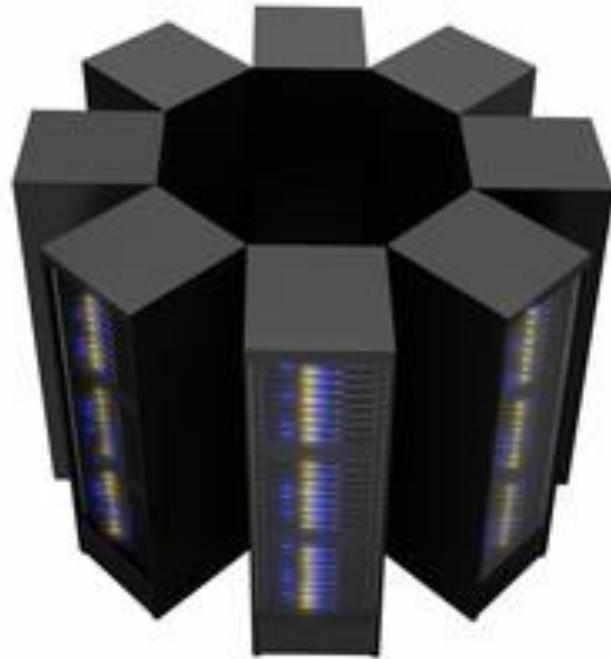
Durabilidad



- Una vez que se complete una transacción, el efecto permanecerá incluso si se cae el servidor.
- Esto implica que los datos se almacenan en disco o cualquier otro medio persistente.

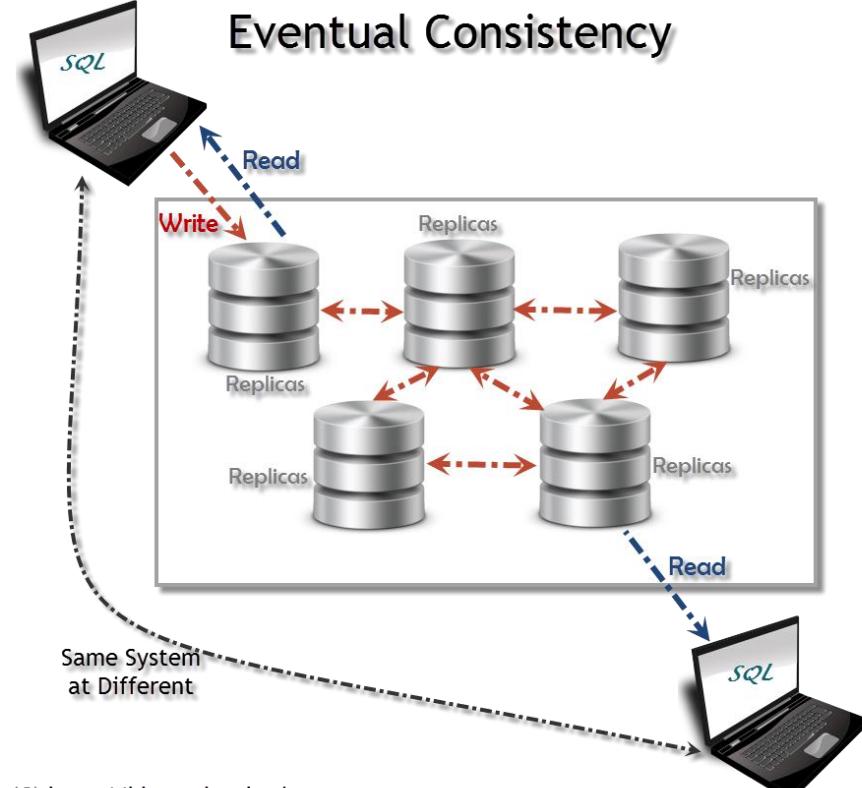
Basically available

- Puede haber fallos parciales en algunas partes del sistema distribuido
- Las BDs NoSQL guardan varias copias en servidores diferentes



Eventually consistent

- La BD puede estar en un estado inconsistente en algunas ocasiones
- En algún momento, el mecanismo de actualización hará desaparecer las inconsistencias



(C) <http://blog.sqlauthority.com>

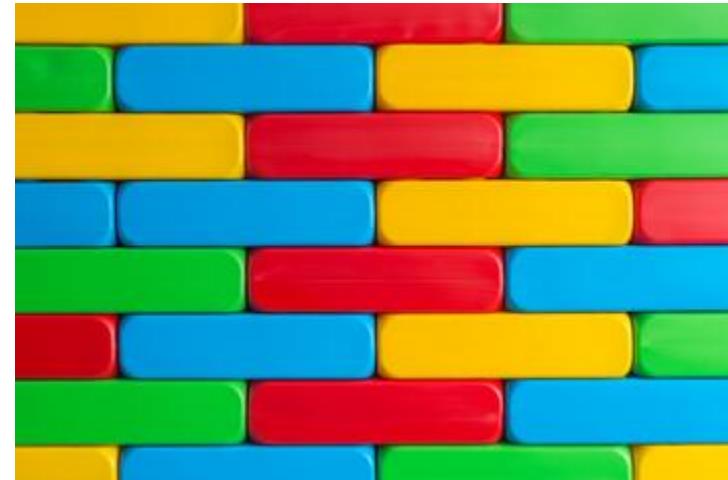
Soft state

- El estado puede cambiar con el tiempo

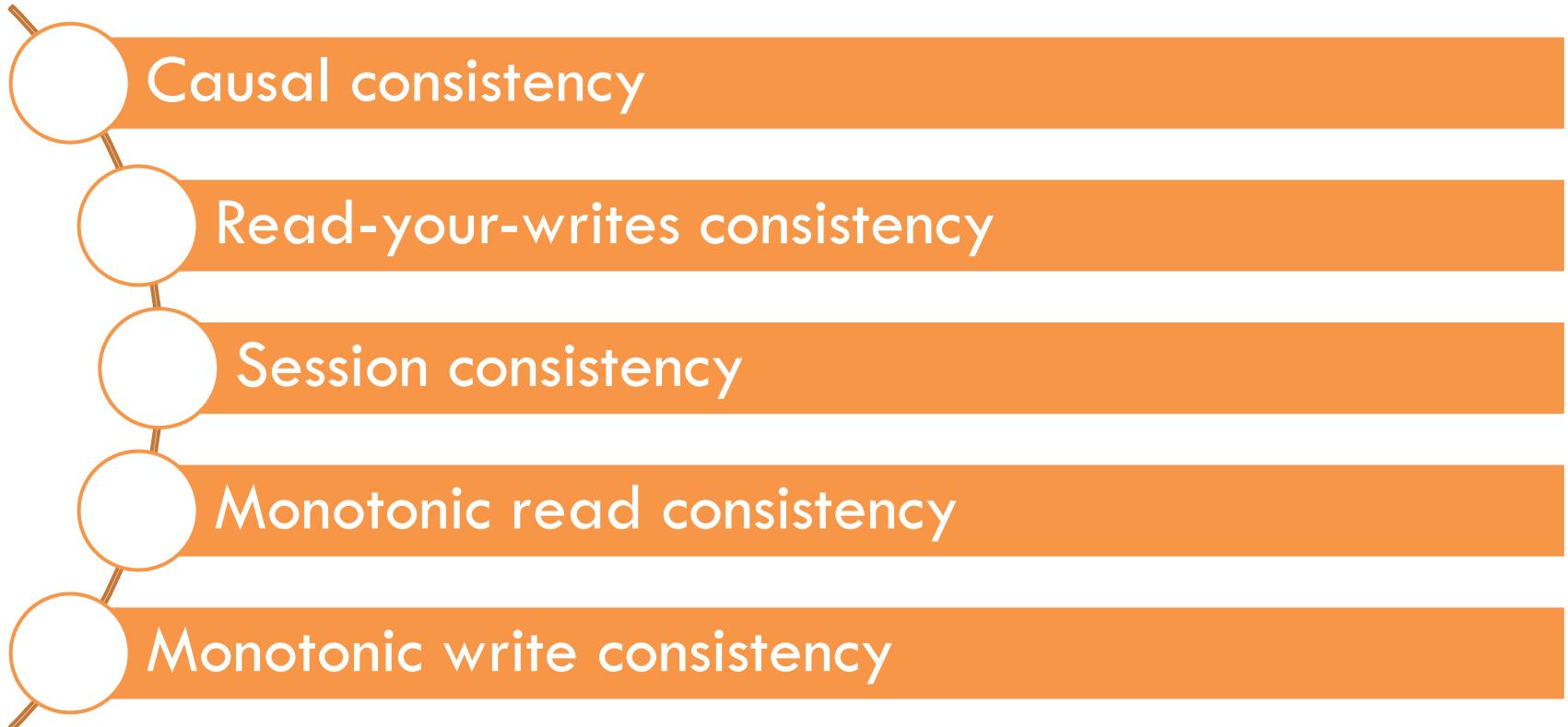
Soft



Hard

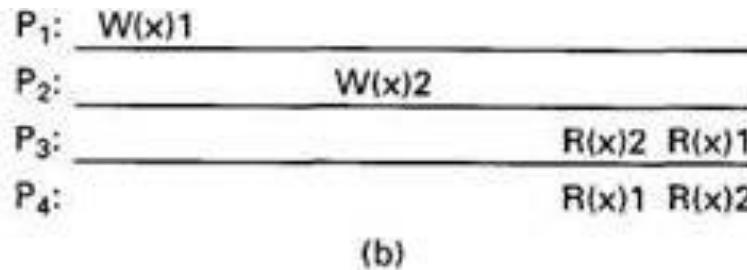


Tipos de consistencia eventual



Causal consistency

- Las escrituras potencialmente relacionadas causalmente deben ser vistas por todos los procesos en el mismo orden
- Operaciones concurrentes = no relacionadas causalmente
- Escritura después de lectura pueden estar relacionadas causalmente



W(x)1 y W(x)2 son concurrentes => P3 y P4 pueden verlas en diferente orden

Causal consistency

P ₁ :	W(x)1
P ₂ :	R(x)1 W(x)2
P ₃ :	R(x)2 R(x)1
P ₄ :	R(x)1 R(x)2

(a)

W(x)2 puede ser causalmente dependiente de W(x)1 porque el 2 puede ser resultado de un cálculo en el que intervenga el valor leído por R(x)1 => P3 y P4 deben verlas en el mismo orden => (a) incorrecto

Read-your-writes consistency

- Cuando actualizas un registro, todas tus lecturas de ese registro devuelven el valor actualizado

1. Alice cambia el saldo de un cliente a 1500\$

2. El update se escribe en un servidor y empieza el proceso de réplica



Durante el proceso de réplica si Alice consulta el saldo se garantiza que verá 1500\$

Session consistency

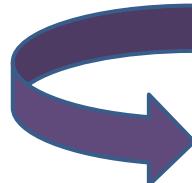
- Sesión
 - conversación entre un cliente y un servidor (usuario y BD)
 - Termina cuando haces log off
- Asegura read-your-writes consistency mientras dura la sesión

Monotonic read consistency

- Si haces una consulta y ves un resultado, en consultas posteriores nunca verás una versión anterior del valor

1. Alice cambia el saldo de un cliente a 2500\$

2. Mas tarde Bob hace una consulta de ese saldo y ve que es 2500\$



Si Bob hace la consulta de nuevo, verá 2500\$ aunque no estén actualizadas todas las copias

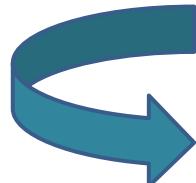
Monotonic write consistency

- Los updates se ejecutan en el orden que se solicitan

1. Alice decide aumentar el saldo en 10% a todos los clientes

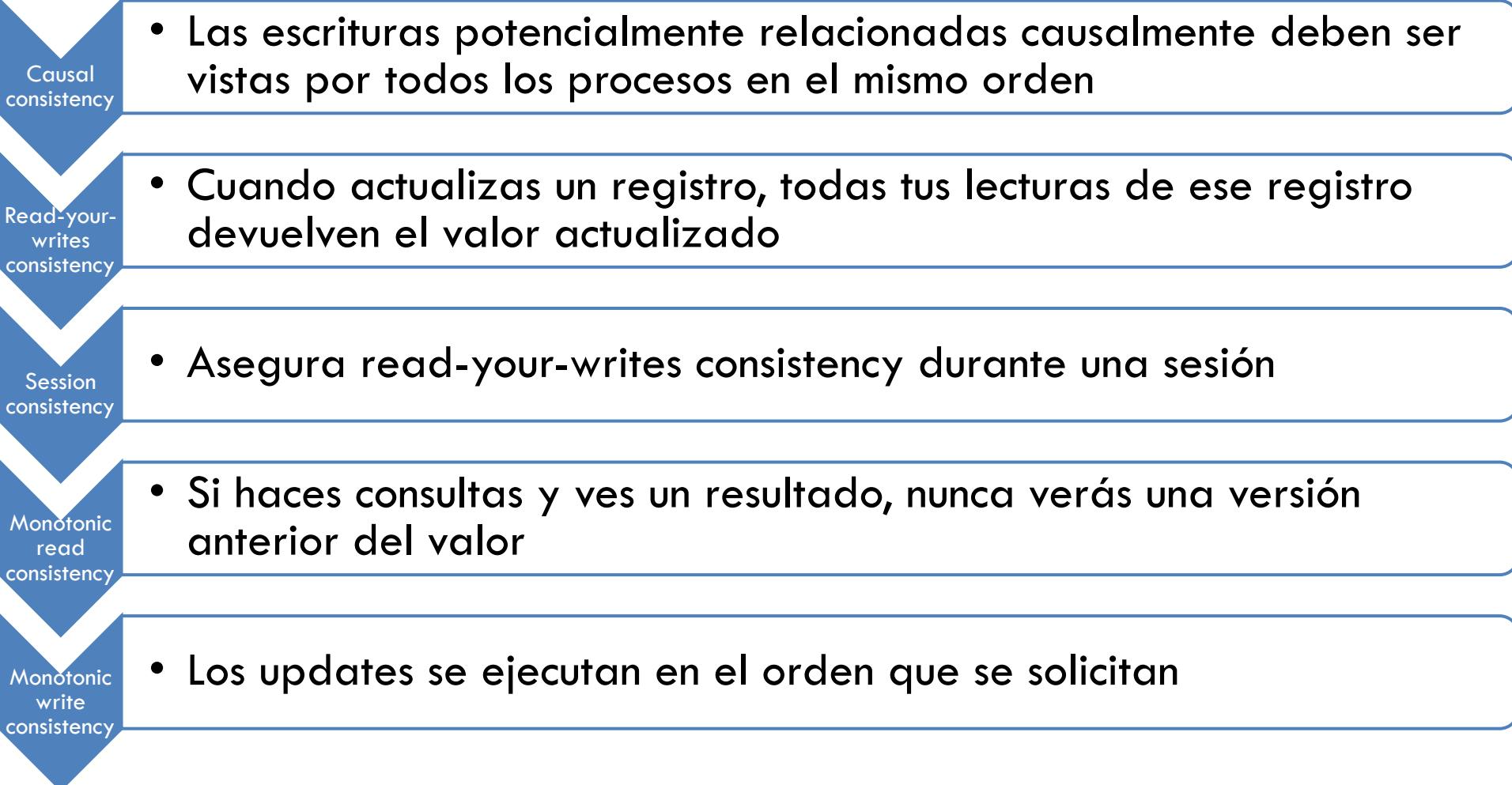
2. Charlie tiene 1000\$ => pasaría a 1100\$

3. Charlie saca 200\$ y ahora su saldo será de 900\$



Si no se mantuviera el orden

$$1000\$ - 200\$ + 10\% = 880\$$$



¿Qué es NoSQL?



NoSQL

The logo consists of the word "NoSQL" in a bold, sans-serif font. The letters "N" and "O" are colored red, while "SQL" is in black. The background of the text area is a light pink gradient.



No utilizan SQL como lenguaje de consulta

No utilizan estructuras fijas como tablas para el almacenamiento de los datos

No suelen permitir operaciones JOIN

Arquitectura distribuida

Ventajas

Datos más abiertos y flexibles

Se pueden hacer cambios de los esquemas sin tener que parar bases de datos.

Se pueden ejecutar en máquinas con pocos recursos

Escalabilidad horizontal

Optimización de consultas para grandes cantidad de datos

Tipos

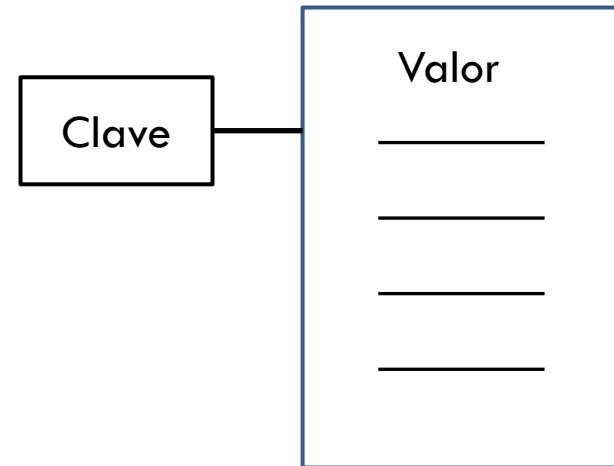
Clave-valor
REDIS

Orientadas
a documento
MONGODB

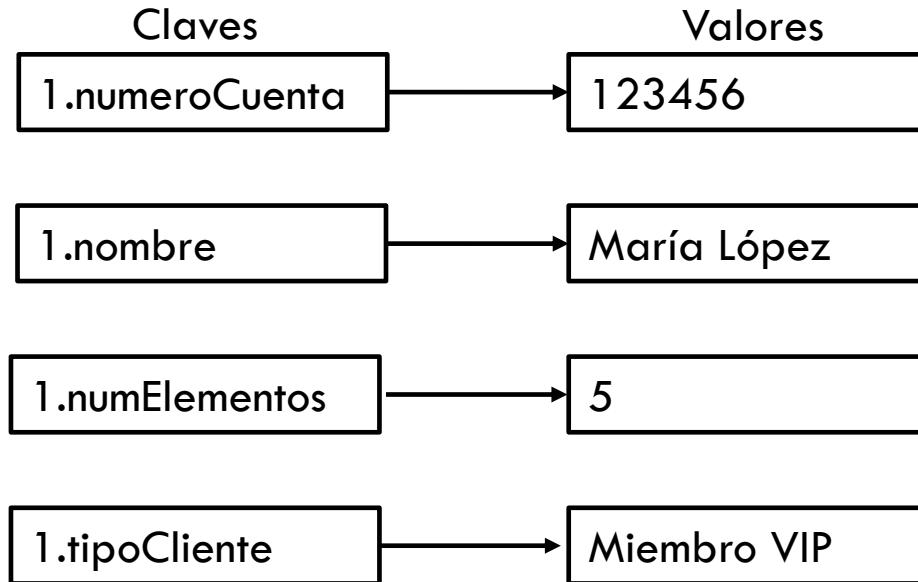
Orientadas
a columnas
CASSANDRA

Orientadas
a grafos
NEO4J

Clave-valor



Web de comercio electrónico



¿Cómo crear claves únicas?

cust1.accountNumber

cust1.name

cust1.address

cust1.numItems

cust1.custType

cust2.accountNumber

cust2.name

cust2.address

cust2.numItems

cust2.custType

wrhs1.number

wrhs1.address

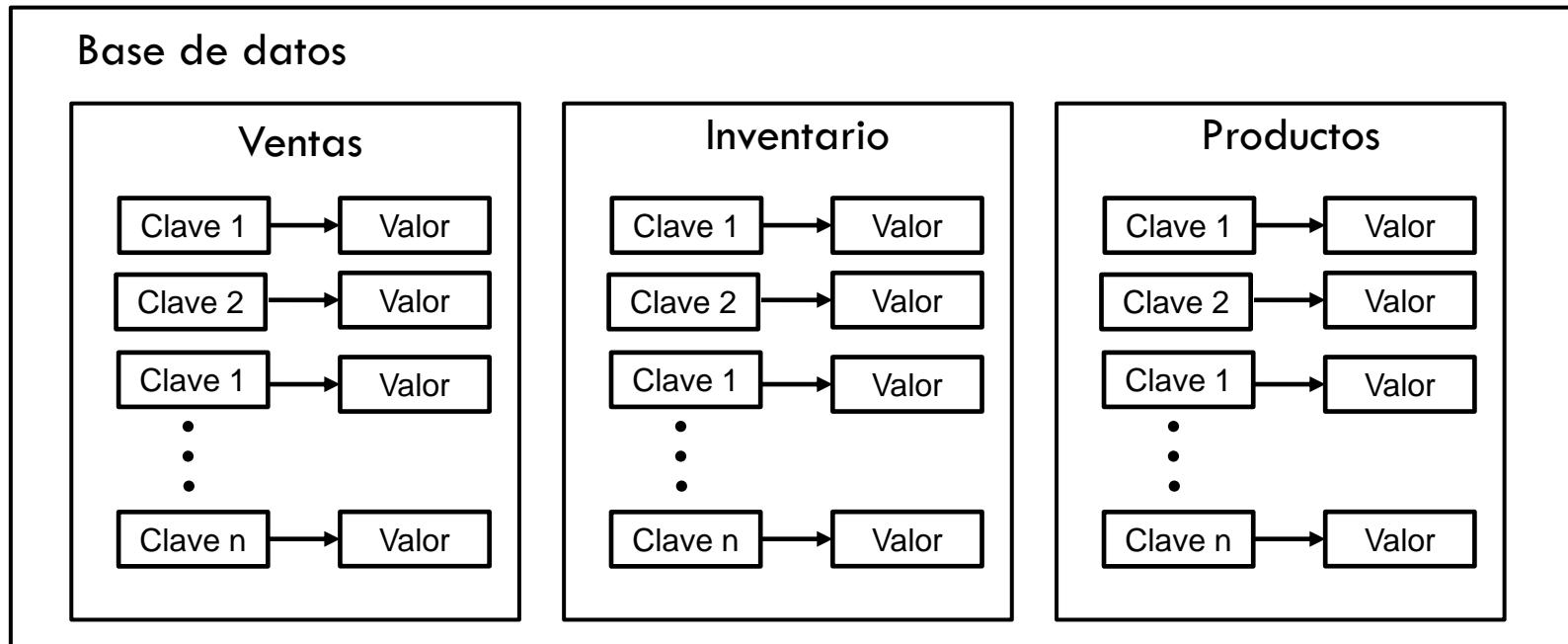
wrhs2.number

wrhs2.address

Namespace o espacio de nombres

- Colección de identificadores
- Las claves deben ser únicas dentro de un namespace

Pueden existir distintos namespaces dentro de una misma BD => **bucket**

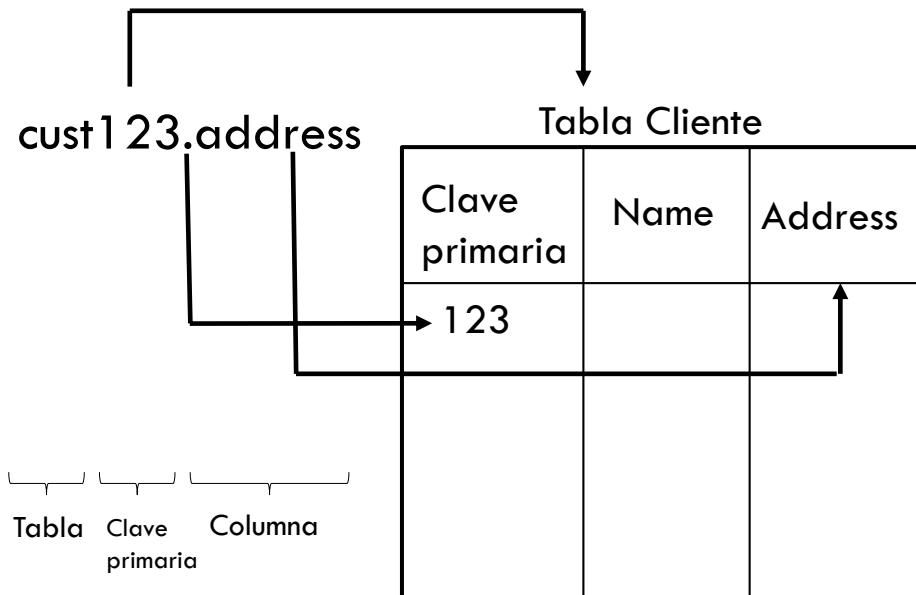


Valores

- Tan sencillos como un string (nombre del cliente, número de productos en carrito, ...)
- O valores más complejos como imágenes
- Gran flexibilidad
 - longitud de strings
 - Tipo: foto o “No disponible”
- Importante incluir checks en los programas

Diferencias con BD relacionales

- No hay tablas => no columnas ni restricciones
 - No hay joins ni claves externas
 - No SQL
 - Buckets ≈ esquema relacional
- Parecido: nombres de claves



Orientadas a documentos

- Documentos en formato JSON o XML

```
{  
    nombre: "Alice",  
    apellido: "Johnson",  
    puesto: "Directora",  
    despacho: "B-314",  
    teléfono: "91-345-98-23",  
}
```

Flexibilidad

- No es necesario crear esquema previo => al añadir documento se crean estructuras necesarias

```
{  
    nombre: "Bob",  
    apellido: "Wilson",  
    puesto: "Coordinador",  
    despacho: "A-409",  
    teléfono: "91-798-36-11,  
}
```

```
{  
    nombre: "Alice",  
    apellido: "Johnson",  
    puesto: "Directora",  
    despacho: "B-314",  
    teléfono: "91-345-98-23,  
}
```

Consultas

- ¿Por qué no utilizar clave-valor?
 - Solo se podría acceder a documentos por su clave
- BD orientadas a documentos => lenguajes de consulta

```
bd.empleado.find( { puesto:"Manager" } )
```

- AND, OR y operaciones de comparación

Diferencias con BD relacionales

- No requieren esquema predefinido
- Documentos embebidos => evita JOINS

{

 nombre: "Bob",
 apellido: "Wilson",
 puesto: "Coordinador",
 despacho: "A-409",
 telefono: "91-798-36-11",
 fechaContratacion: "1-Mar-2010",
 fechaFin: "31-Ago-2016",
 PuestosAnteriores: [

{

 puesto: "Analista",
 fechalni: "1-Mar-2010",
 fechaFin: "28-Feb-2011"

} {

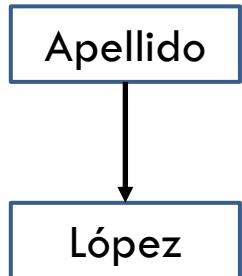
 puesto: "Analista Senior",
 fechalni: "1-Mar-2011",
 fechaFin: "30-Jun-2013"

}]

}

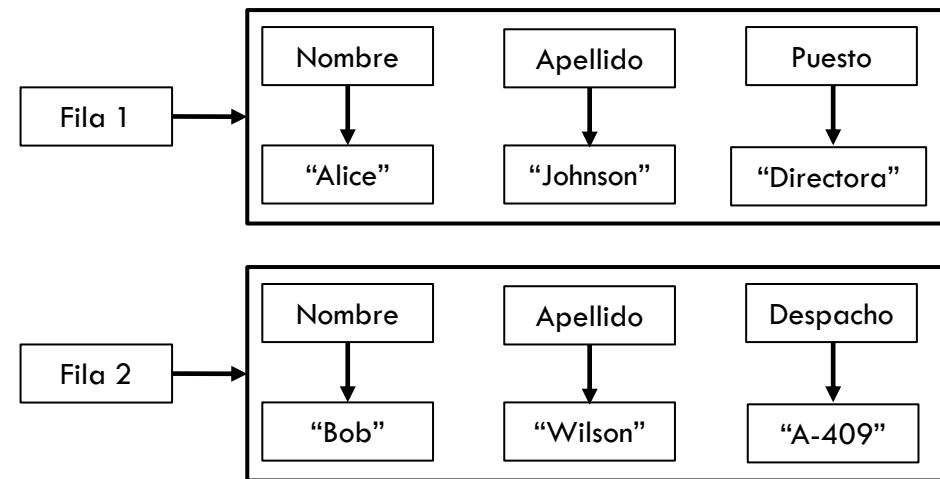
Orientadas a columnas

- Columna = nombre + valor



- Muchas columnas => se pueden agrupar en colecciones o familias de columnas
- Ej.: nombre y apellidos

- Fila = conjunto de columnas
- Distintas filas pueden tener distintas columnas



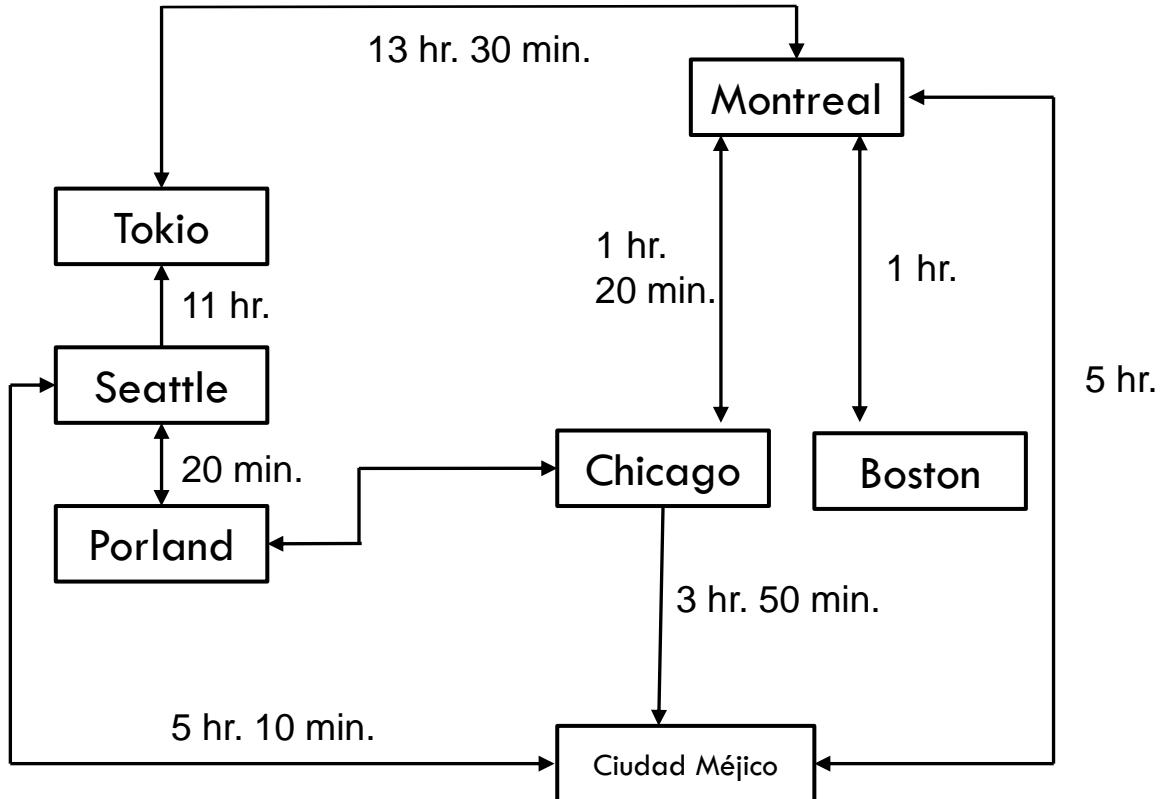
Características

- No requieren esquema predefinido
- Se pueden añadir columnas cuando se necesitan
- Las filas pueden tener distintas columnas
- Diseñadas para filas con muchas columnas (millones)

Diferencias con BD relacionales

- No hay tablas
- El número de columnas varía de una fila a otra
- En BD relacionales: datos sobre un objeto en múltiples tablas => Joins
- En BD orientadas a columnas: en una única fila (muy larga!)
- Lenguaje de consulta similar a SQL: SELECT, INSERT, UPDATE, DELETE, CREATE COLUMNFAMILY.

Orientadas a grafos



Chicago

Aeropuertos: [
 {Nombre: “=‘Hare”
 Símbolo: ORD},
 {Nombre: “Midway”,
 Símbolo: MDW}
]
Población: 2.175.000,
Área: 234 m cuadr.

Diferencias con BD relacionales

- Modelan adyacencia: cada nodo tiene punteros a objetos adyacentes
- Permite acelerar operaciones de búsqueda de caminos entre nodos
- Ejemplo: todas las formas posibles de volar de Montreal a Ciudad de Méjico
- En BD relacional mucho mas ineficiente => Join recursivo

Ciudad 1	Ciudad 2	Tiempo de vuelo
Montreal	Boston	1 hr
Montreal	Chicago	1 hr 20 min.
Montreal	Tokio	13 hr 30 min.
Montreal	Ciudad de Méjico	5 hr
Chicago	Ciudad de Méjico	3 hr. 50 min.
Chicago	Portland	3 hr. 45 min.
Portland	Seattle	20 min.
Seattle	Tokio	11 hr.
Seattle	Ciudad de Méjico	5 hr. 10 min.

BD comerciales

DOCUMENTO



mongoDB



GRAFO



HYPERGRAPHDB

CLAVE-VALOR



COLUMNA



MongoDB

1. Introducción
2. Organización de los datos
3. Modelado de los datos
4. Empezando con MongoDB
5. Algunos ejemplos
6. MongoDB Atlas

Introducción

➤ Introducción (1/6)

- Motor de BBDD documental.
- Almacena objetos BSON (<http://bsonspec.org>)
 - JSON almacenado en formato binario con algunas extensiones (por lo que ocupa menos espacio en memoria que guardar el JSON tal cual – modo texto).
- Gratuita (Existe versión Enterprise, pero no es común).
 - Código abierto.
- Multiplataforma:
 - Disponible para Unix, Linux, Windows y Mac.

➤ Introducción (2/6)

➤ Objetivos

➤ Alto rendimiento:

- Utiliza índices para acelerar consultas.
- Mapa de datos en memoria a nivel de SO.

➤ Alta disponibilidad:

- Replicsets.

➤ Escalable:

- Sharding⁽¹⁾ (particionado)

(1) <http://charlascylon.com/2014-01-30-tutorial-mongodb-explicando-el-sharding-con-una>

➤ Introducción (3/6)

- BBDD NoSQL más popular:

- **Usado en** Facebook, Google, Nokia, SAP, Bosch, Thermo Fisher, Forbes, The Weather Channel, Trend Micro, Sega, Cisco, Telefónica ...

<https://www.mongodb.com/use-cases>

<https://www.mongodb.com/who-uses-mongodb>

- Drivers para la mayoría de lenguajes:

- **Oficiales** C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala.
 - **Comunidad** ActionScript3, Clojure, ColdFusion, D, Dart, Delphi, Entity, Erlang, Factor, Fantom, F#, Go, Groovy, JavaScript, Lisp, Lua, MatLab, Objective C, OCaml, Opa, PowerShell, Prolog, R, REST, Racket, Smalltalk.

➤ Introducción (4/6)

➤ Orígenes

- **Humongous:** extremadamente largo, enorme.
- Evolución:
 - **2007** – almacenamiento de datos para proyecto de plataforma como servicio (PaaS). Ejemplos: Google App Engine, MS Azure.
 - **2009** – producto independiente.
 - **2010** – lanzamiento de versión 1.4, primera considerada suficientemente estable para usar en producción.
 - Versión actual 4.2.7 (mayo 2020).
 - Nosotros usaremos la versión 4.2.7 (mayo 2020).

➤ Introducción (5/6)

➤ Ventajas

- Esquema flexible.
- Lenguaje de consulta y manejo de datos sencillo.
- Fácil integración con las aplicaciones gracias a que utiliza la estructura BSON.
- Accesibilidad a los datos:
 - Posibilidad de realizar lecturas en instancias secundarias, repartiendo la carga de trabajo.

➤ Introducción (6/6)

➤ Desventajas

- Aplicaciones más complejas de construir al trabajar con esquemas flexibles, desnormalizados y dinámicos.
- No garantiza ACID.
 - Consistencia eventual: podrían leerse datos de nodos secundarios que aún no estén actualizados.
 - Sin soporte transaccional.

Organización de los datos

➤ Organización de los datos (1/4)

- Similar a motores relacionales.
 - Estructura dinámica.
 - Bases de datos y colecciones se crean al usarlas.
 - Bases de datos.
 - Agrupan colecciones.
 - Colecciones.
 - Conjunto de documentos en formato BSON.
 - Equivalentes a las tablas del modelo relacional pero:
 - Sin esquema → Cada documento de una colección puede tener diferentes campos.
 - Cada documento dentro de la colección siempre tiene un identificador único.

➤ Organización de los datos (2/4)

- Documentos.
 - Conjunto de pares clave-valor.
 - Equivalentes a las filas del modelo relacional pero:
 - No siguen un esquema de fila pre-establecido: pueden tener tantos campos como se desee.
 - Se almacena el nombre del campo junto a su valor.
 - No hay valores nulos: el campo está o no está (aunque existe un tipo de dato denominado **NULL**).
 - Puede contener otros documentos.
- Campos del documento.
 - Se los asigna valor y son sobre los que trabajan los índices.
 - Equivalentes a las columnas en el modelo relacional.

➤ Organización de los datos (3/4)

➤ Documentos BSON

➤ Estructura libre.

➤ Esquema flexible.

➤ Array asociativo (hash, diccionario).

➤ La clave es una cadena que indica el nombre del campo.

➤ El valor puede ser:

➤ Cadena, número, fecha, datos binarios, ID.

➤ Lista.

➤ Array asociativo.

Base de Datos Documental MongoDB



➤ Organización de los datos (4/4)

➤ Ejemplo

```
{  
    "_id": ObjectId("53f8659a957cf6a0a8b4595"),  
    "name": "The North Face McMurdo II Boot - Women's",  
    "slug": "the-north-face-mc-murdo-ii-boot-womens",  
    "primary_category_id": 8,  
    "price": 135.95,  
    "weight": 1,  
    "variants": {  
        "1": {  
            "name": "Brown, 7"  
        },  
        "2": {  
            "name": "Black, 7"  
        }  
    },  
    "is_active": true,  
    "category_ids": [ 8, 15 ],  
    "date_created": ISODate("2014-08-23T09:57:46Z")  
}
```

Annotations on the JSON document:

- ID** (*Debe ser único y siempre es el primer elemento del documento*)
- cadena**
- Número**
- Array asociativo**
- Array (Comienza en la posición 0)**
- Fecha**

Modelado de datos

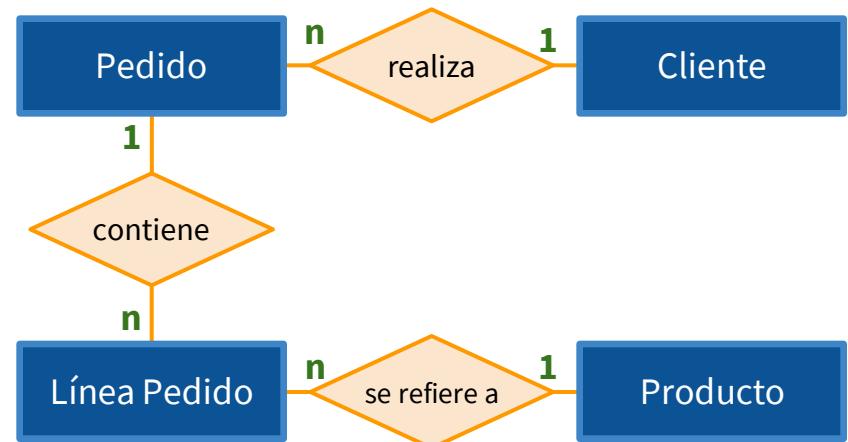
Base de Datos Documental MongoDB



➤ Modelado de datos: relacional (1/4)

➤ Evolución.

➤ Modelo entidad-relación.



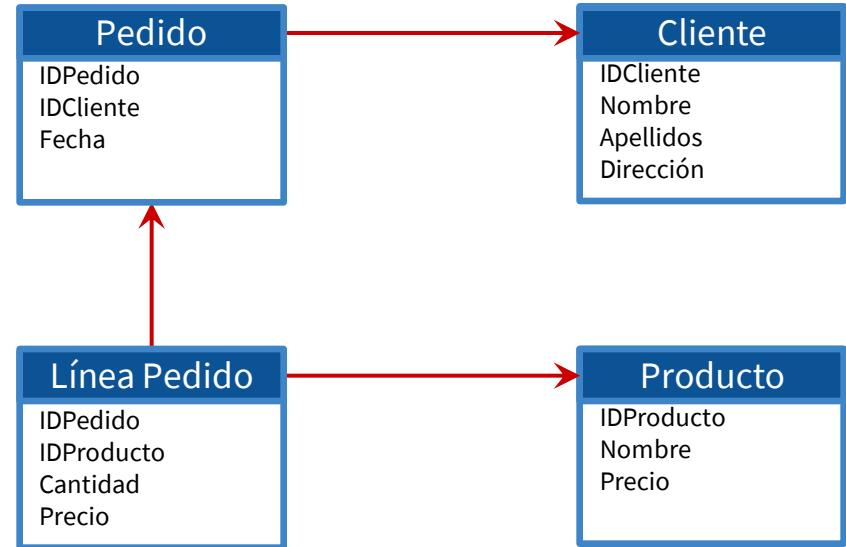
Base de Datos Documental MongoDB



➤ Modelado de datos: relacional (2/4)

➤ Evolución.

- Modelo entidad-relación.
- Modelo relacional.



Base de Datos Documental MongoDB



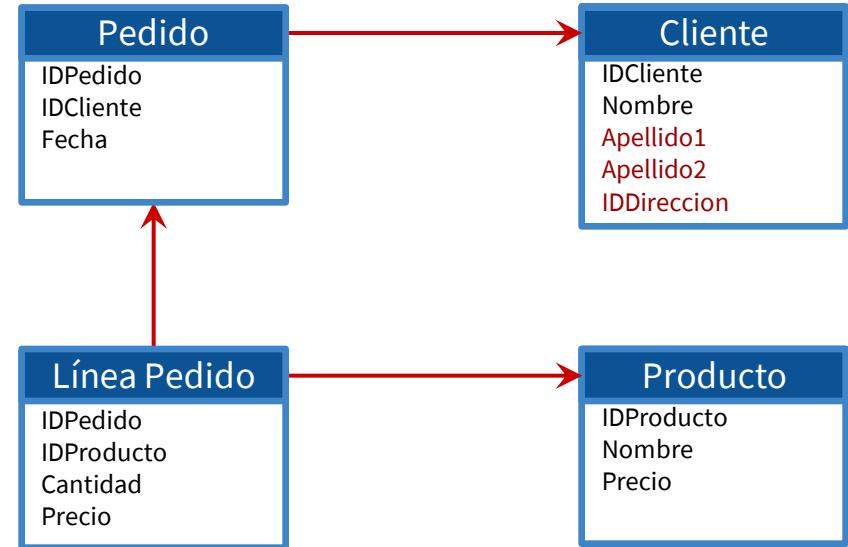
➤ Modelado de datos: relacional (3/4)

➤ Evolución.

➤ Modelo entidad-relación.

➤ Modelo relacional.

➤ Base de datos normalizada.

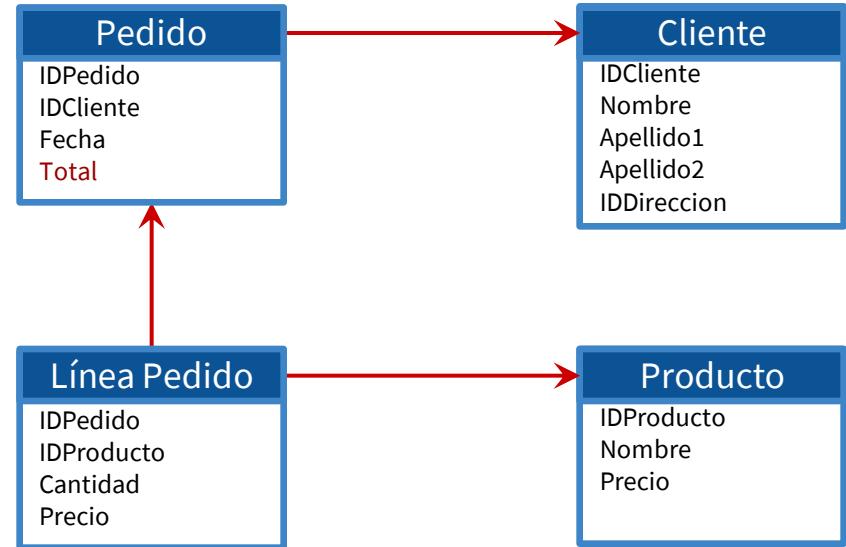


Base de Datos Documental MongoDB



➤ Modelado de datos: relacional (4/4)

- Evolución.
 - Modelo entidad-relación.
 - Modelo relacional.
 - Base de datos normalizada.
 - Denormalización



Base de Datos Documental MongoDB



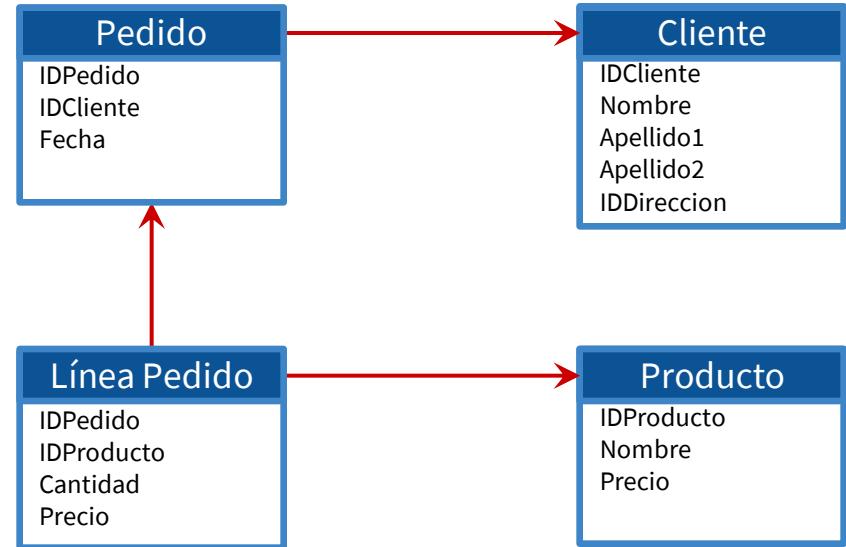
➤ Modelado de datos: MongoDB (1/2)

➤ Evolución.

➤ Modelo entidad-relación.

➤ Modelo relacional.

➤ Base de datos normalizada.

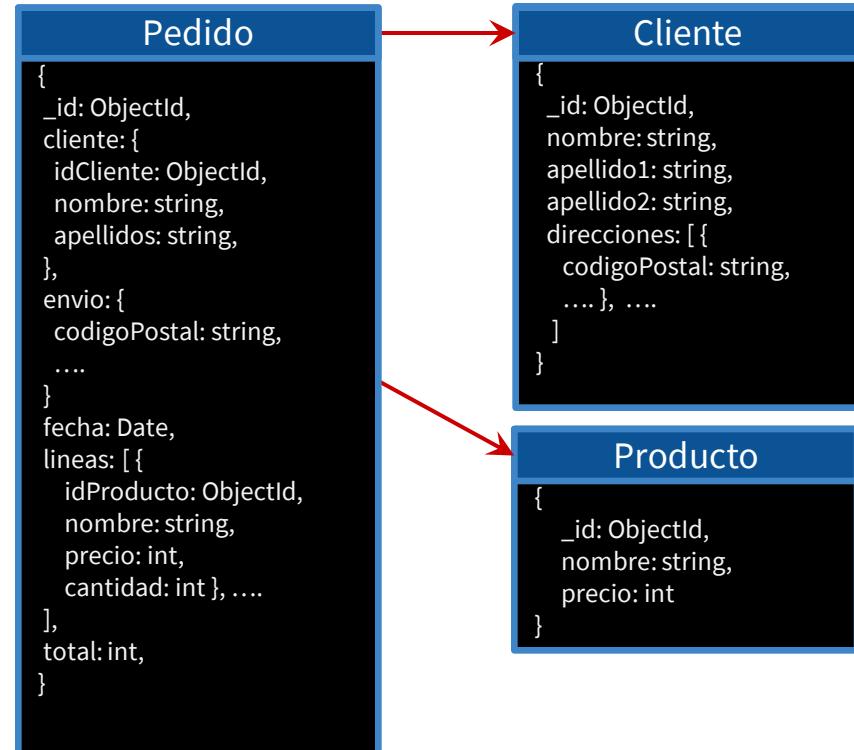


Base de Datos Documental MongoDB



➤ Modelado de datos: MongoDB (2/2)

- Evolución.
 - Modelo entidad-relación.
 - Modelo relacional.
 - Base de datos normalizada.
 - Denormalización.
 - Depende del uso.
 - Entidades anidadas.
 - Referencias poco funcionales.



➤ Modelado de datos (1/10)

➤ Claves para modelar bases de datos relacionales (1/2)

- Denormalización principalmente por rendimiento.
- Diseño "definitivo".
 - Cambios en el modo de uso no suelen alterar el modelo.
- Esquema fijo.
 - Garantía de que el esquema se cumple.
 - Facilita el uso y modificación de los datos.
 - No facilita la introducción de cambios.



- Modelado de datos (2/10)
 - Claves para modelar bases de datos relacionales (2/2)
 - Muy **independiente** del modo de uso.
 - Casi cualquier operación de lectura (listados, detalles, reportes, ...) es posible mediante **JOINs**.
 - Soporte para transacciones permite modificar varios registros **atómicamente**.



- Modelado de datos (3/10)

- Claves para modelar bases de datos MongoDB (1/8)

- Denormalización según uso.

- Diseño iterativo:

- Cambios en el modo de uso, alteran el modelo.

- Esquema flexible:

- Facilita la adaptación del modelo a nuevos escenarios.

- No implica ausencia de esquema.

- La base de datos no garantiza que se cumpla el esquema.

- Requiere mayor control en el uso y modificación de datos.

- Un cambio mal hecho podría “romper” una aplicación.



- Modelado de datos (4/10)

- Claves para modelar bases de datos MongoDB (2/8)

- Muy **dependiente** del modo de uso.

- Cambios en el modo de uso, alteran el modelo.
- Según las lecturas a realizar, las relaciones se resuelven por **anidamiento de entidades**, referencias o ambas.
- Generación de reportes mediante **herramientas de agregación** nativas o externas.
- Las modificaciones también influyen, es deseable garantizar la **atomicidad** de las operaciones, pero también reducir el **crecimiento de los documentos**.



- Modelado de datos (5/10)

- Claves para modelar bases de datos MongoDB (3/8)

- Tipos de **relaciones**:

- *Uno-a-Uno.* En este tipo de relación, uno de los documentos es embebido dentro de otro.
- *Uno-a-Varios.* Los documentos de la relación son embebidos dentro de otro en una estructura de tipo array.
- *Uno-a-Varios con referencias.* Se utilizan referencias a `_id` de los documentos relacionados, en lugar de que sean dichos documentos embebidos.



- Modelado de datos (6/10)

- Claves para modelar bases de datos MongoDB (4/8)

- Relación **Uno-a-Uno**:

- Ejemplo: almacenar los datos de los clientes de un gimnasio. La solución sería incluir la dirección como un subdocumento del documento principal:

```
{  
    "_id": ObjectId("53f8659a957cf6a0a8b4595") ,  
    "name": "Juan",  
    "surname": "Sanchez",  
    ...  
    "address": {  
        "street": "Brown, 7"  
    }  
}
```



- Modelado de datos (7/10)

- Claves para modelar bases de datos MongoDB (5/8)

- Relación **Uno-a-Varios**:

- Ejemplo: almacenar los datos de los clientes de un gimnasio, los cuales pueden tener varias direcciones.

```
{  
    "_id": ObjectId("53f8659a957cf6a0a8b4595") ,  
    "name": "Juan",  
    "surname": "Sanchez",  
    ...  
    "addresses": [  
        { "street": "Brown, 7" } ,  
        { "street": "Red, 14" }  
    ]  
}
```



- Modelado de datos (8/10)

- Claves para modelar bases de datos MongoDB (6/8)

- Relación **Uno-a-Varios con referencias** (1/3):

- A veces puede ser muy pesado el guardar los subdocumentos en arrays.
- Veamos ahora el siguiente ejemplo:

```
{  
    "_id": "UAMEdiciones",  
    "city": "Madrid"  
    ...  
    "books": [  
        { "name": "Introducción a MongoDB",  
          "ISBN": "1-4493-1281-0"  
          ...  
        },  
        { "name": "Introducción a BBDD NoSQL",  
          "ISBN": "3386490158"  
          ...  
        }  
    ]  
}
```



➤ Modelado de datos (9/10)

➤ Claves para modelar bases de datos MongoDB (7/8)

➤ Relación **Uno-a-Varios con referencias** (2/3):

- PROBLEMA: el array puede crecer mucho → Búsquedas ineficientes.
- Otra posible solución:

```
{  
    "_id": "InMongoDB",  
    "name": "Introducción a MongoDB",  
    "ISBN": "1-4493-1281-0"  
    ...  
    "editor": {  
        "name": "UAMEdiciones",  
        "city": "Madrid"  
        ...  
    }  
}
```



➤ Modelado de datos (10/10)

➤ Claves para modelar bases de datos MongoDB (8/8)

➤ Relación Uno-a-Varios con referencias (3/3):

- PROBLEMA: repetición frecuente de los datos **editor** → Ocupando espacio.
- Solución:

```
{  
    "_id": "UAMEDiciones",  
    "city": "Madrid",  
    ...  
}
```

```
{  
    "_id": "InMongoDB",  
    "name": "Introducción a MongoDB",  
    "ISBN": "1-4493-1281-0",  
    ...  
    "editor": "UAMEDiciones"  
}
```



Empezando con MongoDB

- Manejo básico de datos (1/6)
 - Comandos básicos de la consola (1/3)
 - Usar una base de datos (aunque no exista).

```
> use <db>
```

- Listar objetos.

```
> show dbs
```

```
> show collections
```

- Ayuda.

```
> help
```

```
> db.help()
```

```
> db.<colección>.help()
```

Base de Datos Documental MongoDB



- Manejo básico de datos (2/6)
 - Comandos básicos de la consola (2/3)

```
> use bbdduam          > use fwd           > help
switched to db bbdduam switched to db fwd      db.help()
> █                         > show collections      db.mycoll.help()
                                         accounts        sh.help()
                                         auto_increments   rs.help()
                                         carts           help admin
                                         categories       help connect
                                         channels        help keys
                                         discounts        help misc
                                         entries         help mr
                                         orders          show dbs
                                         products        show collections
                                         settings        show users
                                         trash           show profile
                                         > █                         show logs
                                         > █                         show log [name]
                                         > █                         use <db_name>
                                         > █                         db.foo.find()
                                         > █                         db.foo.find( { a : 1 } )
                                         > █                         it
                                         > █                         DBQuery.shellBatchSize = x
                                         > █                         exit
                                         > █                         help on db methods
                                         > █                         help on collection methods
                                         > █                         sharding helpers
                                         > █                         replica set helpers
                                         > █                         administrative help
                                         > █                         connecting to a db help
                                         > █                         key shortcuts
                                         > █                         misc things to know
                                         > █                         mapreduce
                                         > █                         show database names
                                         > █                         show collections in current database
                                         > █                         show users in current database
                                         > █                         show most recent system.profile entries with time >= 1ms
                                         > █                         show the accessible logger names
                                         > █                         prints out the last segment of log in memory, 'global' is default
                                         > █                         set current database
                                         > █                         list objects in collection foo
                                         > █                         list objects in foo where a == 1
                                         > █                         result of the last line evaluated; use to further iterate
                                         > █                         set default number of items to display on shell
                                         > █                         quit the mongo shell
```

- Manejo básico de datos (3/6)
 - Comandos básicos de la consola (3/3)

```
> db.help()
DB methods:
  db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
  db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a cursor
  db.auth(username, password)
  db.cloneDatabase(fromhost) - deprecated
  db.commandHelp(name) returns the help for the command
  db.copyDatabase(fromdb, todb, fromhost) - deprecated
  db.createCollection(name, {size: ..., capped: ..., max: ...})
  db.createView(name, viewOn, [{operator: {...}}], ..., {viewOptions})
  db.createUser(userDocument)
  db.currentOp() displays currently executing operations in the db
  db.dropDatabase()
  db.eval() - deprecated
  db.fsyncLock() flush data to disk and lock server for backups
  db.fsyncUnlock() unlocks server following a db.fsyncLock()
  db.getCollection(cname) same as db['cname'] or db.cname
  db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
  db.getCollectionNames()
  db.getLastErrorMessage() - just returns the err msg string
```

```
> db.categories.help()
DBCollection help
  db.categories.find().help() - show DBCursor help
  db.categories.bulkWrite(operations, <optional params>) - bulk execute write operations, optional parameters are: w, wtimeout, j
  db.categories.count(query = {}, <optional params>) - count the number of documents that matches the query, optional parameters are: limit, skip, hint, maxTimeMS
  db.categories.copyTo(newColl) - duplicates collection by copying all documents to newColl; no indexes are copied.
  db.categories.convertToCapped(maxBytes) - calls {convertToCapped:'categories', size:maxBytes} command
  db.categories.createIndex(keypattern[,options])
  db.categories.createIndexes([keypatterns], <options>)
  db.categories.dataSize()
  db.categories.deleteOne(filter, <optional params>) - delete first matching document, optional parameters are: w, wtimeout, j
  db.categories.deleteMany(filter, <optional params>) - delete all matching documents, optional parameters are: w, wtimeout, j
  db.categories.distinct(key, query, <optional params>) - e.g. db.categories.distinct('x'), optional parameters are: maxTimeMS
  db.categories.drop() drop the collection
```

➤ Manejo básico de datos (4/6)

- Algunos términos pueden resultar ambiguos, por lo que vamos a definir nombres diferentes para cada uso (1/3)

➤ Objeto.

- Array asociativo que puede ser representado mediante JSON.
 - Pares clave-valor.
- Puede contener listas y arrays asociativos como valores.
- El orden de las claves es relevante:

```
db.usuarios.insert { "nombre": "jj", "edad": 43 }
```



```
db.usuarios.insert { "edad": 43, "nombre": "jj" }
```

➤ Manejo básico de datos (5/6)

- Algunos términos pueden resultar ambiguos, por lo que vamos a definir nombres diferentes para cada uso (2/3)

- Documento.

- Objeto que representa un registro de una colección.
 - Aunque todavía no lo sea o lo haya sido.
 - Tiene el campo `_id`.

```
{ "_id": ObjectId("53f8659a957cf6a0a8b4595"), "name": "The North Face McMurdo II Boot - Women's",  
  "slug": "the-north-face-mc-murdo-ii-boot-womens", "primary_category_id": 8, "price": 135.95,  
  "weight": 1, "variants": { "1": { "name": "Brown, 7"}, "2": { "name": "Black, 7" } }, "is_active": true,  
  "category_ids": [8, 15], "date_created": ISODate("2014-08-23T09:57:46Z") }
```

➤ Manejo básico de datos (6/6)

- Algunos términos pueden resultar ambiguos, por lo que vamos a definir nombres diferentes para cada uso (3/3)

➤ Subdocumento.

- Objeto incluido en el valor de una clave de un documento.
- Puede haber varios niveles de anidamiento.
- Todos son subdocumentos.

```
{ "_id": ObjectId("53f8659a957cf6a0a8b4595"), "name": "The North Face McMurdo II Boot - Women's",  
  "slug": "the-north-face-mc-murdo-ii-boot-womens", "primary_category_id": 8, "price": 135.95,  
  "weight": 1, "variants": { "1": { "name": "Brown, 7"}, "2": { "name": "Black, 7"} },  
  "is_active": true, "category_ids": [8, 15], "date_created": ISODate("2014-08-23T09:57:46Z") }
```

- Manejo básico de datos: métodos de lectura (1/15)
 - Sólo se accede a datos de una colección a la vez.
 - A partir de la versión 3.2 (Diciembre 2015) sí existe una especie de JOIN sobre una colección no “shardeada” y que pertenezca a la misma BBBDD.
 - `find` (buscar).
 - Devuelve el cursor para recorrer los resultados.

```
> db.<colección>.find( <filtros>, <proyecciones> )
```
 - `findOne` (buscar un único resultado).
 - Devuelve el primer resultado de la consulta.

```
> db.<colección>.findOne( <filtros>, <proyecciones> )
```



➤ Manejo básico de datos: métodos de lectura (2/15)

```
db.products.find( { id: 4 } )  
=   
> db.products.find( { "id":4 } )  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9, "sku" : "ARC1021", "price" : 559.95, "weight" : 1, "variants" :  
{ "1" : { "name" : "Orange, Small" }, "2" : { "name" : "Brown, Small" }, "3" : { "name" : "Blue, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce acilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 9, 16 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }  
>
```



➤ Manejo básico de datos: métodos de lectura (3/15)

```
> db.products.findOne( { "id":4 } )
{
  "_id" : ObjectId("53f8659a957cf6a0a8b4596"),
  "id" : 4,
  "name" : "Arc'teryx Crossbow Jacket - Men's",
  "slug" : "arcteryx-crossbow-jacket-mens",
  "primary_category_id" : 9,
  "sku" : "ARC1021",
  "price" : 559.95,
  "weight" : 1,
  "variants" : {
    "1" : {
      "name" : "Orange, Small"
    },
    "2" : {
      "name" : "Brown, Small"
    },
    "3" : {
      "name" : "Blue, Small"
    }
  },
  "is_active" : true,
  "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusc e pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
  "category_ids" : [
    9,
    16
  ],
  "date_created" : ISODate("2014-08-23T09:57:46Z")
}
```



- Manejo básico de datos: métodos de lectura (4/15)
 - Filtros (1/3)
 - Permiten seleccionar qué documentos traer de una colección.
 - Por defecto, se traen todos los resultados.
 - Objeto (hash) con criterios de búsqueda:
 - Cada clave indica el nombre de un campo a evaluar.
 - El valor asociado indica el valor que debe tener dicho campo.
 - Un documento debe cumplir TODOS los criterios para formar parte de los resultados.



- Manejo básico de datos: métodos de lectura (5/15)

- Filtros (2/3)

- Ejemplos:

- Obtener todos los productos:

```
> db.products.find( )
```

```
> db.products.find( {} )
```

- Obtener todos los productos con peso 1 y activos:

```
> db.products.find( { "is_active": true, "weight": 1 } )
```



Base de Datos Documental MongoDB



➤ Manejo básico de datos: métodos de lectura (6/15)

➤ Filtros (3/4)

```
> db.products.find()
{ "_id" : ObjectId("53f8659a957cfdea0a8b4593"), "id" : 1, "name" : "The North Face Ava Triclimate Jacket - Women's", "slug" : "the-north-face-ava-triclimate-jacket-womens", "primary_category_id" : 6, "sku" : "TNF7814", "price" : 229.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Black, Small" }, "3" : { "name" : "Pink, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque a convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 6, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
{ "_id" : ObjectId("53f8659a957cfdea0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summit Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" : 7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }
{ "_id" : ObjectId("53f8659a957cfdea0a8b4595"), "id" : 3, "name" : "The North Face McMurdo II Boot - Women's", "slug" : "the-north-face-mc-murdo-ii-boot-womens", "primary_category_id" : 8, "sku" : "TNF6706", "price" : 135.95, "weight" : 1, "variants" : { "1" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 8, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }
{ "_id" : ObjectId("53f8659a957cfdea0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9,
```

```
> db.products.find( {} )
{ "_id" : ObjectId("53f8659a957cfdea0a8b4593"), "id" : 1, "name" : "The North Face Ava Triclimate Jacket - Women's", "slug" : "the-north-face-ava-triclimate-jacket-womens", "primary_category_id" : 6, "sku" : "TNF7814", "price" : 229.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Black, Small" }, "3" : { "name" : "Pink, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque a convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 6, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }
{ "_id" : ObjectId("53f8659a957cfdea0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summit Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" : 7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }
{ "_id" : ObjectId("53f8659a957cfdea0a8b4595"), "id" : 3, "name" : "The North Face McMurdo II Boot - Women's", "slug" : "the-north-face-mc-murdo-ii-boot-womens", "primary_category_id" : 8, "sku" : "TNF6706", "price" : 135.95, "weight" : 1, "variants" : { "1" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 8, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }
{ "_id" : ObjectId("53f8659a957cfdea0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9,
```



- Manejo básico de datos: métodos de lectura (7/15)
 - Filtros (4/4)

```
> db.products.find( { "is_active": true, "weight": 1 } )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4595"), "id" : 1, "name" : "The North Face Ava Triclimate Jacket - W  
" : "TNF7814", "price" : 229.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "nam  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam,  
s sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vit  
tegory_ids" : [ 6, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z" )  
{ "id" : ObjectId("53f8659a957cf0d6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summit Pant - Women's", "sl  
339.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3  
et, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nu  
, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pelle  
"date_created" : ISODate("2014-08-23T09:57:46Z" )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4595"), "id" : 3, "name" : "The North Face McMurdo II Boot - Women's  
6", "price" : 135.95, "weight" : 1, "variants" : { "1" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7  
lit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat  
reit imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesu  
014-08-23T09:57:46Z" )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slu  
9.95, "weight" : 1, "variants" : { "1" : { "name" : "Orange, Small" }, "2" : { "name" : "Brown, Small" }, "3
```

```
< db.products.find( { "weight": 1, "is_active": true } )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4595"), "id" : 1, "name" : "The North Face Av  
" : "TNF7814", "price" : 229.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue,  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fus  
s sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imper  
tegory_ids" : [ 6, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z" )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summ  
339.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3  
et, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc.  
, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vita  
"date_created" : ISODate("2014-08-23T09:57:46Z" )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4595"), "id" : 3, "name" : "The North Face Mc  
6", "price" : 135.95, "weight" : 1, "variants" : { "1" : { "name" : "Brown, 7" }, "2"  
lit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat  
reit imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesu  
014-08-23T09:57:46Z" )  
{ "_id" : ObjectId("53f8659a957cf0d6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbo  
9.95, "weight" : 1, "variants" : { "1" : { "name" : "Orange, Small" }, "2" : { "name" : "Brown, Small" }, "3  
t, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, sc
```



- Manejo básico de datos: métodos de lectura (8/15)

- Proyecciones (1/4)

- Permiten definir qué campos se incluirán en los resultados.
 - Por defecto, se traen todos los campos.
- Objeto con campos a incluir o excluir del resultado:
 - La clave indica el nombre del campo.
 - Valores a `0`: se trae todo menos dichos campos.
 - Valores a `1`: sólo se traen esos campos.
 - El `_id` siempre viene, salvo que se excluya explícitamente.
 - No se pueden mezclar valores a `0` y a `1`, excepto el `_id`.



- Manejo básico de datos: métodos de lectura (9/15)

- Proyecciones (2/4)

- Ejemplos:

- Obtener todos los campos.

```
> db.products.find( )
```

```
> db.products.find( {} , {} )
```

- Excluir el campo descripción.

```
> db.products.find( {}, { "description": 0 } )
```

- Obtener sólo los precios.

```
> db.products.find( {}, { "price": 1, "_id": 0 } )
```



- Manejo básico de datos: métodos de lectura (10/15)
 - Proyecciones (3/4)

```
> db.products.find()  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4593"), "id" : 1, "name" : "The North Face Ava Triclimate Jacket - Women's", "slug" : "the-north-face-ava-triclimate-jacket-womens", "primary_category_id" : 6, "sku" : "TNF7814", "price" : 229.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Black, Small" }, "3" : { "name" : "Pink, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque a t convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 6, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summit Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" : 7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecena s dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices tu rpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse plat ea dictumst.", "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4595"), "id" : 3, "name" : "The North Face Mc Murdoch II Boot - Women's", "slug" : "the-north-face-mc-murdo-li-boot-womens", "primary_c ategory_id" : 8, "sku" : "TNF6706", "prlce" : 135.95, "weight" : 1, "variants" : { "1" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7" } }, "is_active" : true, "desc ription" : "Lorem ipsum dolor sit amet, consectetur adipiscng elit. Quisque sed mauri s enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iacul is ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce p ellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 8, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9,
```

```
> db.products.find( { }, { } )  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4593"), "id" : 1, "name" : "The North Face Ava Triclimate Jacket - Women's", "slug" : "the-north-face-ava-triclimate-jacket-womens", "primary_category_id" : 6, "sku" : "TNF7814", "price" : 229.95, "weight" : 1, "varian ts" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Black, Small" }, "3" : { "name" : "Pink, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque a t convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuad a. In hac habitasse platea dictumst.", "category_ids" : [ 6, 15 ], "date_created" : ISO Date("2014-08-23T09:57:46Z") }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summi Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" : 7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipis cing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nu nc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecena s dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices tu rpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse plat ea dictumst.", "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4595"), "id" : 3, "name" : "The North Face Mc Murdoch II Boot - Women's", "slug" : "the-north-face-mc-murdo-li-boot-womens", "primary_c ategory_id" : 8, "sku" : "TNF6706", "price" : 135.95, "weight" : 1, "variants" : { "1" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7" } }, "is_active" : true, "desc ription" : "Lorem ipsum dolor sit amet, consectetur adipiscng elit. Quisque sed mauri s enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iacul is ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce p ellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 8, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9,
```



- Manejo básico de datos: métodos de lectura (11/15)
 - Proyecciones (4/4)

```
> db.products.find( {}, { "description": 0 } )
{ "_id" : ObjectId("53f8659a957cf6a0a8b4593"), "id" : 1, "name" : "The North Face Ava Triclimate Jacket - Women's", "slug" : "the-north-face-ava-triclimate-jacket-womens", "primary_category_id" : 6, "sku" : "TNF7814", "price" : 229.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Black, Small" }, "3" : { "name" : "Pink, Small" } }, "is_active" : true, "category_ids" : [ 6, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
{ "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summit Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" : 7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }, "is_active" : true, "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
{ "_id" : ObjectId("53f8659a957cf6a0a8b4595"), "id" : 3, "name" : "The North Face McMurdo II Boot - Women's", "slug" : "the-north-face-mc-murdo-ii-boot-womens", "primary_category_id" : 8, "sku" : "TNF6706", "price" : 135.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7" } } ], "is_active" : true, "category_ids" : [ 8, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
{ "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9, "sku" : "ARC1021", "price" : 559.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Canyon, Small" }, "2" : { "name" : "Brown, Small" }, "3" : { "name" : "Blue, Small" } }, "is_active" : true, "category_ids" : [ 9, 16 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
{ "_id" : ObjectId("53f8659a957cf6a0a8b4597"), "id" : 5, "name" : "Columbia Bugaboot Plus XTM Boot - Men's", "slug" : "columbia-bugaboot-plus-xtm-boot-mens", "primary_category_id" : 11, "sku" : "COL3515", "price" : 137.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Brown, 9" }, "2" : { "name" : "Black, 9" } } ], "is_active" : true, "category_ids" : [ 11, 17 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
{ "_id" : ObjectId("53f8659a957cf6a0a8b4598"), "id" : 6, "name" : "Burton AK Freebird Bib Pant - Men's", "slug" : "burton-ak-freebird-bib-pant-mens", "primary_category_id" : 10, "sku" : "BUR4929", "price" : 489.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Beige, Small" }, "2" : { "name" : "Black, Small" } } ], "is_active" : true, "category_ids" : [ 10, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
```

```
> db.products.find( {}, { "price": 1, "_id": 0 } )
{ "price" : 229.95 }
{ "price" : 339.95 }
{ "price" : 135.95 }
{ "price" : 559.95 }
{ "price" : 137.95 }
{ "price" : 489.95 }
{ "price" : 235.95 }
{ "price" : 42.95 }
{ "price" : 39.95 }
{ "price" : 44.95 }
{ "price" : 129.95 }
> ■
```

```
> db.products.find( {}, { "price": 1, "description": 0 } )
-
```

¿SALIDA?



- Manejo básico de datos: métodos de lectura (12/15)

- Ordenar resultados (1/2)

- Métodos asociados al cursor.
- Por defecto, los datos se obtienen en el orden natural.
 - Orden en que están almacenados en disco (no en cómo se introdujeron).
- Objeto con campos a utilizar.
 - Es relevante el orden de los campos.
 - Valor `1` para orden ascendente, `-1` para descendente.

```
> db.products.find().sort( { "price": -1, "name": 1 } )
```



- Manejo básico de datos: métodos de lectura (13/15)
 - Ordenar resultados (2/2)

```
> db.products.find().sort( { "price": -1, "name": 1 } )  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow  
Jacket - Men's", "slug" : "arc-teryx-crossbow-jacket-mens", "primary_category_id" : 9,  
"sku" : "ARCI021", "price" : 559.95, "weight" : 1, "variants" : { "1" : { "name" : "C  
range, Small" }, "2" : { "name" : "Crown, Small" }, "3" : { "name" : "Blue, Small" } }  
, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipisc  
ing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nur  
c. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas  
dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices tur  
pis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse plate  
a dictumst.", "category_ids" : [ 9, 16 ], "date_created" : ISODate("2014-08-23T09:57:4  
6Z" ) }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4598"), "id" : 6, "name" : "Burton AK Freebird  
Bib Pant - Men's", "slug" : "burton-ak-freebird-bib-pant-mens", "primary_category_id"  
: 10, "sku" : "BUR4929", "price" : 489.95, "weight" : 1, "variants" : { "1" : { "name"  
" : "Beige, Small" }, "2" : { "name" : "Black, Small" } }, "is_active" : true, "descri  
ption" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris  
enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat d  
olor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis  
ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pel  
lentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids"  
: [ 10, 14 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summi  
t Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" :  
7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" :  
"Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }  
, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipis  
cing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nu  
nc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecena  
s dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices tu  
rpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse plat  
ea dictumst.", "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:  
46Z" ) }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4599"), "id" : 7, "name" : "The North Face Met  
ropolis Down Parka - Girls'", "slug" : "the-north-face-metropolis-down-parka-girls'", "  
primary_category_id" : 12, "sku" : "TNF6632", "price" : 235.95, "weight" : 1, "variant
```



- Manejo básico de datos: métodos de lectura (14/15)

- Subconjuntos de resultados (1/2)

- Métodos asociados al cursor.
- Limit (limitar).
 - Obtener como máximo X resultados de una consulta.
 - Mejora en rendimiento, evita recorrer resultados extra.
- Skip (saltar).
 - Ignorar los primeros X resultados de una consulta.
 - (!) No envía los resultados ignorados.

```
> db.products.find().skip( 10 ).limit( 10 )
```



- Manejo básico de datos: métodos de lectura (15/15)
 - Subconjuntos de resultados (2/2)

```
> db.products.find().skip( 10 ).limit( 10 )
{ "_id" : ObjectId("53f8659a957cf6a0a8b459d"), "id" : 11, "name" : "Electric EG2 Goggle", "slug" : "electric-eg2-goggle", "primary_category_id" : 21, "sku" : "ELC0649", "price" : 129.95, "weight" : 1, "variants" : { "1" : { "name" : "Orange" }, "2" : { "name" : "Grey" }, "3" : { "name" : "Black" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 18, 21 ], "date_created" : ISODate("2014-08-23T09:57:46Z") }
```

>



- Manejo básico de datos: listas y subdocumentos (1/6)
 - Aportan una nueva dimensión a los datos.
 - Permiten almacenar “varios registros” dentro de uno.
 - En bases de datos relacionales necesitaríamos utilizar varias tablas la mayoría de las veces.
 - Complican un poco las operaciones.
 - Siempre se debe recordar que la unidad básica de trabajo son los documentos.



➤ Manejo básico de datos: listas y subdocumentos (2/6)

➤ Notación de puntos (***dot notation***).

➤ Permite acceder a valores que estén dentro de listas o subdocumentos.

➤ Listas.

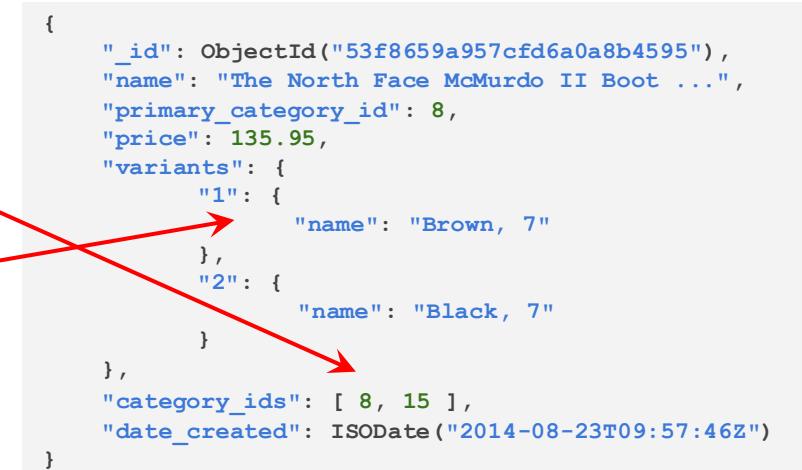
➤ Índice del elemento.

"category_ids.0"

➤ Subdocumentos.

➤ Clave del elemento.

"variants.1.name"



```
{  
  "_id": ObjectId("53f8659a957cf6a0a8b4595"),  
  "name": "The North Face McMurdo II Boot ...",  
  "primary_category_id": 8,  
  "price": 135.95,  
  "variants": {  
    "1": {  
      "name": "Brown, 7"  
    },  
    "2": {  
      "name": "Black, 7"  
    }  
  },  
  "category_ids": [ 8, 15 ],  
  "date_created": ISODate("2014-08-23T09:57:46Z")  
}
```



- Manejo básico de datos: listas y subdocumentos (3/6)

- Búsquedas en listas (1/2)

- Es posible realizar búsquedas exactas o de elementos puntuales.

- Coincidencia exacta.

```
{ "category_ids": [ 8, 15 ] }
```

- Coincidencia puntual.

```
{ "category_ids": 8 }
```

```
{
  "_id": ObjectId("53f8659a957cf6a0a8b4595"),
  "name": "The North Face McMurdo II Boot ...",
  "primary_category_id": 8,
  "price": 135.95,
  "variants": {
    "1": {
      "name": "Brown, 7"
    },
    "2": {
      "name": "Black, 7"
    }
  },
  "category_ids": [ 8, 15 ],
  "date_created": ISODate("2014-08-23T09:57:46Z")
}
```



- Manejo básico de datos: listas y subdocumentos (4/6)
 - Búsquedas en listas (2/2)

```
> db.products.find( {"category_ids.0" : 8 } ).pretty()
{
    "_id" : ObjectId("53f8659a957cf6a0a8b4595"),
    "id" : 3,
    "name" : "The North Face McMurdo II Boot - Women's",
    "slug" : "the-north-face-mc-murdo-ii-boot-womens",
    "primary_category_id" : 8,
    "sku" : "TNF6706",
    "price" : 135.95,
    "weight" : 1,
    "variants" : [
        "1" : {
            "name" : "Brown, 7"
        },
        "2" : {
            "name" : "Black, 7"
        }
    ],
    "is_active" : true,
    "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis que sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pha retria a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
    "category_ids" : [
        8,
        15
    ],
    "date_created" : ISODate("2014-08-23T09:57:46Z")
}
> ■
```

```
> db.products.find( { category_ids: 8 } ).pretty()
{
    "_id" : ObjectId("53f8659a957cf6a0a8b4595"),
    "id" : 3,
    "name" : "The North Face McMurdo II Boot - Women's",
    "slug" : "the-north-face-mc-murdo-ii-boot-womens",
    "primary_category_id" : 8,
    "sku" : "TNF6706",
    "price" : 135.95,
    "weight" : 1,
    "variants" : [
        "1" : {
            "name" : "Brown, 7"
        },
        "2" : {
            "name" : "Black, 7"
        }
    ],
    "is_active" : true,
    "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis que sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pha retria a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
    "category_ids" : [
        8,
        15
    ],
    "date_created" : ISODate("2014-08-23T09:57:46Z")
}
> ■
```

Si tenemos
“category_ids”: 8,
también aparecería.



- Manejo básico de datos: listas y subdocumentos (5/6)

- Búsquedas en subdocumentos (1/2)

- Es posible realizar búsquedas exactas o de elementos puntuales.

- Coincidencia exacta.

```
{ "variants": {  
    "1": { "name": "Brown, 7" },  
    "2": { "name": "Black, 7" }  
}
```

- Coincidencia puntual.

```
{ "variants.2.name": "Black, 7" }
```

```
{  
  "_id": ObjectId("53f8659a957cf6a0a8b4595"),  
  "name": "The North Face McMurdo II Boot ...",  
  "primary_category_id": 8,  
  "price": 135.95,  
  "variants": {  
    "1": {  
      "name": "Brown, 7"  
    },  
    "2": {  
      "name": "Black, 7"  
    }  
  },  
  "category_ids": [ 8, 15 ],  
  "date_created": ISODate("2014-08-23T09:57:46Z")  
}
```



- Manejo básico de datos: listas y subdocumentos (6/6)
 - Búsquedas en subdocumentos (2/2)

```
> db.products.find( { "variants.1.name": "Brown, 7" } ).pretty()
{
    "_id" : ObjectId("53f8659a957cf6a0a8b4595"),
    "id" : 3,
    "name" : "The North Face McMurdo II Boot - Women's",
    "slug" : "the-north-face-mc-murdo-ii-boot-womens",
    "primary_category_id" : 8,
    "sku" : "TNF6706",
    "price" : 135.95,
    "weight" : 1,
    "variants" : {
        "1" : {
            "name" : "Brown, 7"
        },
        "2" : {
            "name" : "Black, 7"
        }
    },
    "is_active" : true,
    "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis que sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
    "category_ids" : [
        8,
        15
    ],
    "date_created" : ISODate("2014-08-23T09:57:46Z")
}
>
```

```
> db.products.find( { variants: { "1": { "name": "Brown, 7" }, "2": { "name": "Black, 7" } } } ).pretty()
{
    "_id" : ObjectId("53f8659a957cf6a0a8b4595"),
    "id" : 3,
    "name" : "The North Face McMurdo II Boot - Women's",
    "slug" : "the-north-face-mc-murdo-ii-boot-womens",
    "primary_category_id" : 8,
    "sku" : "TNF6706",
    "price" : 135.95,
    "weight" : 1,
    "variants" : {
        "1" : {
            "name" : "Brown, 7"
        },
        "2" : {
            "name" : "Black, 7"
        }
    },
    "is_active" : true,
    "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quis que sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
    "category_ids" : [
        8,
        15
    ],
    "date_created" : ISODate("2014-08-23T09:57:46Z")
}
>
```



- Manejo básico de datos: operadores de lectura (1/9)

- Cadenas que empiezan con \$ con semántica predefinida (1/2)

- Aplicable a filtros, proyecciones, modificaciones y otros.

- Ejemplos:

- Obtener todos los productos que cuesten más de 50 €.

```
> db.products.find( { "price": { "$gt": 50 } } )
```

- Obtener todos los productos que estén en la categoría 8 o que no estén activos.

```
> db.products.find( { "$or": [ { "category_ids": 8 }, { "is_active": false } ] } )
```



Base de Datos Documental MongoDB



- Manejo básico de datos: operadores de lectura (2/9)
 - Cadenas que empiezan con \$ con semántica predefinida (2/2)

```
> db.products.find( { "price": { "$gt": 50 } } )  
[{"_id": ObjectId("53f8659a957cf6a0a8b4593"), "id": 1, "name": "The North Face Ava Triclimate Jacket - Women's", "slug": "the-north-face-ava-triclimate-jacket-womens", "primary_category_id": 6, "sku": "TNF7814", "price": 229.95, "weight": 1, "variants": [{"1": {"name": "Blue, Small"}, "2": {"name": "Black, Small"}, "3": {"name": "Pink, Small"}}, {"is_active": true, "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdierit purus. Nunc imperdierit ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."}, {"category_ids": [6, 15], "date_created": ISODate("2014-08-23T09:57:46Z")}, {"_id": ObjectId("53f8659a957cf6a0a8b4594"), "id": 2, "name": "Burton AK 2L Summit Pant - Women's", "slug": "burton-ak-2l-summit-pant-womens", "primary_category_id": 7, "sku": "BUR5019", "price": 339.95, "weight": 1, "variants": [{"1": {"name": "Blue, Small"}, "2": {"name": "Grey, Small"}, "3": {"name": "Black, Small"}}, {"is_active": true, "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdierit purus. Nunc imperdierit ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."}, {"category_ids": [7, 14], "date_created": ISODate("2014-08-23T09:57:46Z")}, {"_id": ObjectId("53f8659a957cf6a0a8b4595"), "id": 3, "name": "The North Face McMurdo II Boot - Women's", "slug": "the-north-face-mcmurdo-ii-boot-womens", "primary_category_id": 8, "sku": "TNF6706", "price": 135.95, "weight": 1, "variants": [{"1": {"name": "Brown, 7"}, "2": {"name": "Black, 7"}}, {"is_active": true, "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdierit purus. Nunc imperdierit ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."}, {"category_ids": [8, 15], "date_created": ISODate("2014-08-23T09:57:46Z")}, {"_id": ObjectId("53f8659a957cf6a0a8b4596"), "id": 4, "name": "Arc'teryx Crossbow Jacket - Men's", "slug": "arcteryx-crossbow-jacket-mens", "primary_category_id": 9, "sku": "ARC1021", "price": 559.95, "weight": 1, "variants": [{"1": {"name": "Orange, Small"}, "2": {"name": "Brown, Small"}}, {"3": {"name": "Blue, Small"}}, {"is_active": true, "description": "Lorem ipsum dolor sit amet, consectetur adipiscin
```

```
> db.products.find( { "$or": [ { "category_ids": 8 }, { "is_active": false } ] } ).pretty()  
{  
    "_id": ObjectId("53f8659a957cf6a0a8b4595"),  
    "id": 3,  
    "name": "The North Face McMurdo II Boot - Women's",  
    "slug": "the-north-face-mcmurdo-ii-boot-womens",  
    "primary_category_id": 8,  
    "sku": "TNF6706",  
    "price": 135.95,  
    "weight": 1,  
    "variants": {  
        "1": {  
            "name": "Brown, 7"  
        },  
        "2": {  
            "name": "Black, 7"  
        }  
    },  
    "is_active": true,  
    "description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdierit purus. Nunc imperdierit ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",  
    "category_ids": [  
        8,  
        15  
    ],  
    "date_created": ISODate("2014-08-23T09:57:46Z")  
}  
|
```



- Manejo básico de datos: operadores de lectura (3/9)

- Operadores para filtros (1/5)

- Comparaciones.

\$gt, \$gte, \$lt, \$lte, \$ne, \$in, \$nin

```
> db.products.find( { "primary_category_id": { "$nin": [ 4, 6, 12 ] } } )
```

- Lógica.

\$or, \$nor, \$and, \$not

```
> db.products.find( { "$not": { "price": 54 } } )
```

- Campos.

\$exists, \$type

```
> db.products.find( { "price": { "$type": 2 } } )
```



<http://docs.mongodb.org/manual/reference/bson-types/>



- Manejo básico de datos: operadores de lectura (4/9)
 - Operadores para filtros (2/5)

```
> db.products.find( { "primary_category_id": { "$nin": [ 4, 6, 12 ] } } )  
[ { "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summit  
Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" :  
7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" :  
"Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } },  
"is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipis  
cing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nu  
nc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecena  
s dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices tu  
rpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse plat  
ea dictumst.", "category_ids" : [ 7, 14 ], "date_created" : ISODate("2014-08-23T09:57:  
46Z" ) }  
, { "_id" : ObjectId("53f8659a957cf6a0a8b4595"), "id" : 3, "name" : "The North Face McM  
Urdo II Boot - Women's", "slug" : "the-north-face-mc-murdo-ii-boot-womens", "primary_c  
ategory_id" : 8, "sku" : "TNF6706", "price" : 135.95, "weight" : 1, "varian  
ts" : { "name" : "Brown, 7" }, "2" : { "name" : "Black, 7" } }, "is_active" : true, "desc  
ription" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauri  
s enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat  
dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iacul  
is ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce p  
ellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids  
" : [ 8, 15 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }  
, { "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4, "name" : "Arc'teryx Crossbow  
Jacket - Men's", "slug" : "arcteryx-crossbow-jacket-mens", "primary_category_id" : 9,  
"sku" : "ARCI021", "price" : 559.95, "weight" : 1, "variants" : { "1" : { "name" : "0  
range, Small" }, "2" : { "name" : "Brown, Small" }, "3" : { "name" : "Blue, Small" } }  
, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipisc  
ing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nu  
nc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecena  
s dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices tu  
rpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse plate  
a dictumst.", "category_ids" : [ 9, 16 ], "date_created" : ISODate("2014-08-23T09:57:  
46Z" ) }  
, { "_id" : ObjectId("53f8659a957cf6a0a8b4597"), "id" : 5, "name" : "Columbia Bugaboot  
Plus XTM Boot - Men's", "slug" : "columbia-bugaboot-plus-xtm-boot-mens", "primary_cate
```

```
> db.products.find( { "price": { "$type": 2 } } )  
> ■
```

```
> db.products.find( { "price": { "$type": 1 } } )  
[ { "_id" : ObjectId("53f8659a957cf6a0a8b4593"), "id" : 1, "name" : "The North Face Ava  
Triclimate Jacket - Women's", "slug" : "the-north-face-ava-triclimate-jacket-womens",  
"primary_category_id" : 6, "sku" : "TNF7814", "price" : 229.95, "weight" : 1, "varia  
nts" : { "1" : { "name" : "Blue, Small" }, "2" : { "name" : "Black, Small" }, "3" : { "name" :  
"Pink, Small" } }, "is_active" : true, "description" : "Lorem ipsum dolor sit  
amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam,  
scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque a  
t convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nu  
nc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuad  
a. In hac habitasse platea dictumst.", "category_ids" : [ 6, 15 ], "date_created" : IS  
ODate("2014-08-23T09:57:46Z" ) }  
, { "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2, "name" : "Burton AK 2L Summi  
t Pant - Women's", "slug" : "burton-ak-2l-summit-pant-womens", "primary_category_id" :  
7, "sku" : "BUR5019", "price" : 339.95, "weight" : 1, "variants" : { "1" : { "name" :  
"Blue, Small" }, "2" : { "name" : "Grey, Small" }, "3" : { "name" : "Black, Small" } }  
, "is_active" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipis  
cing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nu  
nc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecena
```



➤ Manejo básico de datos: operadores de lectura (5/9)

➤ Operadores para filtros (3/5)

➤ Listas.

\$all, \$elemMatch, \$size

```
> db.products.find( { "category_ids": { "$all": [ 8, 15 ] } } )
```

➤ Avanzadas.

\$mod, \$regex, \$where

```
> db.products.find( { "sku": { "$regex": /^tnf/i } } )
```

```
> db.products.find( { "$where": "this.category_ids.indexOf(this.primary_category_id) == 1" } )
```

➤ Índices “especiales”.

\$geoWithin, \$geoIntersects, \$near, \$nearSphere,
\$text



- Manejo básico de datos: operadores de lectura (6/9)
 - Operadores para filtros (4/5)

```
> db.products.find( { "category_ids": { "$all": [ 8, 15 ] } } ).pretty()
{
  "_id" : ObjectId("53f8659a957cf6a0a8b4595"),
  "id" : 3,
  "name" : "The North Face McMurdo II Boot - Women's",
  "slug" : "the-north-face-mc-murdo-ii-boot-womens",
  "primary_category_id" : 8,
  "sku" : "TNF6706",
  "price" : 135.95,
  "weight" : 1,
  "variants" : {
    "1" : {
      "name" : "Brown, 7"
    },
    "2" : {
      "name" : "Black, 7"
    }
  },
  "is_active" : true,
  "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
  "category_ids" : [
    8,
    15
  ],
  "date_created" : ISODate("2014-08-23T09:57:46Z")
}
```

```
> db.products.find( { sku: { $regex: /^tnf/i } } )
{
  "_id" : ObjectId("53f8659a957cf6a0a8b4593"),
  "id" : 1,
  "name" : "The North Face Ave Triclimate Jacket - Women's",
  "primary_category_id" : 6,
  "sku" : "TNF7814",
  "price" : 229.95,
  "weight" : 1,
  "variants" : [
    {
      "1" : {
        "name" : "Blue, Small"
      },
      "2" : {
        "name" : "Black, Small"
      },
      "3" : {
        "name" : "Pink, Small"
      }
    },
    {
      "is_active" : true,
      "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."
    },
    {
      "1" : {
        "name" : "Blue, 7"
      },
      "2" : {
        "name" : "Black, 7"
      }
    }
  ],
  "is_active" : true,
  "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst.",
  "category_ids" : [
    6,
    15
  ],
  "date_created" : ISODate("2014-08-23T09:57:46Z")
},
{
  "_id" : ObjectId("53f8659a957cf6a0a8b4595"),
  "id" : 3,
  "name" : "The North Face McMurdo II Boot - Women's",
  "primary_category_id" : 8,
  "sku" : "TNF6706",
  "price" : 135.95,
  "weight" : 1,
  "variants" : [
    {
      "1" : {
        "name" : "Brown, 7"
      },
      "2" : {
        "name" : "Black, 7"
      }
    }
  ],
  "is_active" : true,
  "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."
},
{
  "_id" : ObjectId("53f8659a957cf6a0a8b4599"),
  "id" : 7,
  "name" : "The North Face Metropolis Down Parka - Girls",
  "primary_category_id" : 12,
  "sku" : "TNF6632",
  "price" : 235.95,
  "weight" : 1,
  "variants" : [
    {
      "1" : {
        "name" : "Black, 7"
      }
    }
  ],
  "is_active" : true,
  "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."
}
```



- Manejo básico de datos: operadores de lectura (7/9)
 - Operadores para filtros (5/5)

```
> db.products.find( { $where: "this.category_ids.indexOf(this.primary_category_id) =  
= 1" } )  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459b"), "id" : 9, "name" : "The North Face Den  
ali Glove - Men's", "slug" : "the-north-face-denali-glove-mens", "primary_category_id  
" : 19, "sku" : "TNF6426", "price" : 39.95, "weight" : 1, "variants" : [ { "id" : [ "0000  
" : "Grey" ], "2" : { "name" : "Black" }, "3" : { "name" : "Dark Blue" } }, "is_active  
" : true, "description" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Qu  
isque sed mauris enim. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris eg  
estas consequat dolor at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, p  
haretra a iaculis ut, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ul  
trices. Fusce pellentesque tempus arcu a malesuada. In hac habitasse platea dictumst."  
, "category_ids" : [ 15, 19 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459c"), "id" : 10, "name" : "Arc'teryx Bird He  
ad Beanie", "slug" : "arcteryx-bird-head-beanie", "primary_category_id" : 20, "sku" :  
"ARC0547", "price" : 44.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Grey" } }, "2  
" : { "name" : "Black" }, "3" : { "name" : "Purple" } }, "is_active" : true, "descript  
ion" : "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris en  
im. Fusce facilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dol  
or at vehicula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis u  
t, laoreet imperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pelle  
ntesque tempus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" :  
[ 16, 20 ], "date_created" : ISODate("2014-08-23T09:57:46Z" ) }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459d"), "id" : 11, "name" : "Electric EG2 Gogg  
le", "slug" : "electric-eg2-google", "primary_category_id" : 21, "sku" : "ELC0649", "p  
rice" : 129.95, "weight" : 1, "variants" : [ { "1" : { "name" : "Orange" } }, "2" : { "nam  
e" : "Grey" }, "3" : { "name" : "Black" } }, "is_active" : true, "description" : "Lore  
m ipsum dolor sit amet, consectetur adipiscing elit. Quisque sed mauris enim. Fusce fa  
cilisis est quam, scelerisque fringilla nunc. Mauris egestas consequat dolor at vehic  
ula. Pellentesque at convallis sem. Maecenas dolor mi, pharetra a iaculis ut, laoreet i  
mperdiet purus. Nunc imperdiet ultrices turpis vitae ultrices. Fusce pellentesque tem  
pus arcu a malesuada. In hac habitasse platea dictumst.", "category_ids" : [ 18, 21 ],  
"date_created" : ISODate("2014-08-23T09:57:46Z" ) }
```

Coge el valor del campo
primary_category_id y mira si
ocupa la **posición 2** en la lista de
category_ids.



- Manejo básico de datos: operadores de lectura (8/9)

- Operadores para proyecciones (1/2)

- Proyección sobre listas.

- Permite obtener sólo una parte de la lista del campo que se proyecta.
 - Si el campo no contiene una lista, no devuelve nada.

```
> db.products.find( { category_ids: 15 }, { name: 1, category_ids: 1 } )
```

- Recomendado para documentos con listas muy grandes.



- Manejo básico de datos: operadores de lectura (9/9)
 - Operadores para proyecciones (2/2)

```
> db.products.find( { category_ids: 15 }, { name: 1, category_ids: 1 } )  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4593"), "name" : "The North Face Ava Triclimate Jacket - Women's", "category_ids" : [ 6, 15 ] }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4595"), "name" : "The North Face McMurdo II Boot - Women's", "category_ids" : [ 8, 15 ] }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4599"), "name" : "The North Face Metropolis Down Parka - Girls'", "category_ids" : [ 12, 15 ] }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459b"), "name" : "The North Face Denali Glove - Men's", "category_ids" : [ 15, 19 ] }  
> █
```



➤ Manejo básico de datos: métodos de escritura (1/13)

➤ Insertar documentos

```
> db.<colección>.insert( <documento>, <opciones> )
```

➤ Si tiene _id

- Se comprueba que no existe otro documento con el mismo valor.

➤ Si no tiene _id

- Se añade un valor “autoincremental” para el campo _id.

```
> db.clientes.insert( {
    "nombre": "JJ",
    "apellidos": "Sánchez Peña",
    "edad": 42,
    "hijos": true,
    "telefonos": ["654126531", "91765321"],
    "dirección": {
        "calle": "C/MongoDB",
        "numero": 28,
        "población": "Tres Cantos",
        "codigo_postal": 28760
    }
})
```



➤ Manejo básico de datos: métodos de escritura (2/13)

```
> use clientes
switched to db clientes
> db.clientes.insert({
... "nombre": "JJ",
... "apellidos": "Sánchez Peña",
... "edad": 42,
... "hijos": true,
... "telefonos": ["654126531", "91765321"],
... "direccion": {
...     "calle": "C/MongoDB",
...     "numero": 28,
...     "poblacion": "Tres Cantos",
...     "codigo_postal": 28760
... }
... })
WriteResult({ "nInserted" : 1 })
> █
```

```
> db.clientes.findOne()
{
    "_id" : ObjectId("5b97d4081be33ebc6fce8c33"),
    "nombre" : "JJ",
    "apellidos" : "Sánchez Peña",
    "edad" : 42,
    "hijos" : true,
    "telefonos" : [
        "654126531",
        "91765321"
    ],
    "direccion" : {
        "calle" : "C/MongoDB",
        "numero" : 28,
        "poblacion" : "Tres Cantos",
        "codigo_postal" : 28760
    }
}
> █
```



- Manejo básico de datos: métodos de escritura (3/13)

- ObjectId

- Tipo especial de datos para almacenar identificadores.
 - 12 bytes de largo.
 - Sello de tiempo (4 bytes).
 - ID de máquina (3 bytes).
 - ID de proceso (2 bytes).
 - Contador de proceso local (3 bytes)
 - Diseñado para evitar colisiones.
 - El valor lo asigna el driver en el cliente y si no en el servidor.
 - Se puede usar en cualquier campo.



- Manejo básico de datos: métodos de escritura (4/13)

- Actualizar documentos (1/3)

- Objeto filtro para elegir qué documento realizar.
- Nuevo documento u objeto para indicar qué operación de actualización se va a realizar.
- Opciones:
 - Actualizar más de un documento a la vez.
 - Por defecto, sólo se modifica el primer documento encontrado.
 - Crear el documento si no existe.

```
> db.<colección>.update( <filtros>, <actualización>, { "multi": <múltiple>, "upsert": <crear>, ... } )
```



- Manejo básico de datos: métodos de escritura (5/13)

- Actualizar documentos (2/3)

- Ejemplo:

- Actualizar el documento añadiendo el campo DNI.
 - El valor del campo `_id` no se puede cambiar.

```
> db.clientes.update( { nombre: "JJ" } ,  
{ "dni": "00000001R" } )
```

!!Error!!

```
> db.clientes.update( { nombre: "JJ" } , { "dni": "00000001R", "nombre": "JJ", "apellidos":  
"Sánchez Peña", "edad": 42, "hijos": true, "telefonos": ["654126531","91765321"],  
"direccion": { "calle": "C/MongoDB", "numero": 28, "poblacion": "Tres Cantos",  
"codigo_postal": 28760 } } )
```



- Manejo básico de datos: métodos de escritura (6/13)
 - Actualizar documentos (3/3)

```
> db.clientes.update( { nombre: "JJ" }, { "dni": "00000001R" } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{ "_id" : ObjectId("5b97d4081be33ebc6fce8c33"), "dni" : "00000001R" }
> 
> db.clientes.findOne()
{
  "_id" : ObjectId("5b97dc9b1be33ebc6fce8c35"),
  "dni" : "00000000R",
  "nombre" : "JJ",
  "apellidos" : "Sánchez Peña",
  "edad" : 42,
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  }
}
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> 
```



➤ Manejo básico de datos: métodos de escritura (7/13)

➤ Borrar documentos.

➤ Objeto filtro para elegir qué documentos borrar.

➤ Opciones:

➤ Limitar borrado a un único documento.

```
> db.<colección>.remove( <filtros>, { "justOne": <único>, ... } )
```

➤ Borrar colecciones.

```
> db.<colección>.drop()
```

➤ Borrar bases de datos.

```
> db.dropDatabase()
```



- Manejo básico de datos: métodos de escritura (8/13)
 - Borrar documentos.

```
> db.clientes.remove( { nombre: "JJ" } )  
WriteResult({ "nRemoved" : 1 })  
> show collections  
clientes  
> show dbs  
admin      0.078GB  
clientes   0.078GB  
config     0.078GB  
fwd        0.078GB  
local      0.078GB  
> db.clientes.drop()  
true  
> show collections  
> show dbs  
admin      0.078GB  
clientes   0.078GB  
config     0.078GB  
fwd        0.078GB  
local      0.078GB
```

```
> db.dropDatabase()  
{ "dropped" : "clientes", "ok" : 1 }  
> show dbs  
admin      0.078GB  
config     0.078GB  
fwd        0.078GB  
local      0.078GB  
> █
```



- Manejo básico de datos: métodos de escritura (9/13)

- Operaciones combinadas (1/5)

- Guardar.

- Permite crear o actualizar según exista el registro.

```
> db.<colección>.save( <documento> )
```

- Ejemplos

```
> db.clientes.save({ "dni": "00000000R", "nombre": "JJ", "apellidos": "Sánchez Peña",
  "edad": 42, "hijos": true, "telefonos": ["654126531", "91765321"],
  "dirección": { "calle": "C/MongoDB", "numero": 28, "población":
  "Tres Cantos", "codigo_postal": 28760 } })
```



```
> db.clientes.save({ "dni": "00000000R", "nombre": "Juan José" })
```



- Manejo básico de datos: métodos de escritura (10/13)
 - Operaciones combinadas (2/5)

```
> db.clientes.save({ "dni": "00000000R", "nombre": "JJ", "apellidos": "Sánchez Peña", "edad": 42, "hijos": true, "telefonos": ["654126531", "91765321"], "direccion": { "calle": "C/MongoDB", "numero": 28, "poblacion": "Tres Cantos", "codigo_postal": 28760 } })  
WriteResult({ "nInserted" : 1 })  
> █  
    > db.clientes.save( { dni: "00000000R", nombre: "Juan José" } )  
    WriteResult({ "nInserted" : 1 })  
    > db.clientes.find().pretty()  
    {  
        "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
        "dni" : "00000000R",  
        "nombre" : "JJ",  
        "apellidos" : "Sánchez Peña",  
        "edad" : 42,  
        "hijos" : true,  
        "telefonos" : [  
            "654126531",  
            "91765321"  
        ],  
        "direccion" : {  
            "calle" : "C/MongoDB",  
            "numero" : 28,  
            "poblacion" : "Tres Cantos",  
            "codigo_postal" : 28760  
        }  
    }  
    {  
        "_id" : ObjectId("5b98aa0b1be33ebc6fce8c37"),  
        "dni" : "00000000R",  
        "nombre" : "Juan José"  
    }  
>
```



- Manejo básico de datos: métodos de escritura (11/13)

- Operaciones combinadas (3/5)

- Encontrar y actualizar:

- Modifica o borra el registro y obtiene su versión anterior.

```
> db.<colección>.findAndModify( { query: <filtro>, sort: <orden>, update: <actualización>,
    new: <nuevo>, fields: <proyección>, upsert: <crear>, ... } )

> db.<colección>.findAndModify( { query: <filtro>, sort: <orden>, remove: <borrar>,
    fields: <proyección> } )
```



```
> db.clientes.findAndModify( { query: { dni: "00000000R" }, update: { $inc: { edad: 1 } } } )

> db.clientes.findAndModify( { query: { dni: "00000000R" }, update: { $inc: { edad: -4 } },
    new: true } )
> db.clientes.findAndModify( { query: { dni: "00000000R" }, update: { $inc: { edad: 3 } },
    new: true, fields: { dni: 1, nombre: 1, edad: 1 } } )
```

- Manejo básico de datos: métodos de escritura (12/13)
 - Operaciones combinadas (4/5)

```
> db.clientes.findAndModify( {  
... query: { dni: "00000000R" },  
... update: { $inc: { edad: 1 } },  
... } )  
  
{  
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
    "dni" : "00000000R",  
    "nombre" : "JJ",  
    "apellido" : "Sánchez Peña",  
    "edad" : 42, 42  
    "hijos" : true,  
    "telefonos" : [  
        "654126531",  
        "91765321"  
    ],  
    "direccion" : {  
        "calle" : "C/MongoDB",  
        "numero" : 28,  
        "poblacion" : "Tres Cantos",  
        "codigo_postal" : 28760  
    }  
}  
|> OK  
  
> db.clientes.findOne()  
{  
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
    "dni" : "00000000R",  
    "nombre" : "JJ",  
    "apellido" : "Sánchez Peña",  
    "edad" : 43 43  
    "hijos" : true,  
    "telefonos" : [  
        "654126531",  
        "91765321"  
    ],  
    "direccion" : {  
        "calle" : "C/MongoDB",  
        "numero" : 28,  
        "poblacion" : "Tres Cantos",  
        "codigo_postal" : 28760  
    }  
}  
|> OK
```



- Manejo básico de datos: métodos de escritura (13/13)
 - Operaciones combinadas (5/5)

```
> db.clientes.findAndModify( {  
... query: { dni: "00000000R" },  
... update: { $inc: { edad: -4 } },  
... new: true  
... })  
{  
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
    "dni" : "00000000R",  
    "nombre" : "JJ",  
    "apellido" : "Sánchez Peña",  
    "edad" : 39  
    "hijos" : true,  
    "telefonos" : [  
        "654126531",  
        "91765321"  
    ],  
    "direccion" : {  
        "calle" : "C/MongoDB",  
        "numero" : 28,  
        "poblacion" : "Tres Cantos",  
        "codigo_postal" : 28760  
    }  
}>
```

```
> db.clientes.findAndModify( {  
... query: { dni: "00000000R" },  
... update: { $inc: { edad: 3 } },  
... new: true,  
... fields: { dni: 1, nombre: 1, edad: 1 } } )  
{  
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
    "dni" : "00000000R",  
    "nombre" : "JJ",  
    "edad" : 42  
}>
```



- Manejo básico de datos: operadores de escritura (1/10)
 - Actualización (1/5)

- Campos.

`$set, $unset, $rename, $setOnInsert`

```
> db.clientes.update( { dni: "00000000R" }, { "$set": { nombre: "Juan José" } } )
> db.clientes.update( { dni: "00000000R" }, { "$rename": { dni: "nif" } } )
> db.clientes.update( { nif: "00000000R" }, { "$unset": { edad: "" } } )
```

- Números.

`$inc, $mul, $min, $max, $bit`

```
> db.clientes.update( { nif: "00000000R" }, { "$max": { "edad": 42 } } )
> db.clientes.update( { nif: "00000000R" }, { "$min": { "edad": 38 } } )
```

- Fecha.

`$currentDate`

```
> db.clientes.update( { nif: "00000000R" }, { "$currentDate": { "fx_ult_modif": true } } )
> db.clientes.update( { nif: "00000000R" }, { "$currentDate": { "fx_ult_modif": { $type: "timestamp" } } } )
```



- Manejo básico de datos: operadores de escritura (2/10)
 - Actualización (2/5)

```
> db.clientes.update( { dni: "00000000R" }, { "$set": { nombre: "Juan José" } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "dni" : "00000000R",
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "edad" : 42,
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  }
}
> |
```

```
> db.clientes.update( { dni: "00000000R" }, { "$rename": { dni: "nif" } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "edad" : 42,
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  },
  "nif" : "00000000R"
}
>
```



- Manejo básico de datos: operadores de escritura (3/10)
 - Actualización (3/5)

```
> db.clientes.update( { nif: "00000000R" }, { "$unset": { edad: "" } } ) > db.clientes.update( { nif: "00000000R" }, { "$max": { "edad": 42 } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 }) WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne() > db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  },
  "nif" : "00000000R"
}
> [ { } ] > [ { } ]
```



- Manejo básico de datos: operadores de escritura (4/10)
 - Actualización (4/5)

```
> db.clientes.update( { nif: "00000000R" }, { "$min": { "edad": 38 } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
    "nombre" : "Juan José",
    "apellidos" : "Sánchez Peña",
    "hijos" : true,
    "telefonos" : [
        "654126531",
        "91765321"
    ],
    "direccion" : {
        "calle" : "C/MongoDB",
        "numero" : 28,
        "poblacion" : "Tres Cantos",
        "codigo_postal" : 28760
    },
    "nif" : "00000000R",
    "edad" : 38
}
```



- Manejo básico de datos: operadores de escritura (5/10)
 - Actualización (5/5)

```
> db.clientes.update( { nif: "00000000R" }, { "$currentDate": { "fx_ult_modif": true } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  },
  "nif" : "00000000R",
  "edad" : 38,
  "fx_ult_modif" : ISODate("2018-09-12T07:43:14.401Z")
}
> |
```

```
> db.clientes.update( { nif: "00000000R" }, { "$currentDate": { $type: "time
stamp" } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  },
  "nif" : "00000000R",
  "edad" : 38,
  "fx_ult_modif" : Timestamp(1536740175, 1)
}
> |
```



- Manejo básico de datos: operadores de escritura (6/10)

- Actualización (1/5)

- Listas (1/2).

- Añadir elementos.

\$push

```
> db.clientes.update( { nif: "00000000R" }, { "$push": { telefonos: "917367200" } } )
```

- Quitar elementos.

\$pop, \$pull, \$pullAll

```
> db.clientes.update( { nif: "00000000R" }, { "$pop": { telefonos: 1 } } ) //Último elemento

> db.clientes.update( { nif: "00000000R" }, { "$set": { idiomas: ["inglés", "francés",
    "alemán", "inglés", "alemán"] } } )

> db.clientes.update( { nif: "00000000R" }, { "$pull": { idiomas: { $in: [ "inglés" ] } } } )
> db.clientes.update( { nif: "00000000R" }, { "$pullAll": { idiomas: [ "alemán" ] } } )
```



- Manejo básico de datos: operadores de escritura (7/10)
 - Actualización (2/5).

```
> db.clientes.update( { nif: "00000000R" }, { "$push": { telefonos: "917367200" } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321",
    "917367200"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  },
  "nif" : "00000000R",
  "edad" : 38,
  "fx_ult_modif" : Timestamp(1536740175, 1)
}
```

```
> db.clientes.update( { nif: "00000000R" }, { "$pop": { telefonos: 1 } } )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.clientes.findOne()
{
  "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),
  "nombre" : "Juan José",
  "apellidos" : "Sánchez Peña",
  "hijos" : true,
  "telefonos" : [
    "654126531",
    "91765321"
  ],
  "direccion" : {
    "calle" : "C/MongoDB",
    "numero" : 28,
    "poblacion" : "Tres Cantos",
    "codigo_postal" : 28760
  },
  "nif" : "00000000R",
  "edad" : 38,
  "fx_ult_modif" : Timestamp(1536740175, 1)
}
```



- Manejo básico de datos: operadores de escritura (8/10)
 - Actualización (3/5).

```
> db.clientes.update( { nif: "00000000R" }, { "$pull": { idiomas: { $in: [ "inglés" ] } } } )    > db.clientes.update( { nif: "00000000R" }, { "$pullAll": { idiomas: [ "alemán" ] } } )  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })                                WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  
> db.clientes.findOne()  
>  
{  
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
    "nombre" : "Juan José",  
    "apellidos" : "Sánchez Peña",  
    "hijos" : true,  
    "telefonos" : [  
        "654126531",  
        "91765321"  
    ],  
    "direccion" : {  
        "calle" : "C/MongoDB",  
        "numero" : 28,  
        "poblacion" : "Tres Cantos",  
        "codigo_postal" : 28760  
    },  
    "nif" : "00000000R",  
    "edad" : 38,  
    "fx_ult_modif" : Timestamp(1536740175, 1),  
    "idiomas" : [  
        "francés",  
        "alemán",  
        "alemán"  
    ]  
}  
>
```



- Manejo básico de datos: operadores de escritura (9/10)

- Actualización (4/5).

- Listas (2/2).

- Coincidencia en filtro.

\$

```
> db.clientes.update( { nif: "00000000R", telefonos: "91765321" }, { "$set": { "telefonos.$": "917653210" } } )
```

- Trabajar como conjuntos, sin duplicados.

\$addToSet

```
> db.clientes.update( { nif: "00000000R" }, { "$addToSet": { "idiomas": [ "inglés", "francés", "alemán" ] } } )
```



- Manejo básico de datos: operadores de escritura (10/10)
 - Actualización (5/5).

```
> db.clientes.update( { nif: "00000000R", telefonos: "91765321" }, { "$set": { "telefonos.$": > db.clientes.update( { nif: "00000000R" }, { "$addToSet": { "idiomas": [ "inglés", "francés", "alemán" ] } } )  
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })  
> db.clientes.findOne()  
{  
    "_id" : ObjectId("5b98a7dc1be33ebc6fce8c36"),  
    "nombre" : "Juan José",  
    "apellidos" : "Sánchez Peña",  
    "hijos" : true,  
    "telefonos" : [  
        "654126531",  
        "917653210"  
    ],  
    "direccion" : {  
        "calle" : "C/MongoDB",  
        "numero" : 28,  
        "poblacion" : "Tres Cantos",  
        "codigo_postal" : 28760  
    },  
    "nif" : "00000000R",  
    "edad" : 38,  
    "fx_ult_modif" : Timestamp(1536740175, 1),  
    "idiomas" : [  
        "francés"  
    ]  
}  
>
```



Bases de Datos NoSQL



- Equivalencias SQL - mongoDB (1/3).
 - Conceptos:

Relacional	MongoDB
Base de datos	Base de datos
Tabla	Colección
Fila	Documento
Columna	Campo
Clave primaria	<code>_id</code>



➤ Equivalencias SQL - mongoDB (2/3).

➤ Consultas:

Relacional (SQL)	MongoDB
SELECT * FROM usuarios	db.usuarios.find()
SELECT id, nombre, estado FROM usuarios	db.usuarios.find({ }, { nombre: 1, estado: 1 })
SELECT nombre, estado FROM usuarios	db.usuarios.find({ }, { nombre: 1, estado: 1, _id: 0 })
SELECT * FROM usuarios WHERE estado = "A"	db.usuarios.find({ estado: "A" })
SELECT user_id, estado FROM usuarios WHERE estado = "A"	db.usuarios.find({ estado: "A" }, { user_id: 1, estado: 1, _id: 0 })



➤ Equivalencias SQL - mongoDB (3/3).

➤ Modificaciones:

Relacional (SQL)	MongoDB
INSERT INTO usuarios (nombre, edad, estado) VALUES ("Roberto", 45, "A")	db.usuarios.insert({ nombre: "Roberto", edad: 45, estado: "A" })
UPDATE usuarios SET status = "C" WHERE edad > 25	db.usuarios.update({ edad: { \$gt: 25 } }, { \$set: { estado: "C" } }, { multi: true })
UPDATE usuarios SET edad = edad + 3 WHERE estado = "A"	db.usuarios.update({ estado: "A" }, { \$inc: { edad: 3 } }, { multi: true })
DELETE FROM usuarios WHERE estado = "D"	db.usuarios.remove({ estado: "D" })
DELETE FROM usuarios	db.usuarios.remove({ })



Características principales

➤ Características

➤ Indexado.

- MongoDB soporta índices secundarios, permitiendo una buena acelaración en muchas consultas, pudiendo indexar cualquier campo de la base de datos.

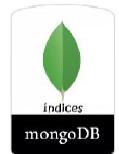
➤ Agregación.

- MongoDB viene con muchas funciones predefinidas que facilitan en gran medida el tratamiento de las colecciones almacenadas.

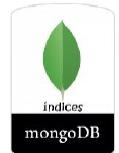
➤ Replicación y balanceo de carga.

- MongoDB utiliza un sistema de replicación empleando la arquitectura maestro-esclavo con el resto de máquinas. Además hace uso de un tipo de particionado horizontal llamado sharding.

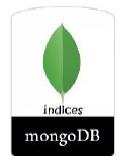
- Estas dos herramientas de escalado hacen de MongoDB una herramienta poderosa cuando se trata de crecer mucho en poco tiempo.



- Uso de índices (1/16)
 - Ejemplo de la guía telefónica (1/2)
 - Permite buscar teléfonos de personas.
 - Índice **agrupado** o **primario** por apellidos y nombre.
 - Los datos están ordenados físicamente en ese orden.
 - Sólo puede haber un índice agrupado.
 - ¿Cuánto cuesta buscar el teléfono asociado a una persona?
 - ¿Cuánto cuesta buscar el nombre asociado a un teléfono dado?



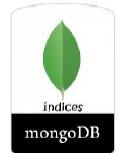
- Uso de índices (2/16)
 - Ejemplo de la guía telefónica (2/2)
 - Podríamos crear un nuevo listado.
 - Ordenado por número telefónico y con el nombre del propietario asociado a cada número.
 - Sería un índice **no agrupado o secundario**.
 - No almacena toda la información.
 - Se pueden crear todos los que sean necesarios.
 - ¿Cuánto cuesta buscar el nombre asociado a un teléfono?
 - ¿Cuánto cuesta buscar la dirección asociada a un teléfono dado?



➤ Uso de índices (3/16)

- Los datos se almacenan en orden de llegada (*orden natural*).
- Cuando un documento crece o se borra, se deja el hueco, que se reutilizará en el futuro.
- Si no se utilizan índices, los resultados se muestran por defecto.

```
> db.<colección>.find().sort( { "$natural": -1 } )
```

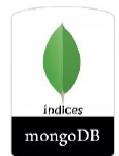


➤ Uso de índices (4/16)

```
> db.products.find( {}, { id: 1 }).sort( { $natural: -1 } )  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459d"), "id" : 11 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459c"), "id" : 10 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459b"), "id" : 9 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b459a"), "id" : 8 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4599"), "id" : 7 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4598"), "id" : 6 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4597"), "id" : 5 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4596"), "id" : 4 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4595"), "id" : 3 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4594"), "id" : 2 }  
{ "_id" : ObjectId("53f8659a957cf6a0a8b4593"), "id" : 1 }
```

```
> db.products.find( {}, { id: 1 })
```

¿SALIDA?

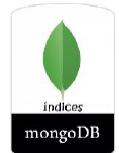


➤ Uso de índices (5/16)

➤ MongoDB permite y recomienda definir índices (1/2)

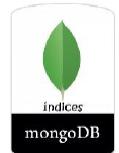
➤ Ventajas:

- Evita recorrer toda la colección para búsquedas.
- Permiten ordenar resultados rápidamente.
- Tamaño menor a la colección indexada.
- Más probable que entren completos en memoria, que hace que sean más eficientes las búsquedas.



➤ Uso de índices (6/16)

- MongoDB permite y recomienda definir índices (2/2)
- Desventajas:
 - Espacio de almacenamiento extra.
 - Mantenimiento adicional durante las escrituras.
 - Necesario cumplir condiciones para que se utilicen.
 - Sólo se usan si se filtra u ordena por los campos indexados.
 - Hay que comprobar que se está utilizando el índice.

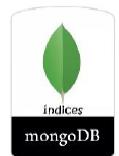


- Uso de índices (7/16)
 - Simple: un solo campo (1/2)
 - Comando de creación:
 - Se puede establecer el orden.
 - 1 para orden ascendente, -1 para orden descendente.

```
> db.<colección>.createIndex( { <campo>: <orden> } , <opciones> )
```

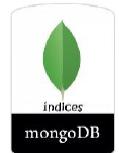
- El campo indexado puede pertenecer a un subdocumento.
 - Utilizar notación de puntos.

```
> db.<colección>.createIndex( { <campo_padre>.<campo_hijo>: <orden> } )
```



- Uso de índices (8/16)
 - Simple: un solo campo (2/2)
 - El campo indexado puede contener un subdocumento.
 - Búsquedas requieren coincidencia exacta.

```
{ "variant": { "name": "Orange", "price": 123.5 } }  
      !=  
{ "variant": { "price": 123.5, "name": "Orange" } }
```



➤ Uso de índices (9/16)

➤ Compuesto: varios campos (1/3)

➤ Influye el orden de aparición de los campos (1/3).

```
> db.<colección>.createIndex( { <campo1>: 1, <campo2>: -1 } )
```

```
> db.<colección>.find( ).sort( { <campo1>: 1, <campo2>: -1 } )
```



```
> db.<colección>.find( ).sort( { <campo2>: 1, <campo1>: -1 } )
```



```
> db.<colección>.find( ).sort( { <campo1>: -1, <campo2>: 1 } )
```



```
> db.<colección>.find( ).sort( { <campo1>: 1, <campo2>: 1 } )
```



- Uso de índices (10/16)
 - Compuesto: varios campos (2/3)
 - Influye el orden de aparición de los campos (2/3).

```
> db.<colección>.createIndex( { <campo1>: 1, <campo2>: -1 } )
```

```
> db.<colección>.find( ).sort( { <campo1>: -1, <campo2>: 1 } )
```

```
> db.<colección>.find( ).sort( { <campo1>: 1, <campo2>: -1 } )
```

Ascendente		Descendente	
Nombre	Edad	Nombre	Edad
Alberto	18	Marta	21
Ana	42	Luis	34
Cristina	39	Juan	25
Isabel	43	Juan	24
Isabel	25	Juan	23
Juan	25	Isabel	43
Juan	24	Isabel	25
Juan	23	Cristina	39
Luis	34	Ana	42
Marta	21	Alberto	18



➤ Uso de índices (11/16)

➤ Compuesto: varios campos (3/3)

➤ Influye el orden de aparición de los campos (3/3).

```
> db.<colección>.createIndex( { <campo1>: 1, <campo2>: -1, <campo3>: -1 } )
```

➤ Permite uso parcial.

➤ Siempre deben aparecer los primeros campos.

```
> db.<colección>.find( { <campo1>: <valor>, <campo2>: <valor> } )
```



```
> db.<colección>.find( { <campo1>: <valor> } )
```



```
> db.<colección>.find( { <campo2>: <valor> } )
```



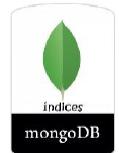
➤ Uso de índices (12/16)

➤ Multiclave: índices sobre campos que contienen listas (1/2)

```
> db.<colección>.createIndex( { <campo_lista>: 1 } )
```

- Se crea una entrada en el índice por cada elemento de la lista.
- Se puede usar en índices compuestos, pero sólo puede haber un campo con una lista como valor.

```
> db.<colección>.createIndex( { <campo_lista>.<campo>: 1 } )
```

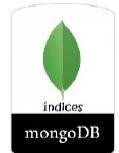


➤ Uso de índices (13/16)

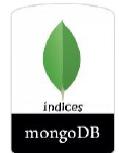
- Multiclave: índices sobre campos que contienen listas (2/2)
 - También se pueden indexar campos de subdocumentos contenidos en las listas.

```
{ "_id": ObjectId("53fc71231328e36cd97257c5"), "autor": "jj", "texto": "¿Esta es una entrada?",  
  "comentarios": [  
    { "autor": "pepe@ejemplo.com", "fecha": ISODate("2014-08-18T11:36:40.553Z"), "texto": "Si." },  
    { "autor": "juan@ejemplo.com", "fecha": ISODate("2014-08-22T08:23:21.938Z"), "texto": "No." }  
  ], ...  
}
```

```
> db.<colección>.createIndex( { "comentarios.autor": 1 } )
```



- Uso de índices (14/16)
 - Índices avanzados.
 - Hashed.
 - Indexa el hash del valor del campo.
 - Texto.
 - Permite realizar búsquedas de texto.
 - Geo-espacial.
 - Optimiza y potencia las búsquedas de documentos con información espacial.

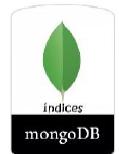


➤ Uso de índices (15/16)

➤ Valores únicos.

- Garantiza que cada documento tiene valores distintos para los campos del índice.
- La ausencia del campo se asigna al valor `null`, que tampoco puede estar repetido.
- No aplicable a índices hashed.
- El campo `_id` siempre tiene un índice ascendente único.

```
> db.<colección>.createIndex( { <campo>: 1 }, { "unique": true } )
```

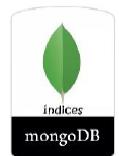


➤ Uso de índices (16/16)

➤ Campos poco frecuentes.

- Sólo tiene entradas para los documentos en los que está presente algún campo indexado.
 - Tamaño menor.
- Permite crear índices únicos sobre un campo ausente en muchos documentos.
- Puede no obtener resultados completos si se fuerza su uso.

```
> db.<colección>.createIndex( { <campo>: 1 }, { "sparse": true } )
```



- Métodos básicos de agregación (1/2)

- Métodos asociados a colecciones (1/2)

- Muy fáciles de usar.
- Flexibilidad limitada.
- Contar.
- Obtener elementos distintos.
- Agrupar documentos.



- Métodos básicos de agregación (2/2)

- Métodos asociados a colecciones (2/2)

- Count (contar).

- Devuelve el número total de registros.
- Aplicable también a cursores.

```
> db.<colección>.count()
```

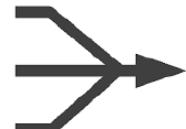
```
> db.<collection>.find( <filtro> ).count()
```

- Distinct (valores diferentes para un campo).

- Devuelve una lista con los diferentes valores.

```
> db.clientes.distinct( "nombre" )
```

```
> db.clientes.distinct("nombre")
[ "Juan José", "JJ", "Juanjo" ]
> █
```



➤ Aggregation Framework (1/10)

- Sistema de flujo de datos.
 - Una colección de origen.
 - Flujo a través de etapas que transforman los datos.
 - Devuelve un cursor o almacena datos en una colección.
 - Permite realizar JOINs (desde versión 3.2 – diciembre 2015).
- A medio camino entre métodos sencillos y MapReduce.
 - Complejidad media.
 - Flexibilidad media.
 - Rendimiento medio.



➤ Aggregation Framework (2/10)

- Aggregate (método acumular).
 - Acepta lista de operaciones a realizar.
 - Operación = operador + argumentos.
 - Opciones:
 - Incluir información de depuración.
 - Utilizar disco para operaciones grandes.
 - Tamaño de los lotes de resultados del cursor.

```
> db.<colección>.aggregate(  
    [ { <operador>: <argumentos> }, ... ],  
    { "explain": <depuración>, "allowDiskUse": <disco>, "cursor": { "batchSize": <tamaño> } }  
)
```



➤ Aggregation Framework (3/10)

➤ Operadores (1/7).

➤ Project (proyectar).

➤ Incluir o excluir campos.

- Excluir el campo `_id` explícitamente.

```
{ "$project": { <campo>: 1, <campo>: 1 } }
```

➤ Crear nuevos campos:

- Valores de otros campos utilizando `$` y su nombre.
- Expresiones similares a las de actualización, pero más potentes: comparaciones, condicionales, booleanas, aritméticas, cadenas, fechas, listas, conjuntos, variables, literales.

```
{ "$project": { <campo_nuevo>: "$otro_campo", <campo_nuevo>: <expresion> } }
```

➤ Aggregation Framework (4/10)

➤ Operadores (2/7).

➤ Match (filtrar documentos).

➤ Recomendable al principio.

```
{ "$match": <expresion> }
```

➤ Limit (limitar).

```
{ "$limit": <numero> }
```

➤ Skip (saltar).

```
{ "$skip": <numero> }
```

➤ Aggregation Framework (5/10)

➤ Operadores (3/7).

➤ Sort (ordenar).

➤ Expresión de orden.

```
{ "$sort": <expresion> }
```

➤ Por cercanía a un punto.

➤ Colección con un índice geo-espacial (almacenar datos geográficos).

```
{ "$geoNear": <opciones> }
```

➤ Aggregation Framework (6/10)

➤ Operadores (4/7).

➤ Group (agrupar).

- Utilizar expresiones como en la operación de proyección.
- Elegir el campo identificador.
- Aplicar operaciones de agregación para generar valores.

```
{ "$group": { "_id": <expresion>, <campo>: { <operador>: <expresion> }, ... } }
```

- Se pueden usar varios campos como identificador.

```
{ "$group": { "_id": { <campo1>: <expresion>, ... }, <campo>: { <operador>: <expresion> }, ... } }
```

\$addToSet, \$first,
\$last, \$max, \$min,
\$avg, \$push, \$sum

➤ Aggregation Framework (7/10)

➤ Operadores (5/7).

➤ Unwind (desagrupar).

- Convierte documentos con una lista de valores en un campo, en un conjunto de documentos con un único valor en dicho campo.

```
{ "$unwind": <expresion> }
```

➤ Out (almacenar).

- Debe ser la última operación de la lista.
- Almacena los resultados en la colección indicada.
 - La crea si no existe.
 - La reemplaza atómicamente si existe.
 - Si hay errores, no hace nada.

```
{ "$out": <colección> }
```

➤ Aggregation Framework (8/10)

➤ Operadores (6/7).

➤ Lookup (“join”) (1/2).

- Permite un left outer join de una colección no “shardeada” y que se encuentra en la misma BBDD.

```
{  
  $lookup:  
  {  
    from: <colección a aplicar el join>,  
    localField: <campo de los documentos de entrada>,  
    foreignField: <campo de los documentos de la “colección”>  
    as: <campo array donde se guardan los resultados>  
  }  
}
```



➤ Aggregation Framework (9/10)

➤ Operadores (7/7).

➤ Lookup (“join”) (2/2).

➤ Ejemplo:

Colección **orders**:

```
{ "_id": 1, "item": "abc", "price": 12, "qty": 2 }
{ "_id": 2, "item": "jkl", "price": 20, "qty": 1 }
{ "_id": 3 }
```

```
db.orders.aggregate ([
  $lookup:
    {
      from: "inventory",
      localField: "item",
      foreignField: "sku"
      as: "inventory_docs"
    }
])
```



Input → \$match → \$group → \$sort → output

Colección **inventory**:

```
{ "_id": 1, "sku": "abc", "desc": "p1", "stock": 120 }
{ "_id": 2, "sku": "def", "desc": "p2", "stock": 80 }
{ "_id": 3, "sku": "ijk", "desc": "p3", "stock": 60 }
{ "_id": 4, "sku": "jkl", "desc": "p4", "stock": 70 }
{ "_id": 5, "sku": null, "desc": "Incomplete" }
{ "_id": 6 }
```

```
{ "_id": 1, "item": "abc", "price": 12, "qty": 2, "inventory_docs": [
  { "_id": 1, "sku": "abc", "desc": "p1", "stock": 120 } ]
}
{ "_id": 2, "item": "jkl", "price": 20, "qty": 1, "inventory_docs": [
  { "_id": 4, "sku": "jkl", "desc": "p4", "stock": 70 } ]
}
{ "_id": 3, "inventory_docs": [
  { "_id": 5, "sku": null, "desc": "Incomplete" },
  { "_id": 6 } ] }
```

SALIDA

➤ Aggregation Framework (10/10)

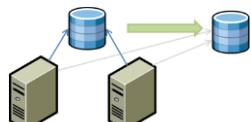
➤ Ejemplo.

- Obtener nombres de productos agrupados por categorías a las que pertenecen.

```
> db.products.aggregate(  
  { "$sort": { "name": 1 } },  
  { "$project": { "category_ids": 1, "name": 1 } },  
  { "$unwind": "$category_ids" },  
  { "$group": { "_id": "$category_ids", "products": { "$push": "$name" }, "total": { "$sum": 1 } } },  
  { "$sort": { "_id": 1 } }  
)
```

➤ Replicación (1/15)

- Almacenar los datos duplicados en varios servidores.
 - Redundancia de los datos.
 - Recuperación ante problemas.
 - Distribución de la carga.
 - Lecturas masivas.
 - Creación de copias de seguridad.
 - Creación de índices (posibilidad de utilizar ReplicaSets).



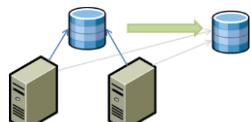
➤ Replicación (2/15)

➤ ReplicaSets: replicación en MongoDB (1/3)

➤ Permite crear una estructura de servidores que cumpla los objetivos planteados.

- Gestión automática.
- Sin cambios en cliente.
- Sin limitaciones en funcionalidad respecto a 1 servidor.

➤ Reemplaza la funcionalidad discontinuada maestro-esclavo.



➤ Replicación (3/15)

➤ ReplicaSets: replicación en MongoDB (2/3)

➤ Distintos tipos de miembros:

➤ Primario → Único nodo que recibe escrituras y las lecturas por defecto.

➤ Secundarios.

➤ Reciben lecturas (si se configura así).

➤ Replicación asíncrona, puede tener desfase de los datos.

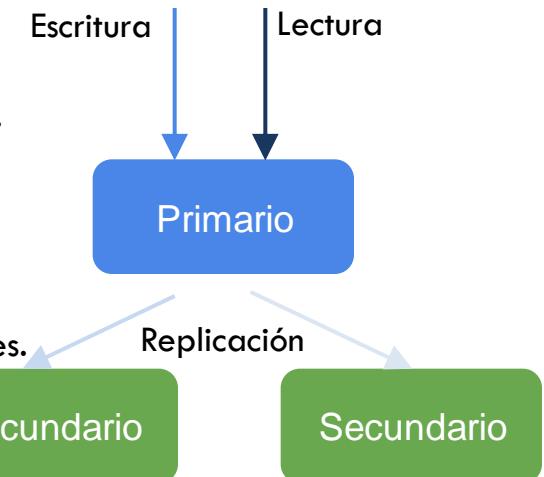
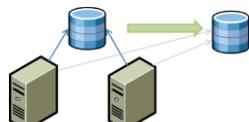
➤ Basada en registro de operaciones.

➤ Árbitros.

➤ No almacenan copias de los datos.

➤ Ayudar a elegir un nuevo nodo primario en las votaciones.

➤ Se recomienda su uso cuando se prevean empates.

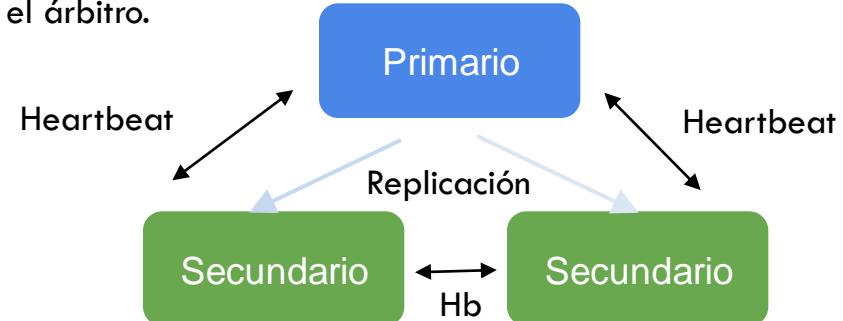
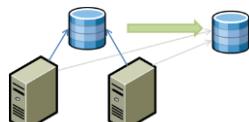


➤ Replicación (4/15)

➤ ReplicaSets: replicación en MongoDB (3/3)

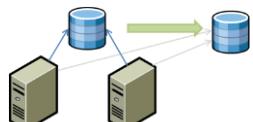
➤ Heartbeat

- Comunicación entre nodos: mensaje “keep alive” cada 2 segundos.
- Si el primario cae (se detecta si lleva 10 segundos sin responder):
 - Se debe elegir un nuevo primario entre ellos.
 - Los secundarios votan para decidir quién es el nuevo primario.
 - ¿Y si hay empate? Entra en juego el árbitro.



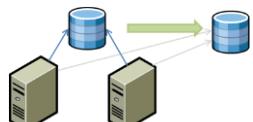
➤ Replicación (5/15)

- Recuperación ante fallos del servidor primario.
 - Hay que elegir un nuevo servidor primario.
 - No se permiten escrituras hasta que no se elija uno nuevo.
 - Se elige nuevo servidor entre nodos secundarios.
 - Prioridad para ser elegidos.
 - Los servidores con prioridad 0 no son elegibles.
 - Servidores que no tienen voto.
 - Sólo pueden tener voto hasta 7 nodos.
 - Un ReplicaSet puede tener hasta 12 miembros.



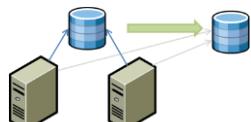
➤ Replicación (6/15)

- Elección del nuevo servidor primario.
 - Algún servidor convoca elecciones al no contactar con el servidor primario.
 - El nuevo primario debe cumplir:
 - Tiene la mayor prioridad.
 - Puede ver a una mayoría de servidores que tengan voto.
 - Está al día con los datos.
 - Conviene tener un número impar de servidores (mínimo 3).
 - Si se cae 1, cada servidor ve un número impar de nodos.



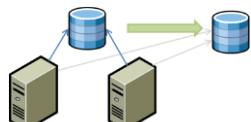
➤ Replicación (7/15)

- Servidores ocultos.
 - No son accesibles desde los clientes ni pueden ser primarios.
 - Pueden votar.
 - Pueden tener un retraso arbitrario con respecto al primario.
 - Ideales para copias de seguridad.



➤ Replicación (8/15)

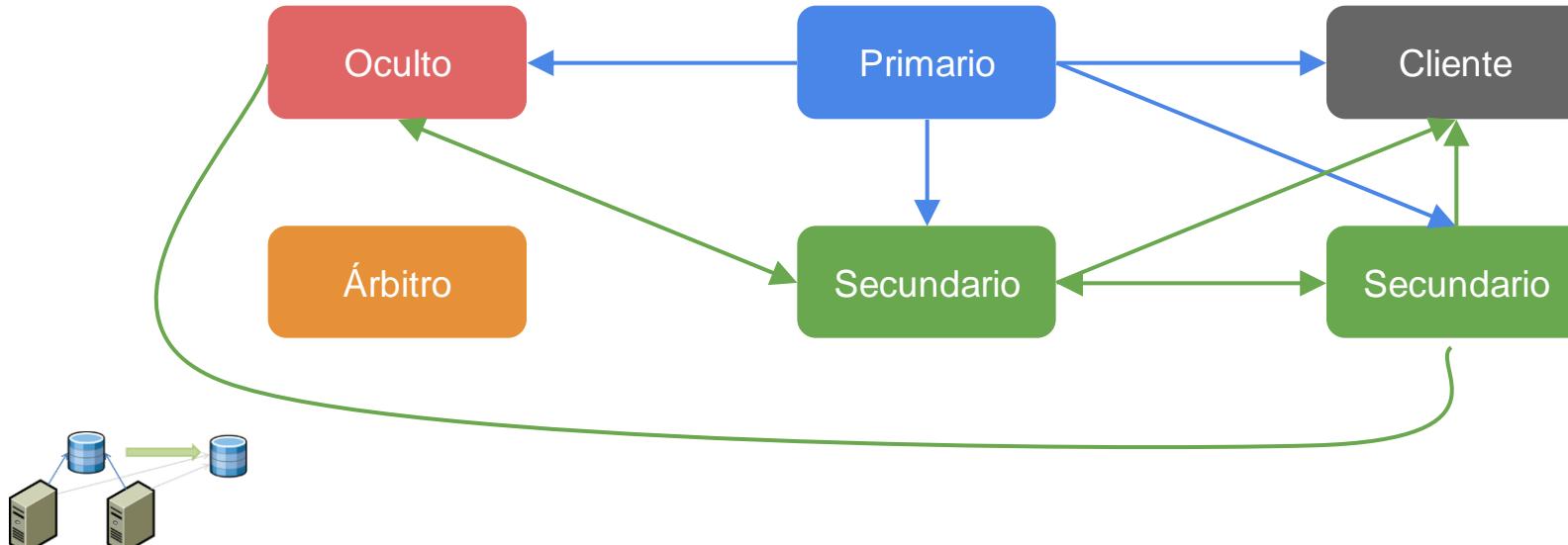
- Replicación encadenada.
 - Los servidores secundarios pueden replicar entre ellos.
 - Menor sobrecarga del servidor primario.
 - Mayor retraso en la sincronización de los datos.
 - Se puede desactivar.



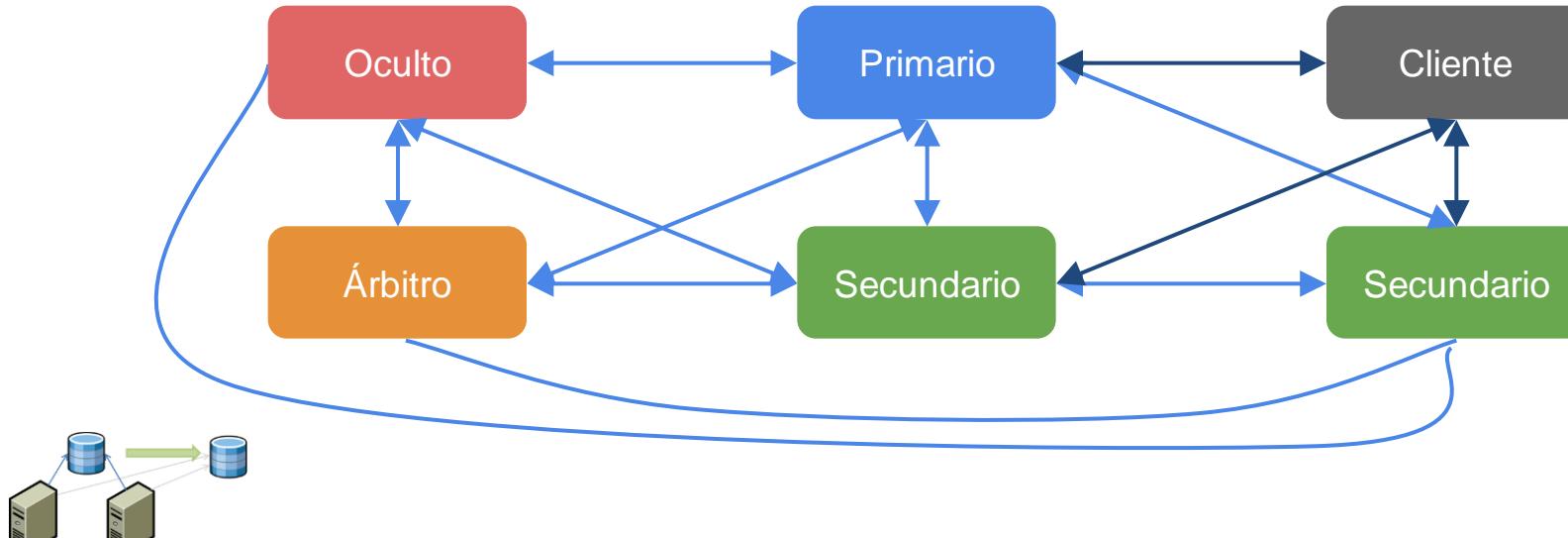
Base de Datos Documental MongoDB



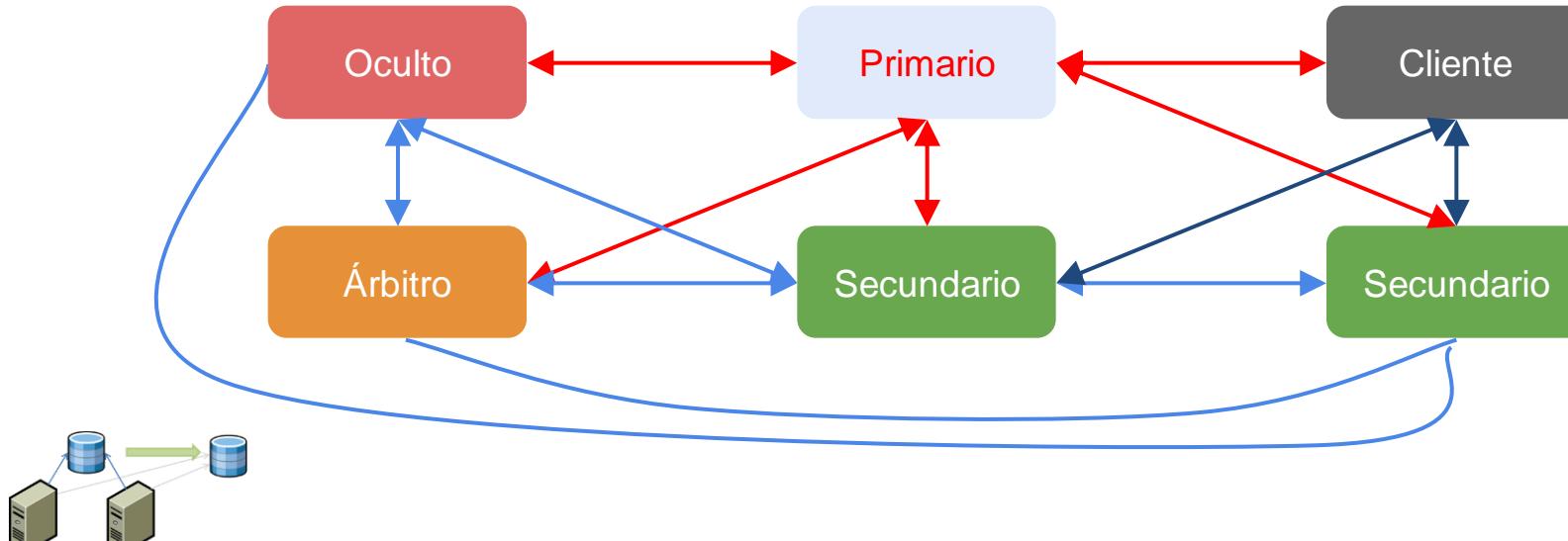
- Replicación (9/15)
 - Transferencia de datos.



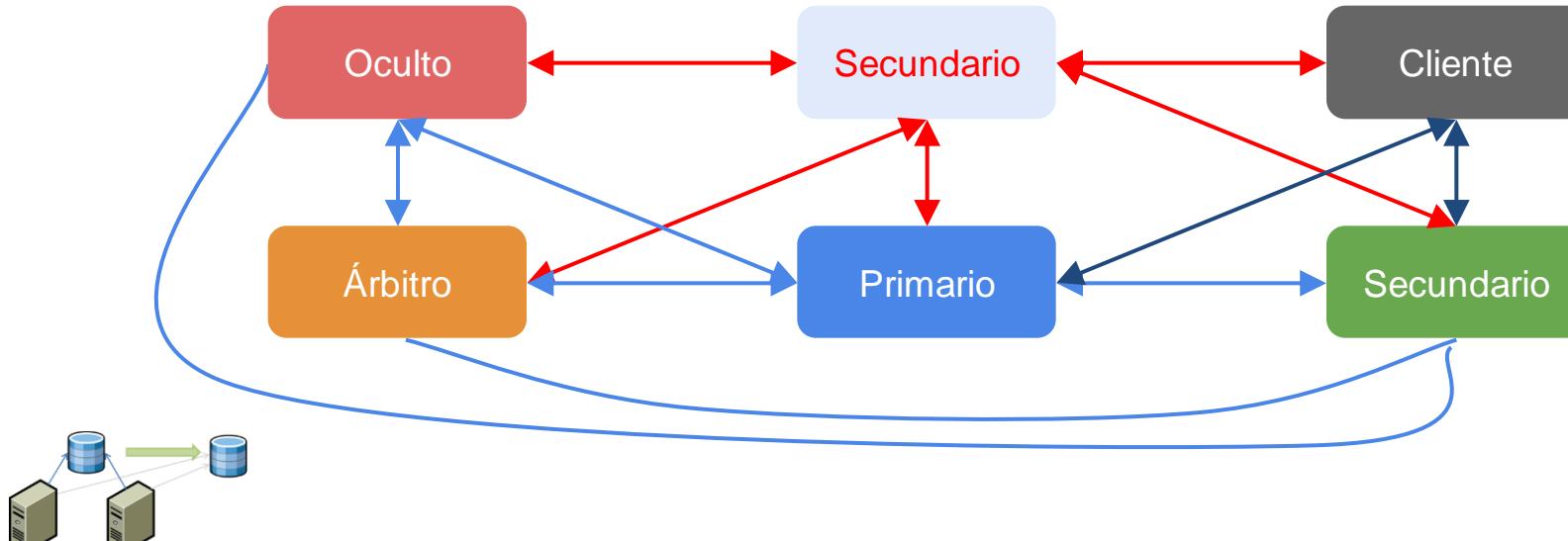
- Replicación (10/15)
 - Comprobación de estado.



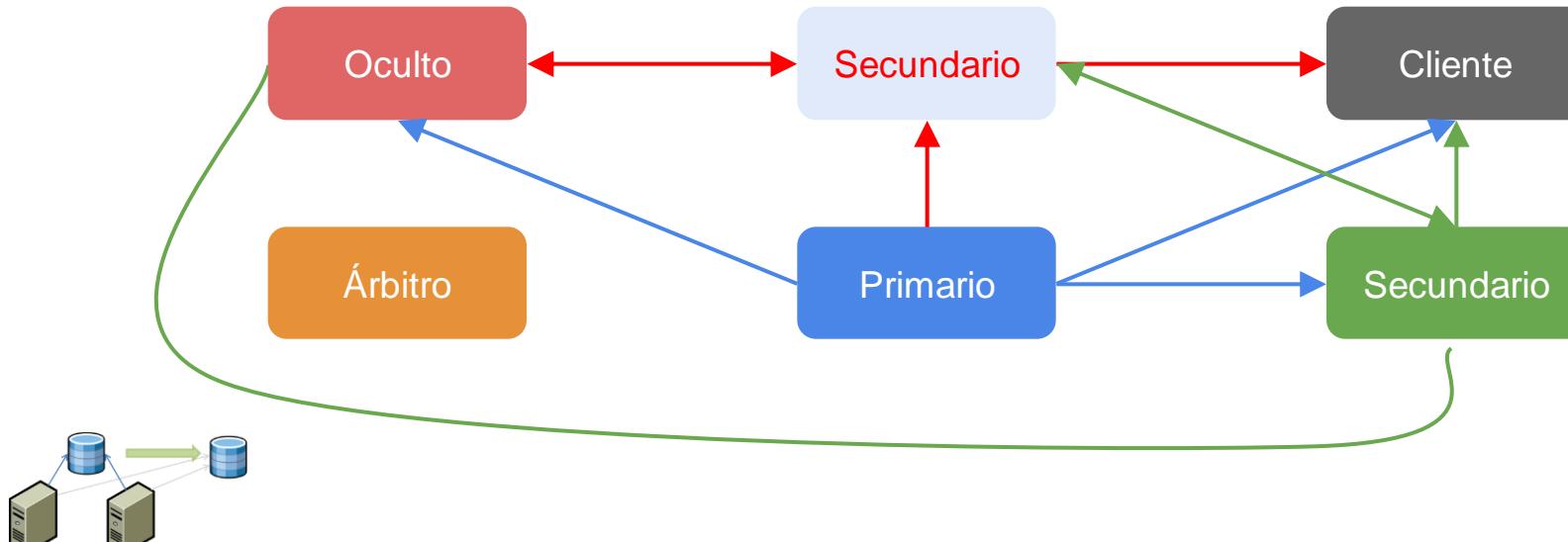
- Replicación (11/15)
 - Caída del servidor primario.



- Replicación (12/15)
 - Elección del nuevo servidor primario.

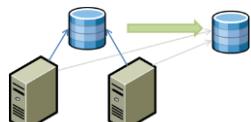


- Replicación (13/15)
 - Transferencia de datos con nuevo primario.



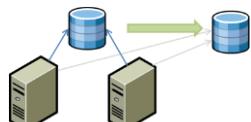
➤ Replicación (14/15)

- Cuando el servidor primario cae, pueden perderse los cambios (1/2)
 - Si los cambios no han sido transferidos a ningún otro miembro del ReplicaSet.
 - Otro miembro debe ejercer de primario y no conoce el cambio.
 - Cuando el servidor caído vuelve a conectar, debe descartar los cambios no propagados y ponerse al día con el primario.
 - Hay una forma de evitar que esto ocurra.
 - Preocupación por la escritura.



➤ Replicación (15/15)

- Cuando el servidor primario cae, pueden perderse los cambios (2/2)
 - Los cambios descartados se almacenan en BSON.
 - Se pueden ver con `bsondump` y aplicar manualmente `mongorestore`.
 - Hasta 300 MB de datos, si no hay que realizar el proceso manualmente.



- Distribución de datos – Sharding (1/14)
 - Manejo de conjuntos de datos extremadamente grandes.
 - CPU limita máximo de consulta a procesar.
 - Índices no caben en memoria.
 - Uso de disco penaliza el rendimiento.
 - Escalado
 - Vertical: mejores máquinas → caro y limitado.
 - Horizontal: más máquinas baratas e “ilimitadas”.



- Distribución de datos – Sharding (2/14)
 - Base de datos distribuida.
 - Los datos se parten en conjuntos disjuntos.
 - Cada conjunto va a un servidor.
 - Balanceo de carga.
 - Cada servidor maneja menos datos.
 - Las consultas puede afectar a uno o varios servidores.
 - Las escrituras sólo afectan a un servidor.
 - Penalización por gestión de comunicación entre ordenadores.



- Distribución de datos – Sharding (3/14)

- Sharding: distribución de datos en MongoDB.
 - Implementa una base de datos distribuida.
 - Indicar criterio para separar los datos (clave de sharding).
 - Montar infraestructura de servicios (ReplicaSets).
 - Distribución automática de los datos.
 - Distribución automática de las consultas y modificaciones.



- Distribución de datos – Sharding (4/14)

- Sharding. Elementos (1/2):

- Clúster.

- Conjunto de todos los servidores de un sharding.

- Shards.

- Partes en las que se organizan y almacenan los datos.

- “Shardearemos” colecciones y no BBDD enteras.

- Pueden consistir en un servidor o un ReplicaSet.

- Se recomienda utilizar un ReplicaSet para producción.



➤ Distribución de datos – Sharding (5/14)

➤ Sharding. Elementos (2/2):

➤ Servidores de configuración.

- Almacenan metadatos del clúster.
- Necesarios exactamente 3 para producción (requisito de MongoDB, al igual que en replicación, número impar).

➤ Enrutadores de consultas.

- No almacenan datos.
- Atiende a consultas y modificaciones de los clientes.
- Reparten cada operación al *shard* o a los *shards* que corresponde.



➤ Distribución de datos – Sharding (6/14)

➤ Shard Key (1/3):

- Se utiliza para distribuir una colección en varios shards.
- Clave que determina cómo se divide una colección en diferentes chunks.
- Se crea sobre uno o varios campos (Shard Key primer campo), dicho campo debe estar indexado.

```
> sh.shardCollection("db.alumnos", { NIA: "hashed" })
> sh.shardCollection("db.alumnos", { NIA: 1 }) //1 ascendente, -1 descendente
```

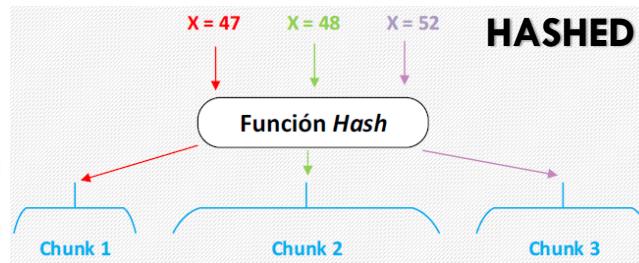
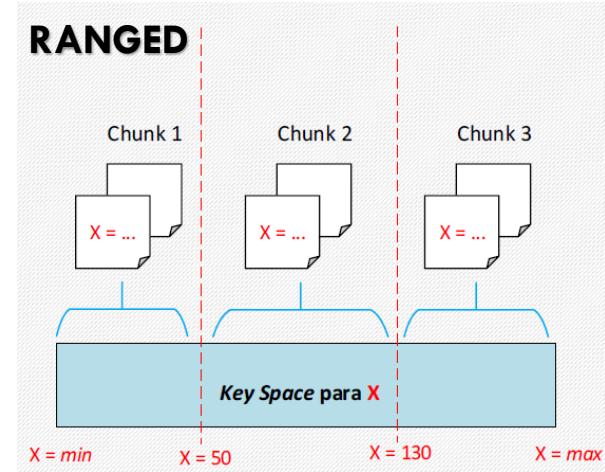
- Si la colección está vacía, esta instrucción crea automáticamente el índice si no existe.
- Si la colección tiene documentos, el índice ha de ser creado anteriormente.
- Puede ser **hashed** o **ranged**.



➤ Distribución de datos – Sharding (7/14)

➤ Shard Key (2/3):

- Los chunks se crean en base a rangos de valores de la Shard Key.
- Los documentos con un valor de Shard Key similar están probablemente en el mismo shard.
- Bueno para consultas.
- Si hay muchos documentos con un Shard Key en el mismo rango, habrá shards sobrecargados.



- No hay rangos.
- Los documentos se reparten de forma “aleatoria” entre los shards.
- Las consultas por rangos, requerirán acceder a varios shards (más lentas).

➤ Distribución de datos – Sharding (8/14)

➤ Shard Key (3/3):

➤ Características:

- Si en una consulta, la shard key:
 - Se incluye, sólo se consultarán aquellos shards que contengan los documentos requeridos.
 - No se incluye, se busca en todos los shards (búsqueda secuencial e ineficiente).
- En una colección que haga sharding, únicamente el campo **_id** y el índice definido en la Shard Key son únicos.



➤ Distribución de datos – Sharding (9/14)

➤ Sharding, ¿cuándo utilizar?

➤ Indicadores de que debe utilizarse:

- Se superan los límites de tamaño en la base de datos.
- No es posible aprovechar la caché.
 - No se usa más un subconjunto de los datos que el total.
- Necesidad de actualización de datos muy frecuente.

➤ Prever antes de llegar a estas situaciones.

- Más fácil desde el principio.
- Distribuir datos existentes es un proceso lento.
- Limitación de funcionalidades.



➤ Distribución de datos – Sharding (10/14)

➤ Criterio de partición de los datos.

➤ Clave de sharding:

- Campo que debe existir en todos los documentos.
- Se establecen rangos de valores dinámicos.

➤ Trozos (chunks)

- Conjunto de documentos comprendidos en un rango de valores de la clave, que tienen el tamaño máximo (64 MB por defecto).
- Cada trozo se asigna a un shard.



Base de Datos Documental MongoDB



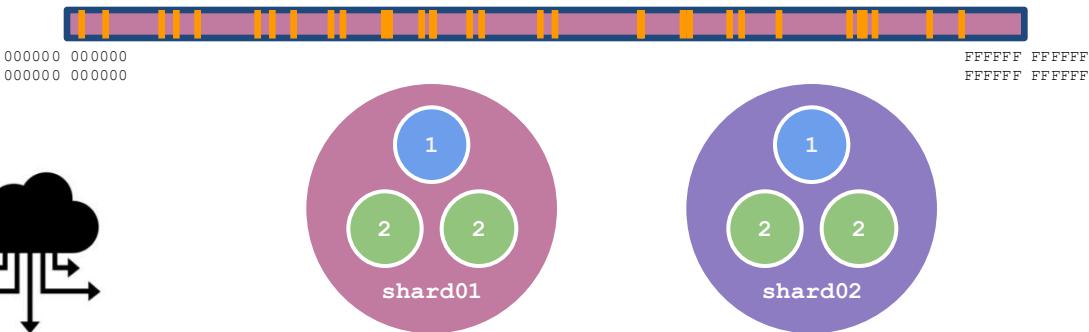
➤ Distribución de datos – Sharding (11/14)

➤ Partición de los datos de una colección (1/4)

- Primer trozo en primer shard.

config.chunks

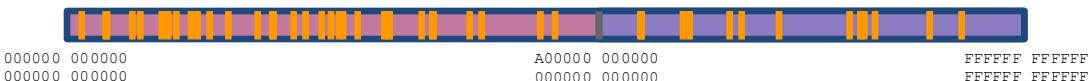
```
{  
  "min": { "_id": 000000 000000 000000 000000 },  
  "max": { "_id": FFFFFF FFFFFF FFFFFF FFFF },  
  "shard": "shard01"  
}
```



➤ Distribución de datos – Sharding (12/14)

➤ Partición de los datos de una colección (2/4)

- Se parte en dos el único trozo porque supera el tamaño máximo.



config.chunks

```
{  
  "min": { "_id": 000000 000000 000000 000000 },  
  "max": { "_id": 9FFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF },  
  "shard": "shard01"  
}  
  
{  
  "min": { "_id": A00000 000000 000000 000000 },  
  "max": { "_id": FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF },  
  "shard": "shard02"  
}
```



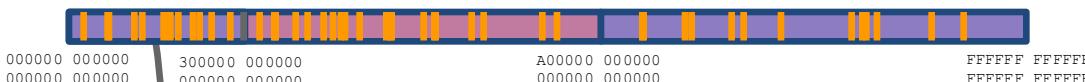
Base de Datos Documental MongoDB



➤ Distribución de datos – Sharding (13/14)

➤ Partición de los datos de una colección (3/4)

- En cada shard habrá múltiples trozos.



config.chunks

```
{  
  "min": { "_id": 000000 000000 000000 000000 },  
  "max": { "_id": 2FFFFF FFFFFF FFFFFF FFFFFF },  
  "shard": "shard02"  
}  
  
{  
  "min": { "_id": 300000 000000 000000 000000 },  
  "max": { "_id": 9FFFFF FFFFFF FFFFFF FFFFFF },  
  "shard": "shard01"  
}  
  
{  
  "min": { "_id": A00000 000000 000000 000000 },  
  "max": { "_id": FFFF 000000 000000 000000 },  
  "shard": "shard02"  
}
```



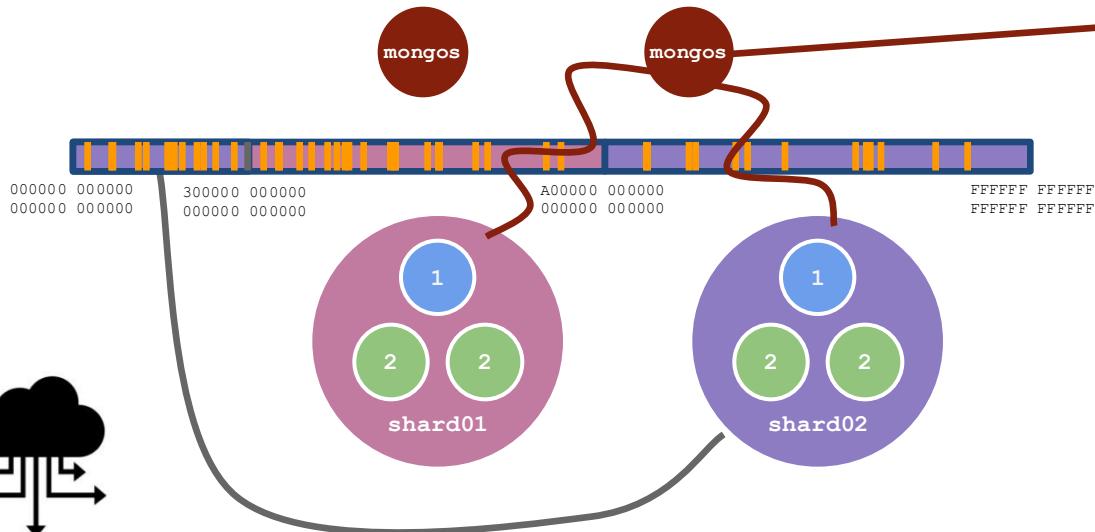
Base de Datos Documental MongoDB



➤ Distribución de datos – Sharding (14/14)

➤ Partición de los datos de una colección (4/4)

- El enrutador de consultas responde a las peticiones de cliente.



config.chunks

```
{  
  "min": { "_id": 000000 000000 000000 000000 },  
  "max": { "_id": 2FFFFF FFFFFF FFFFFF FFFF },  
  "shard": "shard02"  
}  
  
{  
  "min": { "_id": 300000 000000 000000 000000 },  
  "max": { "_id": 9FFFFF FFFFFF FFFFFF FFFF },  
  "shard": "shard01"  
}  
  
{  
  "min": { "_id": A00000 000000 000000 000000 },  
  "max": { "_id": FFFF FFFF FFFF FFFF },  
  "shard": "shard02"  
}
```

Algunos ejemplos

- Aadhaar (<https://es.wikipedia.org/wiki/Aadhaar>)
 - Proyecto de Identificación Única de la India.
 - Objetivo: capturar datos demográficos y biométricos de más de 1.200 millones de residentes → Almacena las imágenes en MongoDB y los datos demográficos en MySQL.
- eBay (<https://www.ebay.es/>)
 - Venta de productos por Internet.
 - MongoDB: sugerencias de búsqueda, almacenamiento de metadatos, administración de la nube y categorización de la comercialización.

- Foursquare (<https://es.foursquare.com/>)
 - Red social que permite a los usuarios marcar los lugares visitados y compartir la localización con amigos.
 - MongoDB fue elegido por:
 - Particionado automático. Es muy fácil añadir nodos al clúster.
 - Indexación geográfica. Facilita las búsquedas en base a dicho posicionamiento.
 - Los Replicaset. Ofrece alta disponibilidad y redundancia de nodos en caso de caídas.
 - Su modelo de datos dinámico y adaptable a las necesidades que se planteen.

- MTV Networks (<https://www.metlife.es/>)
 - Canal de TV + spike.tv + gametrailers.com + otros
 - MongoDB fue elegido por:
 - Modelo de datos flexible.
 - Las consultas e indexado de información son potentes y completos.
 - Facilidad de operaciones: escalabilidad y alta disponibilidad.

Base de Datos Documental MongoDB



- Shutterfly (<https://www.shutterfly.com/>)
 - Empresa de publicación de fotos e información personal.
 - Administra más de 6 billones de imágenes y hasta 10.000 operaciones por segundo.
 - MongoDB → para los metadatos asociados con las fotos cargadas.
 - Para operaciones transaccionales (facturación, gestión de cuentas, etc.), se utiliza un RDBMS tradicional.

MongoDB Atlas

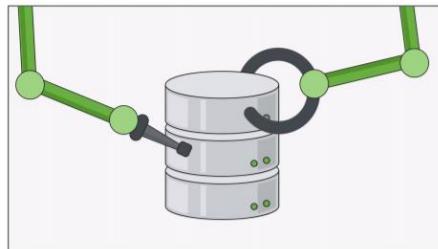
Base de Datos Documental MongoDB



➤ MongoDB Atlas (1 / 5)

<https://www.mongodb.com/cloud/atlas>

- La base de datos como servicio de MongoDB.



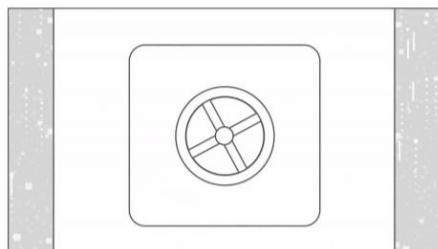
Automatización



Disponibilidad bajo demanda



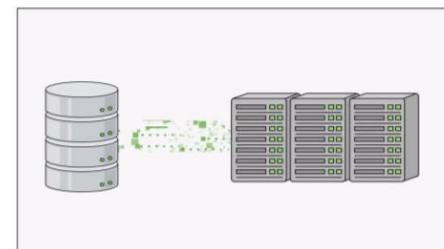
Gran escalabilidad



Seguridad



Alta disponibilidad



Copias de seguridad automáticas

- MongoDB Atlas (2/5) <https://www.mongodb.com/cloud/atlas>
- Características
 - *Automatización*: muy fácil crear, lanzar y escalar aplicaciones en MongoDB.
 - *Flexibilidad*: base de datos como servicio con todo lo necesario para los nuevos desarrollos.
 - *Seguridad*: ofrece varios niveles de seguridad.
 - *Escalabilidad*: gran escalabilidad sin interrumpir la actividad.
 - *Alta disponibilidad*, implementando funciones de tolerancia a errores y auto-reparación.
 - *Alto rendimiento*.

- MongoDB Atlas (3/5) <https://www.mongodb.com/cloud/atlas>
- Ventajas (1/2)

Ejecución	Protección y Seguridad	Libertad de movimiento
Puesta en marcha de un clúster en segundos.	Autenticación y cifrado.	Modelo de planes de precio según demanda: se factura por hora.
Implementaciones replicadas y sin interrupción.	Copias de seguridad continuas con recuperación en un punto en el tiempo.	Compatible con diferentes tipos de servicios cloud: AWS, GCP y Azure)

➤ MongoDB Atlas (4/5)

<https://www.mongodb.com/cloud/atlas>

➤ Ventajas (2/2)

Ejecución	Protección y Seguridad	Libertad de movimiento
Total escalabilidad tanto horizontal como vertical: con unos clics y sin interrumpir la actividad.	Supervisión detallada y alertas personalizadas.	Parte de un paquete de productos y servicios para todas las fases de la aplicación.
Revisiones automáticas y actualizaciones para las nuevas funciones de MongoDB.		

Base de Datos Documental MongoDB



➤ MongoDB Atlas (5/5)

➤ Zonas de distribución

➤ 18 zonas de AWS:

- América del Norte, América del Sur, Europa, Australia.

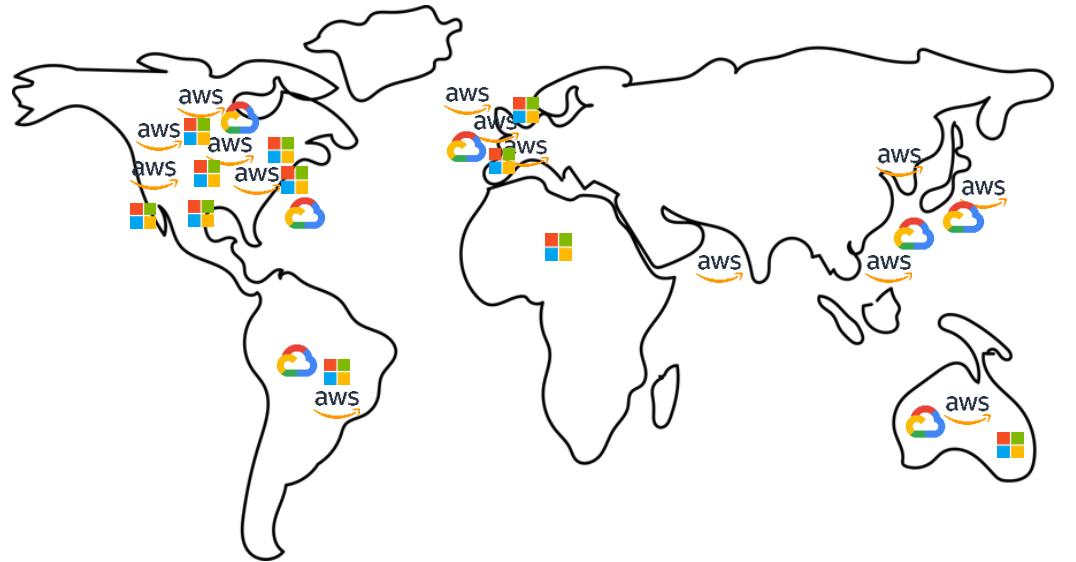
➤ 33 zonas de Azure:

- América del Norte, América del Sur, Europa, Asia, Australia, África.

➤ 21 zonas GCP:

- América del Norte, América del Sur, Europa, Australia, Asia.

<https://www.mongodb.com/cloud/atlas>



Cassandra

1. Introducción
2. Almacenamiento de registros
3. Casos de uso
4. Características
5. Rendimiento
6. Cassandra vs SGBD
7. CQL
8. Arquitectura
9. Replicación
10. Consistencia
11. Conclusiones



Introducción

Base de Datos Cassandra



➤ Introducción (1/36)

- Motor de BBDD orientada a columna.
- Gratuita.
 - Código abierto en java.
- Multiplataforma.
- Fusiona DynamoDB⁽¹⁾ de Amazon (paper 2007) con BigTable⁽²⁾ de Google (paper 2006), que era código cerrado.

(1) https://docs.aws.amazon.com/dynamodb/index.html#lang/es_es

(2) <https://cloud.google.com/bigtable/>

➤ Introducción (2/36)

<http://cassandra.apache.org/>

➤ Orígenes

- Julio 2008 es desarrollado y utilizado por Facebook, para intentar solventar la problemática relacionada con el rendimiento del motor de búsquedas.
- Febrero 2010 pasa a ser proyecto de primer nivel de Apache.
- Desde versión 0.8, tenemos CQL (Cassandra Query Language).
- Drivers para la mayoría de los lenguajes:
 - C++, .NET, Java, Node.js, Perl, PHP, Python, Ruby, Go, Clojure, Haskell, ...

➤ Introducción (3/36)

➤ Funcionalidades (1/7)

- La topología es la de un anillo a través del cual se distribuyen los datos para minimizar cuellos de botella en el acceso a los mismos.
- Distribuida a través de los diferentes nodos y descentralizada.
 - No uso de tecnología maestro-esclavo.
 - Todos los nodos son idénticos.
 - Uso de protocolos P2P y Gossip⁽¹⁾ para mantener los nodos sincronizados.
 - No hay un único punto de fallo.
- Fácil de trabajar y mantener porque todos los nodos son iguales.

(1) https://en.wikipedia.org/wiki/Gossip_protocol

Base de Datos Cassandra



➤ Introducción (4/36)

➤ Funcionalidades (2/7)

➤ Altamente escalable:

- Escalabilidad horizontal: eliminar/añadir nodos sin afectar a su funcionamiento.
- No son necesarios cambios en las aplicaciones ni reiniciar procesos.
- No hay que balancear la carga al añadir/eliminar nodos.

➤ Alta disponibilidad y tolerancia a fallos:

- Siempre disponible.
- No hay tiempo de inactividad.

<http://cassandra.apache.org/>

Aaron Turner (@synfinatic)
took me 10hrs to notice a #cassandra node had a hw failure because everything just kept working.
#sweet

4 RETWEETS 1 FAVORITE

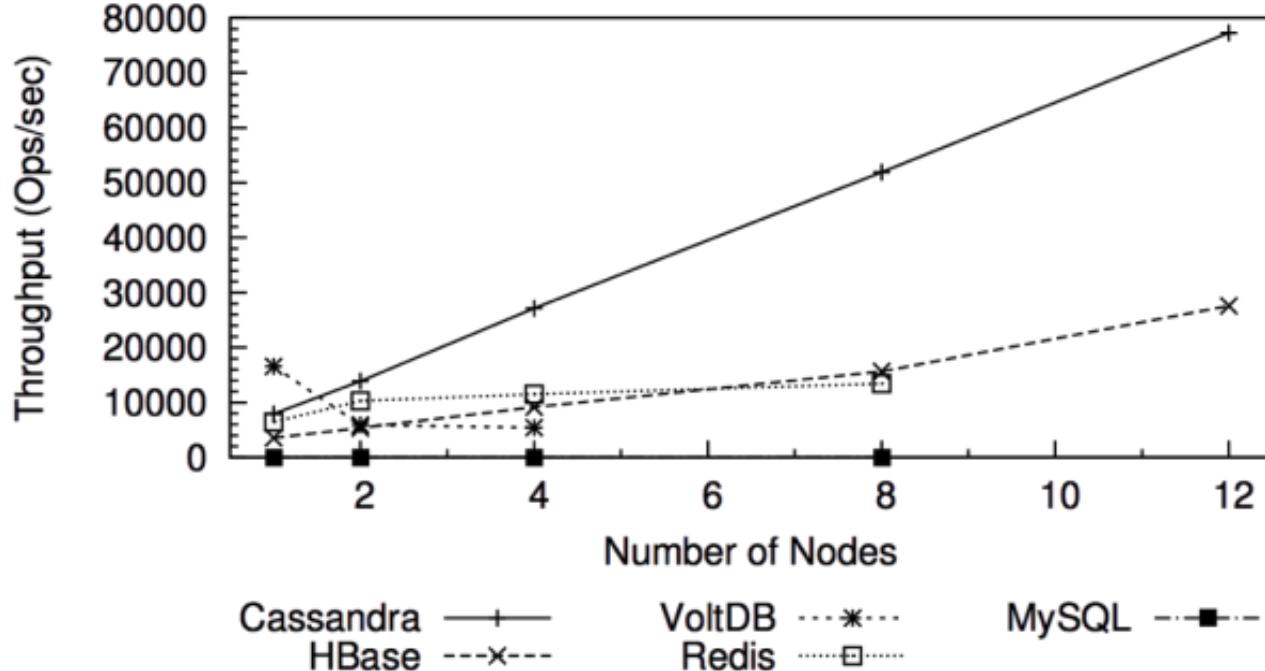
Base de Datos Cassandra



➤ Introducción (5/36)

<http://cassandra.apache.org/>

➤ Funcionalidades (3/7)



➤ Introducción (6/36)

<http://cassandra.apache.org/>

➤ Funcionalidades (4/7)

➤ Gran rendimiento lineal de escala:

- Cassandra es escalable linealmente, es decir, aumenta su rendimiento a medida que aumenta el número de nodos del clúster.
- Mantiene un tiempo de respuesta rápido.

➤ Almacenamiento de datos flexible:

- Datos estructurados, semiestructurados y no estructurados.
- Permite alojar de forma dinámica los cambios en sus estructuras de datos de acuerdo a su necesidad.

Base de Datos Cassandra



➤ Introducción (7/36)

<http://cassandra.apache.org/>

➤ Funcionalidades (5/7)

➤ Distribución de datos fácil:

- Proporciona flexibilidad para distribuir los datos mediante la replicación de datos a través de múltiples centros de datos.

➤ Soporte de transacciones:

- Admite propiedades ACID (Atomicidad, Coherencia, Aislamiento y Durabilidad).

➤ Estructuras rápidas:

- Diseñado para ejecutarse en HW barato.
- Lleva a cabo escrituras de forma muy rápida, pudiendo almacenar cientos de terabytes de datos, sin sacrificar la eficiencia de lectura.

Base de Datos Cassandra



➤ Introducción (8/36)

<http://cassandra.apache.org/>

➤ Funcionalidades (6/7)

- Permite ajustar la consistencia de los datos en base a los requerimientos de la aplicación: datos consistentes eventualmente vs datos consistentes inmediatamente.
- Diseñado para manejar grandes cantidades a través de múltiples servidores.
- Contiene grandes cantidades de datos no organizados.
- Fácil de implementar y desplegar.
- Imita a los SGBD tradicionales, pero con *triggers* y *transacciones* ligeros.

Base de Datos Cassandra



➤ Introducción (9/36)

<http://cassandra.apache.org/>

➤ Funcionalidades (7/7)

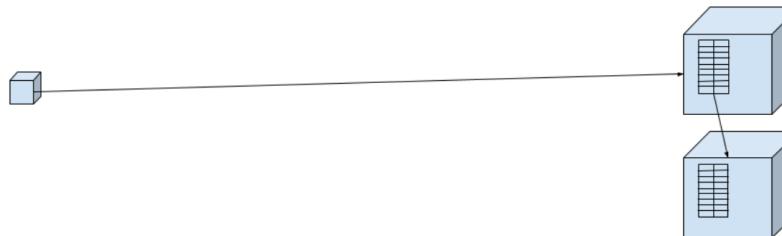
➤ Gran rendimiento:

- Diseñado para aprovechar la ventaja de la tecnología de las máquinas multiprocesador/multicore y para trabajar sobre máquinas alojadas en diferentes centros de datos (datacenter).
- Trabaja perfectamente con cientos de TB.
- Gran rendimiento incluso cuando el sistema está muy cargado.

- Introducción (10/36)

- Protocolos Gossip (1/2)

- Se utiliza para descubrir la localización y la información de estado de los otros nodos que participan en el clúster de Cassandra.
- Red de protocolos de comunicación inspirados en la propagación de rumores como en la vida real.
- Comunicación periódica, entre cada par de nodos.
- Selección aleatoria de parejas.



Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Introducción (11/36)

➤ Protocolos Gossip (2/2)

➤ Ejemplo: el nodo A desea buscar un patrón en los datos.

- RONDA1: el nodo A busca localmente y luego “cotillea” con el nodo B.
- RONDA2: los nodos A y B “cotillean” con los nodos C y D.
- RONDA3: los nodos A, B, C y D “cotillean” con otros cuatro nodos ...

➤ La duplicación ronda a ronda hace que el protocolo sea muy robusto.

➤ Introducción (12/36)

➤ Modelo relacional

- Cada BBDD contiene tablas.
- Cada tabla tiene su propio nombre y está formada por filas (tuplas).
- Cada tupla está formada por columnas. Cada columna tiene su nombre.
 - Añadimos datos a las tablas, dando un valor a cada columna; aquella columna que no tiene valores, se le da el valor NULL.
- Las filas se corresponden a registros.
- Cada registro tiene una clave primaria, la cual permite el acceso a dicho registro.
- Podemos actualizar algunos o todos los registros.
- Este modelo se rige bajo el modelo SQL.

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Introducción (13/36)

➤ Modelo de datos Cassandra (1/19)

- Supongamos que tenemos una lista con varios valores:

Ricardo

Sánchez

rsanchez@uam.es

- Si quisiéramos consultar dichos valores, tendríamos que examinar valor por valor para así saber qué se está guardando en cada celda.
- **SOLUCIÓN:**
 - Añadir otra dimensión a la lista, permitiéndome representar una descripción de los datos, formando un mapa estructurado:

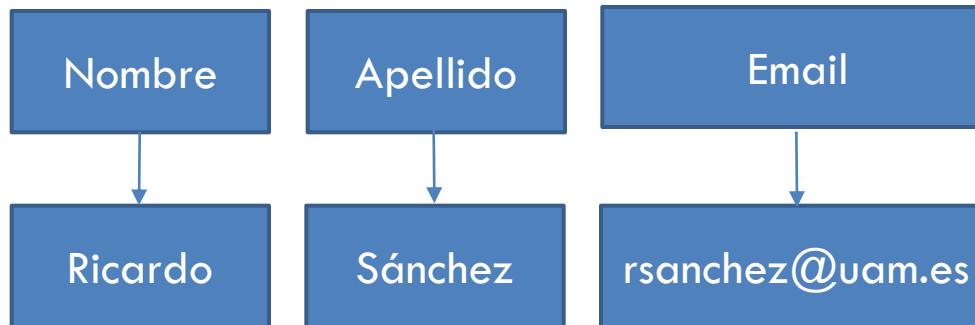
Base de Datos Cassandra



➤ Introducción (14/36)

➤ Modelo de datos Cassandra (2/19)

<http://cassandra.apache.org/>



- Esta estructura únicamente nos permite añadir un registro.
- Para almacenar varios registros, se necesitarían varias de estas estructuras.
- Tampoco tenemos nada que nos unifique los datos de una columna, para diferenciar un registro de otro → Clave.

➤ Introducción (15/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra (3/19)

- Cassandra no soporta un modelo de datos relacional completo.
- Proporciona clientes con un modelo de datos simple que soportan un control dinámico sobre la disposición de los datos y el formato.
- Cassandra define una familia de columnas (“*column family*”), para asociar datos similares.
- Una column family es análoga a una tabla en el modelo relacional.
- Dos estructuras básicas:
 - Column: son pares nombre/valor.
 - Column Family: contenedor de registros que contienen columnas similares.

➤ Introducción (16/36)

➤ Modelo de datos Cassandra (4/19)

- En BBDD relacionales, los nombres de columnas siempre son de tipo cadena; sin embargo en Cassandra tanto las claves de registro como las columnas pueden ser cadenas, enteros u otro tipo.
- En Cassandra no tenemos que guardar un valor para cada columna a la hora de crear un nuevo registro. Ejemplo: un usuario tiene un segundo número de teléfono, en lugar de poner NULL en aquellos usuarios que no tienen un segundo número, que ocuparía espacio, no se tiene en cuenta la columna de ese registro.
- Ejemplo:

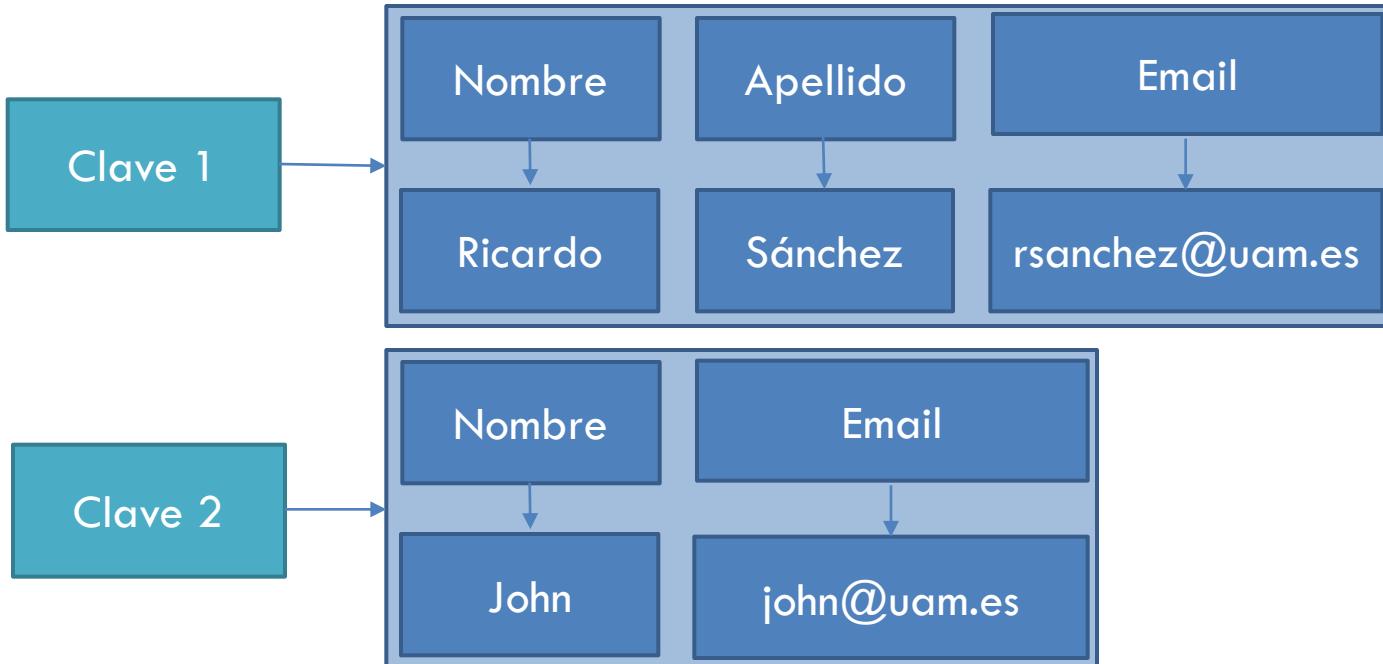
Base de Datos Cassandra



➤ Introducción (17/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra (5/19)



Base de Datos Cassandra



➤ Introducción (18/36)

➤ Modelo de datos Cassandra (6/19)

➤ El modelo de datos de Cassandra define cinco tipos de estructuras de datos:

- Column
- Column Family
- Super Column
- Keyspace
- Cluster

<http://cassandra.apache.org/>

➤ Introducción (19/36)

➤ Modelo de datos Cassandra (7/19)

➤ Column (1/3)

- Unidad mínima de información en Cassandra. Cada columna es una tupla {nombre, valor, timestamp}. Estos valores son todos suministrados por el cliente.
- El valor de cada columna está formado por un array de bytes.
- Son inmutables para evitar problemas de multithreading.
- Se organizan dentro de las *columns families*.
- No es necesario definir todas las columnas en el momento de diseñar la base de datos.
- Se ordenan por un tipo. Pueden ser: [AsciiType](#), [BytesType](#), [LongType](#), [TimeUUIDType](#) y [UTF8Type](#).

- Introducción (20/36)

- Modelo de datos Cassandra (8/19)

- Column (2/3)

- Tiene asociada las filas, que es el lugar donde va almacenado el valor al que hace referencia al nombre de la columna.
- La información almacenada que hace referencia a esas columnas se le aplica un hash para generar la **row-key**, para diferenciar cada uno de los datos.
- La **row-key** es el equivalente a la clave principal del modelo E/R y está formada por dos partes:
 - **Partition key:** objetivo es identificar en la partición o el nodo en que se encuentra esa fila/dato.
 - **Clustering key:** determina el orden físico en el que se almacenan las filas.

Base de Datos Cassandra

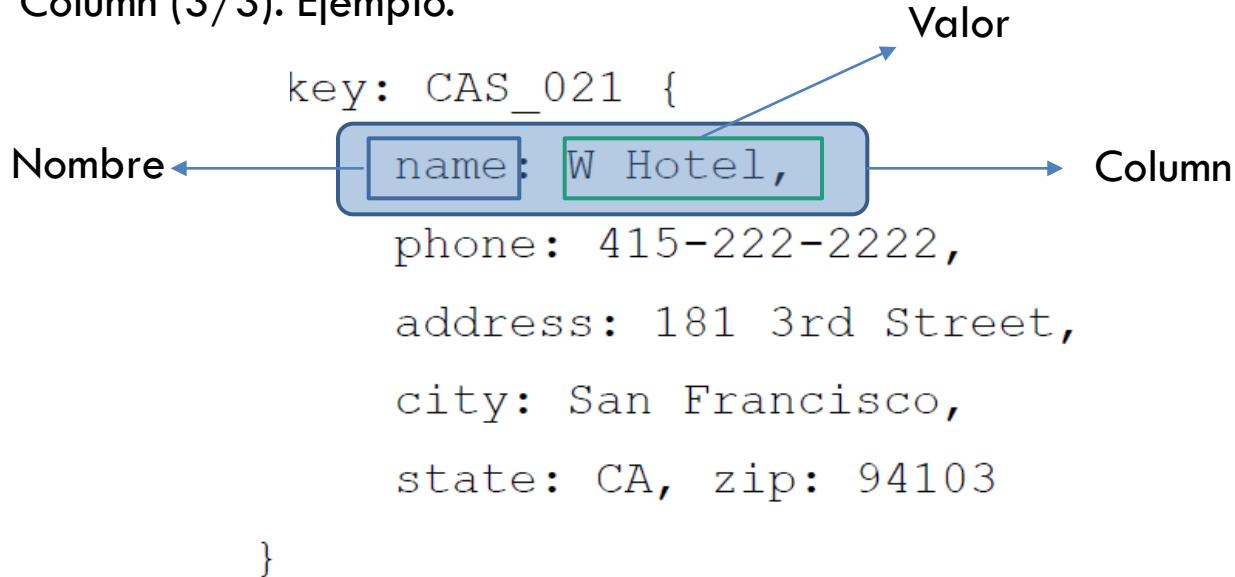


<http://cassandra.apache.org/>

➤ Introducción (21/36)

➤ Modelo de datos Cassandra (9/19)

➤ Column (3/3). Ejemplo.



- Introducción (22/36)

- Modelo de datos Cassandra (10/19)

- Column Family (1/4)

- Análogo a una tabla en un modelo relacional.
- Pero ... ¿son exactamente iguales las tablas de BBDD relacionales y las Column Family en Cassandra? No, por tres razones:
 - Aunque las Column Family son definidas, las columnas no, ya que podemos añadir columnas en cualquier momento.
 - La Column Family tiene dos atributos: nombre y comparador. El atributo comparador muestra la forma en cómo se mostrarán los datos en una consulta.
 - En el modelo relacional, las tablas están formadas por columnas y registros; en una Column Family, existen columnas relacionadas con las supercolumn.

Base de Datos Cassandra



➤ Introducción (23/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra (11/19)

➤ Column Family (2/4)

- A la hora de escribir en una Column Family, se dan valores para una o más columnas.
- Al conjunto de esos valores, con una clave que los identifica, es lo que se llama fila. La Column Family puede tener un número “infinito” de filas.
- En Cassandra, la estructura de datos se llama a menudo hash cuatridimensional:

[Keyspace] [Column Family][Key][Column]

Base de Datos Cassandra



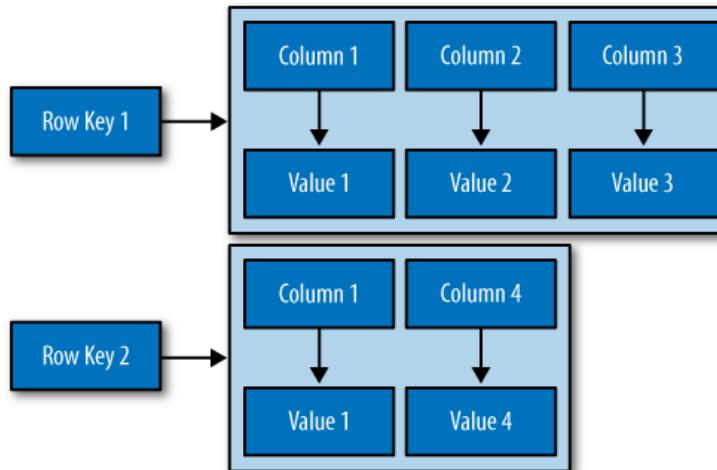
<http://cassandra.apache.org/>

➤ Introducción (24/36)

➤ Modelo de datos Cassandra (12/19)

➤ Column Family (3/4)

- Cada Column Family (familia de columnas) se almacena en un fichero distinto y consta de un nombre y un mapa con una clave/valor.



Base de Datos Cassandra



➤ Introducción (25/36)

➤ Modelo de datos Cassandra (13/19)

➤ Column Family (4/4). Ejemplo.

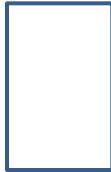
Column Family que se llama **Hotel** que guarda hoteles así como información relacionada con el mismo.



Nombre de la Column Family



Clave primaria



Columnas de la Column Family

<http://cassandra.apache.org/>

```
Hotel {  
    key: AZC_043 {  
        name: Cambria Suites Hayden, phone: 480-444-4444,  
        address: 400 N. Hayden Rd.,  
        city: Scottsdale,  
        state: AZ, zip: 85255  
    }key: AZS_011 {  
        name: Clarion Scottsdale Peak,  
        phone: 480-333-3333,  
        address: 3000 N. Scottsdale Rd,  
        city: Scottsdale,  
        state: AZ, zip: 85255  
    }key: CAS_021 {  
        name: W Hotel,  
        phone: 415-222-2222,  
        address: 181 3rd Street,  
        city: San Francisco,  
        state: CA, zip: 94103  
    }  
}
```

- Introducción (26/36)

- Modelo de datos Cassandra (14/19)

- Super Column (1/3)

- Son un tipo especial de columna:
 - Es una tupla con un nombre y un valor.
 - A diferencia de las columnas, no tienen un timestamp definido y una columna almacena el valor para esa columna y una super column contiene un mapa de columnas.
- No aconsejan usarla puesto que afecta al rendimiento en la búsqueda de información.
- La clave tiene el mismo valor que el nombre de la columna a la que se refiere.
- Están formados por un número determinado de familias de columnas.

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Introducción (27/36)

➤ Modelo de datos Cassandra (15/19)

➤ Super Column (2/3)

- Cassandra pasa de ser un hash de 4 dimensiones a serlo de 5:
[Keyspace] [Column Family][Key][Super Column][Sub Column]
- Para definir una super column generamos una column family como si fuera tipo super.
- Para acceder a una Super Column, se puede acceder de la misma forma que para acceder a una Column Family, simplemente habría que indicar el nombre de la Super Column.
- No podemos crear índices de los datos almacenados en las Super Column; cada vez que cargamos en memoria una Super Column, se cargarán todas las columnas asociadas.

Base de Datos Cassandra

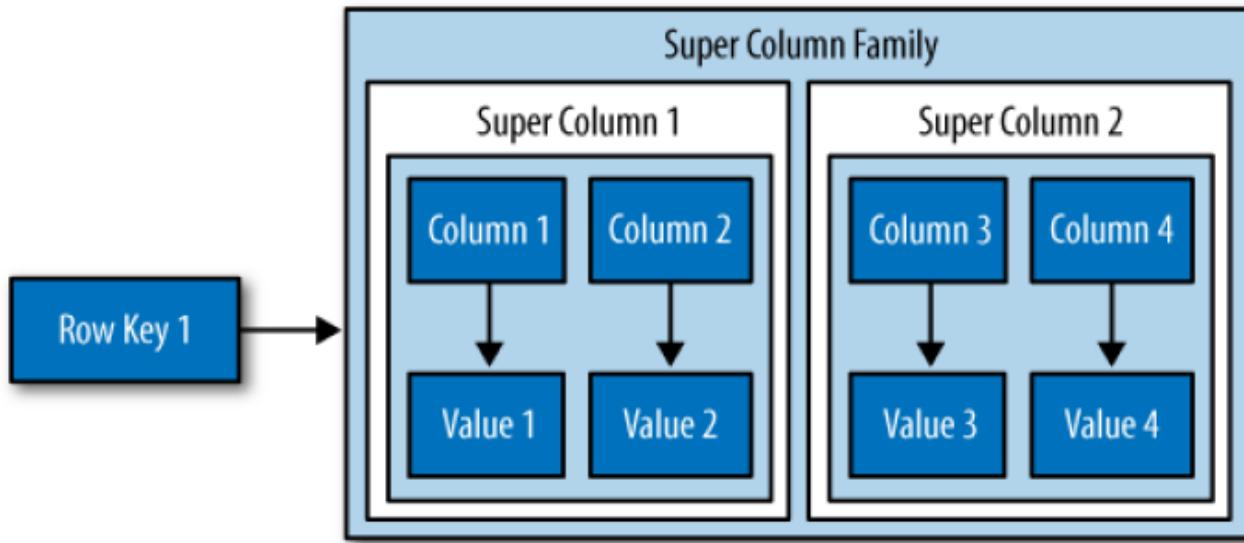


➤ Introducción (28/36)

➤ Modelo de datos Cassandra (16/19)

➤ Super Column (3/3)

<http://cassandra.apache.org/>



➤ Introducción (29/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra (17/19)

➤ Keyspace (1/2)

- Un clúster está formado por keyspaces (aunque normalmente clúster = keyspace).
- Similar a un contenedor de tablas en el modelo relacional (esquema de una BBDD). Es el contenedor de datos más externo de Cassandra, que guarda ordenadas column family y super column family.
- Usado en Cassandra para separar aplicaciones.
- Cada keyspace contiene un nombre y propiedades, que definen su comportamiento.

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Introducción (30/36)

➤ Modelo de datos Cassandra (18/19)

➤ Keyspace (2/2)

➤ Las propiedades que se pueden configurar son:

- *Factor de replicación*: número de copias que debe almacenarse de cada dato. A mayor factor de replicación, impacta en la consistencia, disponibilidad y rendimiento.
- *Estrategia de localización de réplicas*: establece qué política se aplicará para guardar las réplicas en el anillo y para indicar qué nodos serán los que alojen qué copias. Tiene en cuenta aspectos como: dónde se encuentran los servidores, los CPDs, la topología de la red, etc.

➤ Introducción (31/36)

➤ Modelo de datos Cassandra (19/19)

➤ Clúster:

- Conjunto de máquinas que dan soporte a Cassandra y son vistas por los clientes como una única máquina.
- Es la estructura más externa de Cassandra.
- Si se necesita ejecutar un único nodo, Cassandra no es la mejor opción.
- Cassandra está diseñado para distribuirse en múltiples máquinas, trabajando conjuntamente y el usuario final lo ve como una entidad única.
- El protocolo P2P hace que los nodos se comuniquen entre ellos para conocer su estado y repliquen los datos de forma transparente al usuario.

Base de Datos Cassandra



➤ Introducción (32/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra vs Modelo en RDBMS (1/5)

➤ Al modelar una BBDD relacional ... ¿cómo lo hacemos?

➤ Empezamos por el dominio conceptual:

- Representar los sustantivos en el dominio por tablas.
- Usamos claves primarias y foráneas para modelar relaciones.
- Una vez definidas las tablas, podemos comenzar a implementar instrucciones para leer y escribir datos.

➤ En Cassandra se suele empezar diseñando las consultas.

➤ Veamos un ejemplo.

➤ Introducción (33/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra vs Modelo en RDBMS (2/5)

➤ Ejemplo:

- Objetivo: un hotel desea registrar huéspedes en un libro de reservas.
- Nuestro dominio conceptual tiene las siguientes características:
 - Hoteles.
 - Huéspedes que se alojan en los hoteles.
 - Cada hotel tiene un número de habitaciones y un libro de reservas.
 - Cada hotel tiene también una lista de puntos de interés cercanos: museos, parques, tiendas de compras, monumentos, etc.
 - Se deben mantener la localización geográfica de los hoteles y de los puntos de interés, para que puedan ser localizados en los mapas y poder calcular las distancias.

Base de Datos Cassandra



➤ Introducción (34/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra vs Modelo en RDBMS (3/5)

➤ Consultas que deberíamos modelar:

- Buscar hoteles en una zona solicitada.
- Buscar información sobre un hotel específico como nombre y localización.
- Listar los puntos de interés que estén cerca de un hotel.
- Buscar una habitación libre para una fecha dada.
- Listar los precios y características de las habitaciones.
- Reservar una habitación, añadiendo los datos del huésped.

Base de Datos Cassandra

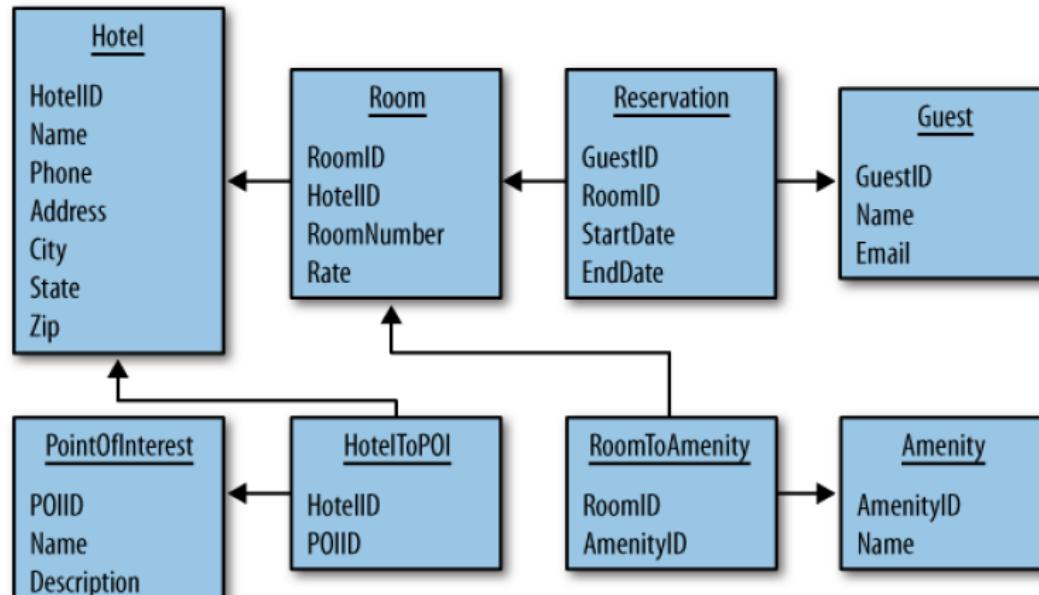


➤ Introducción (35/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra vs Modelo en RDBMS (4/5)

➤ Modelo E/R



Base de Datos Cassandra

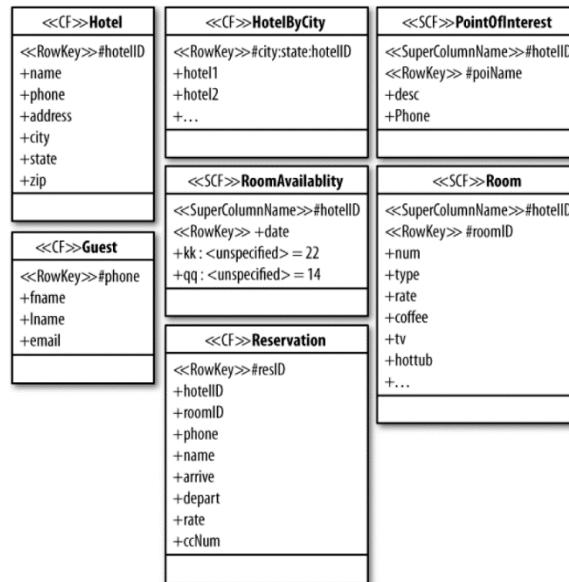


➤ Introducción (36/36)

<http://cassandra.apache.org/>

➤ Modelo de datos Cassandra vs Modelo en RDBMS (5/5)

➤ Modelo en Cassandra



Almacenamiento de registros

Base de Datos Cassandra



- Almacenamiento de registros (1/3) <http://cassandra.apache.org/>
 - Una BBDD está representada por una keyspace, que almacenará las tablas.
 - Para crear una tabla es necesario crear:
 - Clave de partición: su función es definir el nodo donde se almacenará el registro. Para ello, utiliza la función hash.
 - Clave de clúster: una vez que se ha definido el nodo donde se guardará el registro, el paso siguiente es guardarla teniendo en cuenta el orden definido por el usuario. Suele ser una marca de tiempo (timestamp).
 - Veamos un ejemplo donde se ve más claramente esto.

Base de Datos Cassandra



➤ Almacenamiento de registros (2/3)

<http://cassandra.apache.org/>

➤ Ejemplo (1/2)

- Supongamos un clúster de N nodos utilizado para almacenar 1 TB de información cada día. Nuestra aplicación es Twitter y cada tweet contendrá: id del usuario, dispositivo, la marca de tiempo y el texto.

```
CREATE TABLE tweet (
    id_usuario bigint,
    dispositivo text,
    timestamp timestamp,
    tweet text,
    PRIMARY KEY (id_usuario, timestamp)
) WITH CLUSTERING ORDER BY (timestamp DESC);
```

- La **clave de partición** será el `id_usuario`, es decir, cuando un nodo reciba un nuevo registro a almacenar, aplicará la siguiente función $\text{hash}(\text{id_usuario}) = \{1, 2, 3, \dots, N\}$, es decir, los posibles valores serán del 1 al N.
- La **clave de clúster** es la que se encarga de ordenar los registros en el nodo (el campo `timestamp` de manera descendente).

Base de Datos Cassandra

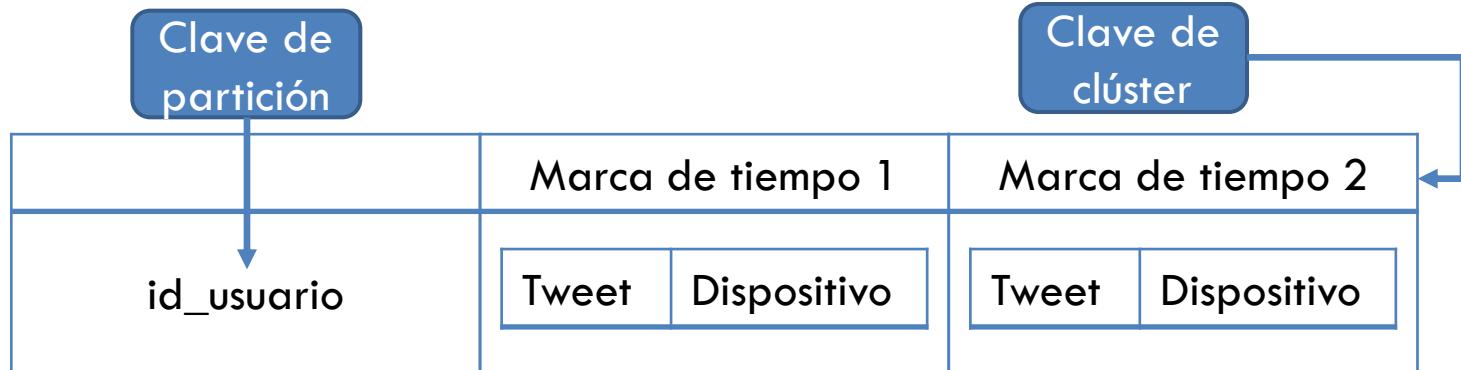


➤ Almacenamiento de registros (3/3)

<http://cassandra.apache.org/>

➤ Ejemplo (2/2)

- *Clave de clúster:* cada nodo Cassandra almacenará los registros pertenecientes a diferentes claves de partición (es decir, en nuestro ejemplo, tenemos N nodos, pero podemos tener miles y miles de usuarios). Por tanto, Cassandra define el almacenamiento por clave partición en forma horizontal.





Casos de uso

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Casos de uso (1/4)

➤ Netflix

- Minorista de películas online en DVD y Blu-ray.
- Un estudio de Nielsen⁽¹⁾ mostró que el 38% de los americanos usan o se suscriben a Netflix.
- ¿Por qué Cassandra?
 - El uso de una BBDD SQL centralizada impactó en la escalabilidad y disponibilidad.
 - La expansión internacional requiere una solución de múltiples centros de datos.
 - Necesidad de una replicación configurable, consistencia y resistencia en caso de fallo.
 - Cassandra en Amazon Web Service ofreció altos niveles de escalabilidad y disponibilidad.



(1) <https://eu.usatoday.com/story/life/tv/2013/09/18/netflix-hulu-amazon-nielsen-viewership-data/2831535/>

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Casos de uso (2/4)

➤ Spotify (1/2)

- Servicio de streaming de música digital.
- Música en cualquier momento: ordenador, móvil, Tablet, etc.
- ¿Por qué Cassandra?
 - Debido a muchos usuarios, los problemas de escalabilidad aparecieron usando PostgreSQL.
 - Con múltiples centros de datos, la replicación en PostgreSQL era problemática (ejemplo, gran volumen de escrituras).
 - Gracias a Cassandra:
 - Desaparecen los puntos de fallo únicos.
 - No hay pérdida de datos (debido a la replicación en diferentes nodos).



Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Casos de uso (3/4)

➤ Spotify (2/2)

➤ ¿Por qué Cassandra?

➤ Cassandra se comporta mejor en ciertos casos específicos:

- Mejor replicación (especialmente en la escritura).
- Mejor comportamiento tras la aparición de problemas y fallos de red, tales como particiones e interferencias intermitentes.
- Mejor comportamiento en ciertas clases de fallo, tales como el servidor “muere” y los enlaces de red caen.



Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Casos de uso (4/4)

➤ Hulu

- Página web y servicio de suscripción que ofrece contenidos de vídeo bajo demanda.
- Alrededor de 30 millones de usuarios diferentes al mes.
- ¿Por qué Cassandra?
 - Necesidad de disponibilidad y escalabilidad.
 - Buen rendimiento.
 - Escalabilidad lineal.
 - Replicación geolocalizada.
 - Requisitos de mantenimiento mínimos.





Características

Base de Datos Cassandra



➤ Características (1/10)

<http://cassandra.apache.org/>

- Objetivo: manejar grandes cantidades de datos a través de múltiples nodos sin ningún punto único de fallo.
- Tiene sistema distribuido de igual a igual a través de sus nodos, distribuyéndose los datos entre todos los nodos de un clúster.
 - Todos los nodos de un clúster juegan el mismo papel.
 - Cada nodo es independiente y está interconectado a otros nodos.
 - Cada nodo de un clúster puede aceptar solicitudes de L/E, independientemente de donde se encuentre realmente el clúster.
 - Cuando un nodo se cae, las peticiones de L/E se realizan contra otros nodos de la red.

➤ Características (2/10)

➤ Particionado (1/2)

- Elegir cómo dividir los datos en los nodos del sistema.
- Toda la información manejada por el clúster se representa como un anillo.
- Dicho anillo se divide en tantos rangos como números de nodos haya.
- Cada nodo es responsable de uno o más rangos de los datos globales.
- Antes de unir un nuevo nodo al anillo, hay que asignarle un token.
 - Este token determina la posición que ocupa en el anillo y el rango de datos del que será responsable.
- La aplicación especifica la clave y Cassandra la utiliza para enrutar las peticiones.

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Características (3/10)

➤ Particionado (2/2)

- Cada nodo es responsable de la región en el anillo entre él y su nodo predecesor.
- Permite que la “llegada” o “salida” de un nodo solo afecta a sus vecinos inmediatos y no a todos los nodos.

➤ Características (4/10)

➤ Replicación (1/2)

- Objetivo: conseguir una alta disponibilidad y durabilidad.
- Cada elemento de datos se replica en N nodos.
 - N es el factor de replicación configurado.
- El nodo coordinador se encarga de la replicación de los elementos de datos que caen dentro de su rango.
- Además, el nodo coordinador replica estas claves en los N-1 nodos del anillo.

<http://cassandra.apache.org/>

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Características (5/10)

➤ Replicación (2/2)

➤ Tres tipos diferentes de políticas de replicación:

- **Rack Unaware:** se escogen N-1 sucesores del nodo coordinador de esos datos, para almacenar las réplicas de los datos.
- **Rack Aware:** usa Zookeeper para elegir un líder entre sus nodos, para indicarles cuál es su rango para las réplicas.
- **Datacenter Aware:** usa también Zookeeper para elegir un líder entre sus nodos, para indicarles cuál es su rango para las réplicas, pero dicho líder es elegido a nivel de centro de datos en lugar de a nivel de rack. Además el líder procura que ningún nodo sea responsable de más de N-1 rangos del anillo.

Base de Datos Cassandra



➤ Características (6/10)

<http://cassandra.apache.org/>

- Añadir miembros y detección de fallos (1/4)
 - Cassandra usa el protocolo Gossip para conocer la información sobre el estado y localización de otros nodos.
 - Gossip: protocolo de comunicación P2P que permite a los nodos intercambiar información periódicamente sobre sí mismos y sobre otros nodos.
 - Cuando un nuevo nodo se une al clúster la primera vez, busca en su fichero de configuración a qué clúster pertenece y a qué nodos (seeds) para comenzar a trabajar.
 - Esta información se almacena en el fichero de configuración *Cassandra.yaml*.

Base de Datos Cassandra



➤ Características (7/10)

<http://cassandra.apache.org/>

➤ Añadir miembros y detección de fallos (2/4)

➤ Para evitar cortes de comunicación en el protocolo gossip:

- Todos los nodos deben tener la misma lista de nodos en su fichero de configuración.
- Este hecho, es más crítico durante el arranque, por la primera vez que se ejecuta.

Base de Datos Cassandra



➤ Características (8/10)

<http://cassandra.apache.org/>

➤ Añadir miembros y detección de fallos (3/4)

➤ Detector de Fallos Acumulativo:

- El detector de fallos devolverá un valor que representa el nivel de sospecha de cada nodo monitorizado.

➤ Respecto a la detección de fallos, Cassandra utiliza la misma para evitar intentos de comunicación con nodos que son inalcanzables durante varias operaciones.

➤ Motivos que provocan fallos en los nodos: fallos de discos, comportamiento erróneo de CPU, fallo en el suministro eléctrico, fallo de red, etc.

➤ Características (9/10)

<http://cassandra.apache.org/>

- Añadir miembros y detección de fallos (4/4)
 - Rara vez una caída de un nodo representa una salida definitiva del clúster.
 - No es necesario realizar un rebalanceo.
 - No es necesario realizar la reparación de réplicas inalcanzables.
 - Podría ocurrir la puesta en marcha involuntaria de nuevos nodos en Cassandra.
 - Cada mensaje contiene el nombre del clúster de cada instancia de Cassandra.
 - Dicho mensaje evita que un nodo intente unirse a una instancia incorrecta de Cassandra.

➤ Características (10/10)

<http://cassandra.apache.org/>

➤ Arranque de nodos

➤ Cuando un nodo arranca por primera vez:

- Escoge un token que es equivalente a su posición al anillo.
- La información del token es transmitida por todo el clúster, para que todos los nodos sepan su posición y la del resto de nodos del anillo.
- Lee su fichero de configuración, que contiene una lista con una serie de puntos de acceso (denominados semillas del clúster) para unirse al clúster.

Rendimiento

Base de Datos Cassandra



➤ Rendimiento (1 / 4)

➤ Comparación con MySQL⁽¹⁾

<http://cassandra.apache.org/>

	Datos	Media Escrituras	Media Lecturas
MySQL	>50 GB	~300 ms	~350 ms
Cassandra	>50 GB	0,12 ms	15 ms

(1) Datos obtenidos usando datos de Facebook.

➤ Rendimiento (2/4)

<http://cassandra.apache.org/>

- Comparación usando YCSB⁽¹⁾(Yahoo Cloud Serving Benchmark) (1/3)
 - Estos resultados se han obtenido gracias al “Benchmarking Cloud Serving Systems with YCSB” by Brian F. Cooper et all.
 - Es un framework que permite realizar pruebas de rendimiento sobre BBDD.
 - Dicha análisis de rendimiento se ha realizado sobre Cassandra, HBase, PNUTS⁽²⁾ y MYSQL.
 - Veamos algunas conclusiones obtenidas a través de dicho framework.

(1) Paper: <http://www.rte.espol.edu.ec/index.php/tecnologica/article/view/445>

(2) Paper: <https://people.mpi-sws.org/~druschel/courses/ds/papers/cooper-pnuts.pdf>

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Rendimiento (3/4)

➤ Comparación usando YCSB (2/3)

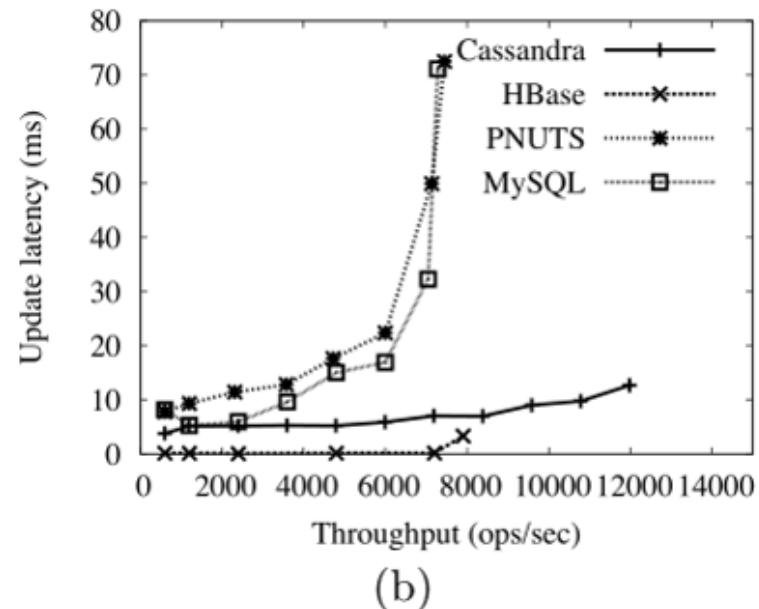
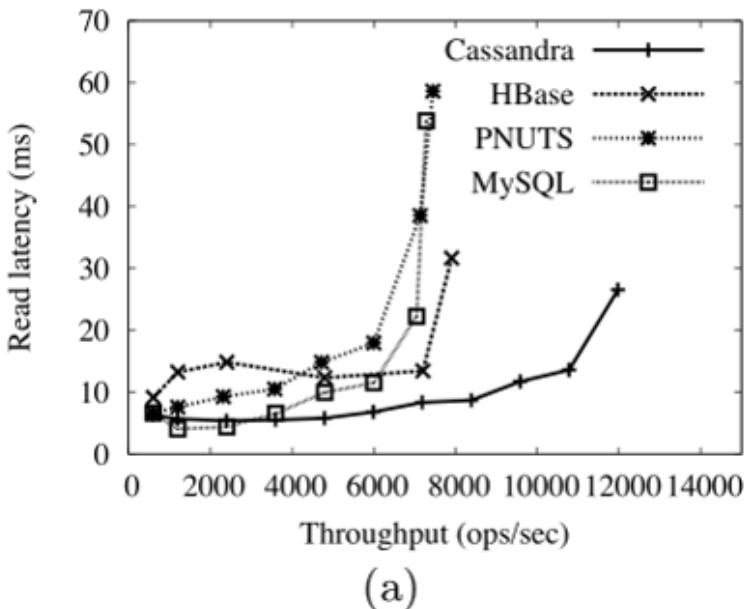
- Cassandra y Hbase tienen un tiempo de latencia de lectura más grande, estando sobrecargada la BBDD, que PNUTS y MySQL.
- Cassandra y Hbase tienen un tiempo de latencia de escritura más pequeño, estando sobrecargada la BBDD, que PNUTS y MySQL.
- PNUTS y Cassandra escalan bien mientras el número de servidores y la sobrecarga se vayan incrementando proporcionalmente.
- Cassandra, Hbase y PNUTS son capaces de crecer elásticamente mientras la sobrecarga se está ejecutando.

Base de Datos Cassandra



➤ Rendimiento (4/4)

➤ Comparación usando YCSB (3/3)





Cassandra vs SGBD

Base de Datos Cassandra



➤ Cassandra vs SGBD

<http://cassandra.apache.org/>

	Cassandra	SGBD
Modelo de datos	<ul style="list-style-type: none">- Diseñado para consultas específicas.- El esquema es ajustado a las nuevas consultas.	<ul style="list-style-type: none">- Normalizado, sin tener en cuenta las consultas.- SQL puede “devolver casi todo” gracias a JOINS.
Características	<ul style="list-style-type: none">- Sin JOINS, relaciones ni claves ajenas.- Una tabla separada por consulta.	



Cassandra Query Language

Base de Datos Cassandra



➤ Cassandra Query Language (CQL) (1/4)

- Lenguaje de alto nivel.
- Prácticamente igual al lenguaje SQL.
- CQL vs SQL: no permite realizar consultas con JOINs ni subconsultas.
- Consola de línea de comandos realiza en Python.

```
<cassandra-path>bin/cqlsh
```

<http://cassandra.apache.org/doc/cql3/CQL-3.0.html>

Base de Datos Cassandra



➤ Cassandra Query Language (CQL) (2/4)

- **CREATE KEYSPACE:** crea una BBDD.

```
CREATE KEYSPACE demo WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
```

- **SimpleStrategy** (valor por defecto): asigna el mismo factor de replicación a todo el clúster. Es únicamente usada para la evaluación en entornos de desarrollo y pruebas sobre un único data center.
- **NetworkTopologyStrategy**: asigna factores de replicación a cada data center separados por comas. Usado para entornos de producción.
- **Durable_writes** (opcional y valor por defecto **true**): al recibir una petición de escritura, dicha operación es escrita en el log.
- **DESCRIBE KEYSPACE:** muestra todos los keyspaces definidos.

```
DESCRIBE KEYSPACE demo;
```

<http://cassandra.apache.org/doc/cql3/CQL-3.0.html>

Base de Datos Cassandra



➤ Cassandra Query Language (CQL) (3/4)

- **USE:** para empezar a trabajar sobre un keyspace.

```
USE demo;
```

- **CREATE TABLE:** crear una tabla.

```
CREATE TABLE emp ( empID int, deptID int, first_name varchar, last_name varchar, PRIMARY KEY (empID, deptID) );
```

- **INSERT:** insertar elementos en una tabla.

```
INSERT into emp ( empID, deptID, first_name, last_name ) VALUES (104, 15, 'jane', 'smith');
```

```
INSERT into emp ( empID, deptID, first_name, last_name ) VALUES (105, 16, 'john', 'smith');
```

- **SELECT:** muestra las filas que cumplan la búsqueda.

```
SELECT * FROM emp;
```

<http://cassandra.apache.org/doc/cql3/CQL-3.0.html>

Base de Datos Cassandra



➤ Cassandra Query Language (CQL) (4/4)

- **ALTER TABLE:** agrega/elimina un campo dentro de la tabla.

```
ALTER TABLE emp ADD address text;
```

```
ALTER TABLE emp DROP address;
```

- **DROP TABLE:** permite eliminar una tabla.

```
DROP TABLE emp;
```

- **DELETE:** permite borrar una fila.

```
DELETE FROM emp = WHERE empID = 104;
```

- **UPDATE:** permite modificar el valor de una fila.

```
UPDATE emp SET address = 'Universidad Autónoma' WHERE empID = 105 and deptID = 16.
```

<http://cassandra.apache.org/doc/cql3/CQL-3.0.html>



Arquitectura

Base de Datos Cassandra

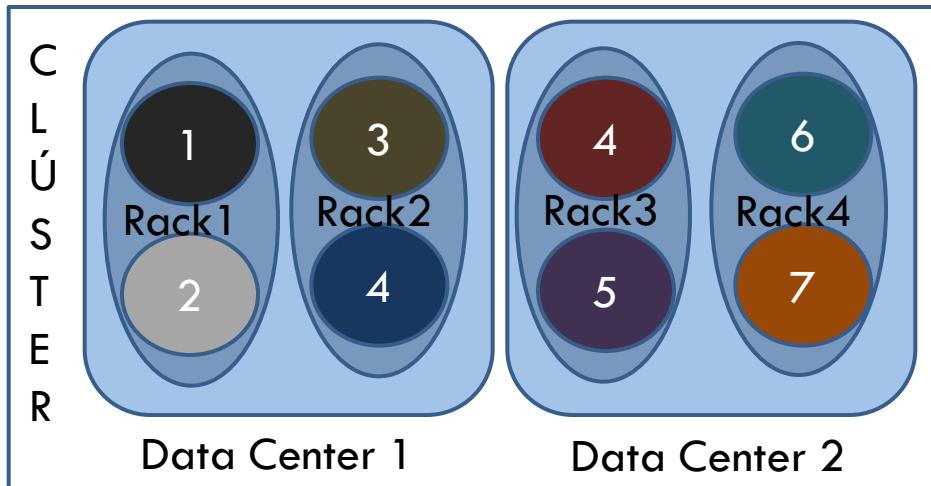


<http://cassandra.apache.org/>

➤ Arquitectura (1/6)

➤ Clúster

- Nodo: una instancia de Cassandra.
- Rack: un conjunto lógico de nodos.
- Data Center: un conjunto lógico de racks.
- Clúster: conjunto completo de nodos los cuales están mapeados a una topología de anillo.



➤ Arquitectura (2/6)

<http://cassandra.apache.org/>

➤ Coordinador

- Es el nodo elegido por el cliente para realizar operaciones de lectura/escritura dentro del clúster.
- Cualquier nodo puede coordinar cualquier petición.
- Cada solicitud de un mismo cliente puede ser gestionada por diferentes nodos.
- Gestiona el factor de replicación (RF): indica el número de nodos en los que debe haber una copia del dato.
- Aplica el nivel de consistencia (CL): decide el nivel de consistencia que se necesita en la lectura y escritura frente al nivel de disponibilidad.

➤ Arquitectura (3/6)

➤ Particionador (“Partitioner”) (1/2)

- Encargado de decidir cómo se distribuirán los datos.
- Cassandra ofrece tres tipos de particionado:
 - Murmur3Partitioner (por defecto): distribuye uniformemente los datos a lo largo del clúster basado en valores hash Murmurhash⁽¹⁾
 - RandomPartitioner: distribuye uniformemente los datos a lo largo del clúster basado en valores hash MD5.
 - ByteOrderedPartitioner: particionado de versiones antiguas de Cassandra, donde los datos se distribuyen léxicamente basado en bytes clave.

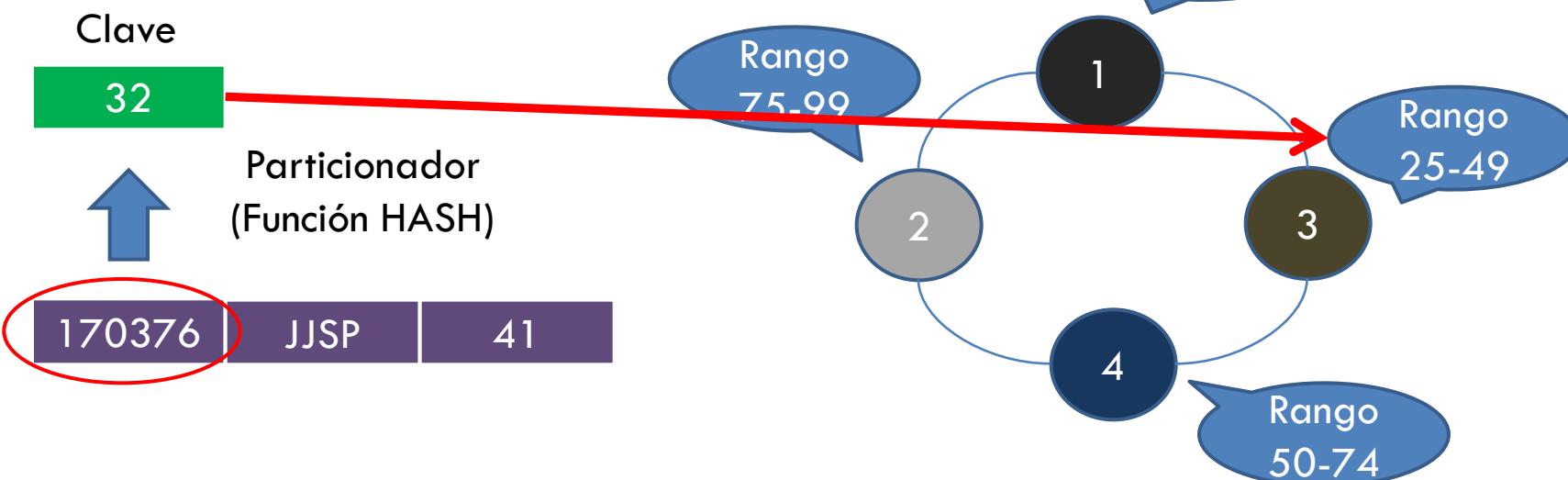
(1) <https://en.wikipedia.org/wiki/MurmurHash>

Base de Datos Cassandra



<http://cassandra.apache.org/>

- Arquitectura (4/6)
 - Particionador (“Partitioner”) (2/2)



- Arquitectura (5/6)

- Nodos virtuales (1/2)

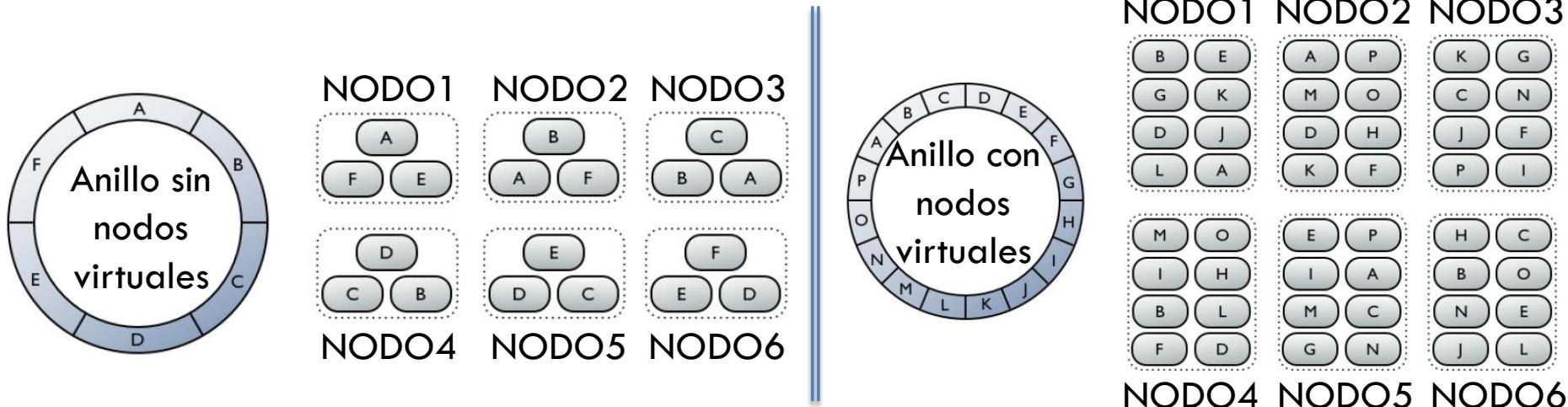
- Hay un token por nodo, por lo que cada nodo posee un rango contiguo de tokens dentro del anillo.
- Gracias a los nodos virtuales, nos permiten asignar varios rangos por nodo.
- Dentro de un clúster, pueden ser seleccionados al azar y no ser contiguos, obteniendo así muchos rangos más pequeños en cada nodo.

Base de Datos Cassandra



<http://cassandra.apache.org/>

- Arquitectura (6/6)
 - Nodos virtuales (2/2)



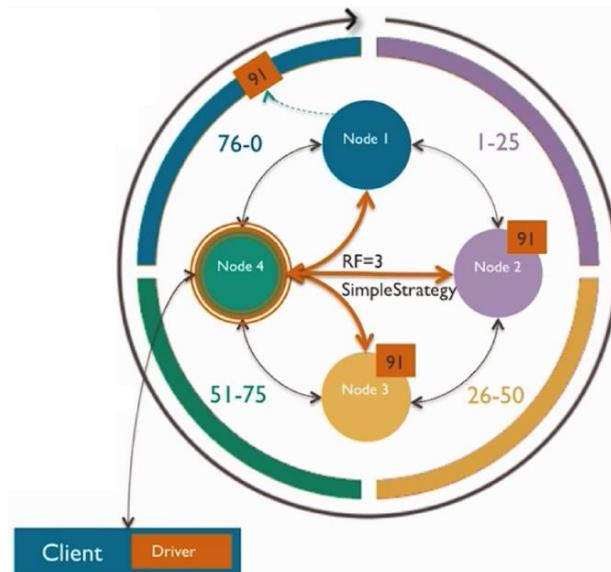
Replicación

Base de Datos Cassandra



<http://cassandra.apache.org/>

- Replicación (1/3)
 - Proceso de replicación (1/2)
 - Entre nodos (SimpleStrategy)



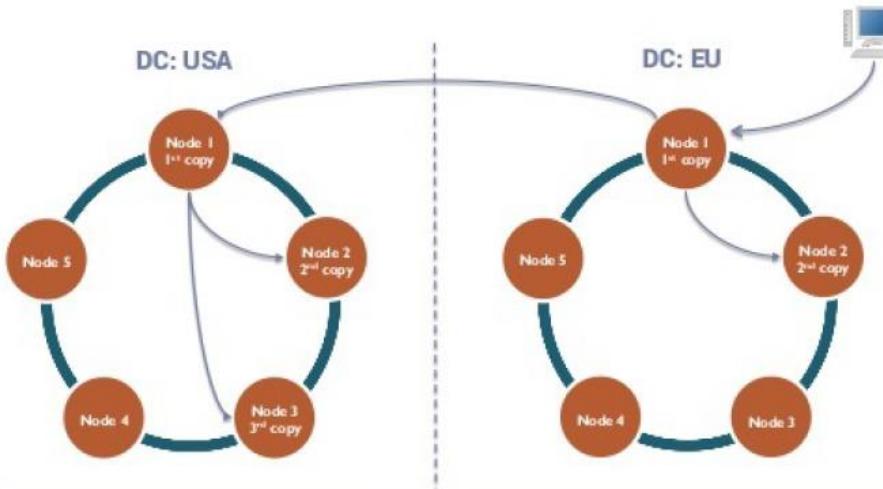
Base de Datos Cassandra



- Replicación (2/3)
 - Proceso de replicación (2/2)
 - Entre data centers

<http://cassandra.apache.org/>

```
CREATE KEYSPACE johnny WITH REPLICATION =  
{ 'class': 'NetworkTopologyStrategy', 'USA':3, 'EU': 2};
```



➤ Replicación (3/3)

➤ Hinted handoff

- Consiste en un mecanismo de recuperación para escribir en nodos que están offline.
- El nodo coordinador puede almacenar dicha información si el nodo sobre el que hay que escribir:
 - Se sabe que está caído o
 - Falla la confirmación de que hay comunicación.
- En estos casos, el coordinador guarda dicha información en la tabla **system.hints** y será escrita en el nodo cuando este esté recuperado.

Consistencia

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Consistencia (1/2)

➤ ¿Qué es?

- Permite definir cuántos nodos deben dar respuesta a una petición de un cliente.
 - Petición de escritura: ¿cuántos nodos deben responder que han recibido y escrito la petición de escritura?
 - Petición de lectura: ¿cuántos nodos deben responder con la información más reciente que tienen respecto a la petición de lectura?

➤ Tipos:

- El nivel de consistencia más bajo es 1, que indica que al leer o escribir basta con completar la operación en uno de los nodos.
- El nivel de máxima consistencia consiste la operación de lectura o escritura debe ser realizada en todos los nodos.

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Consistencia (2/2)

➤ Tipos (Cont.)

- Existe un modo intermedio, llamado quorum, donde se busca mayoría simple (la mitad más uno) para considerar una operación completada.

Conclusiones

Base de Datos Cassandra



<http://cassandra.apache.org/>

➤ Conclusiones (1/3)

➤ Ventajas

- Perfecta para datos de series de tiempo.
- Alto rendimiento.
- Descentralizada.
- Escalabilidad casi lineal.
- Soporta replicación.
- No hay puntos únicos de fallo.
- Soporte al paradigma MapReduce.

➤ Conclusiones (2/3)

➤ Debilidades

- No existe integridad referencial.
 - No existe el concepto de JOIN.
- Las opciones de consulta para recuperar datos son limitadas.
- La ordenación de datos es una decisión de diseño.
 - No existe la instrucción GROUP BY.
- No hay soporte para operaciones atómicas.
 - Si la operación falla, los cambios pueden producirse.
- “Primero se piensa sobre las consultas, después sobre el modelo de datos”.

Base de Datos Cassandra



➤ Conclusiones (3/3)

➤ Puntos a considerar

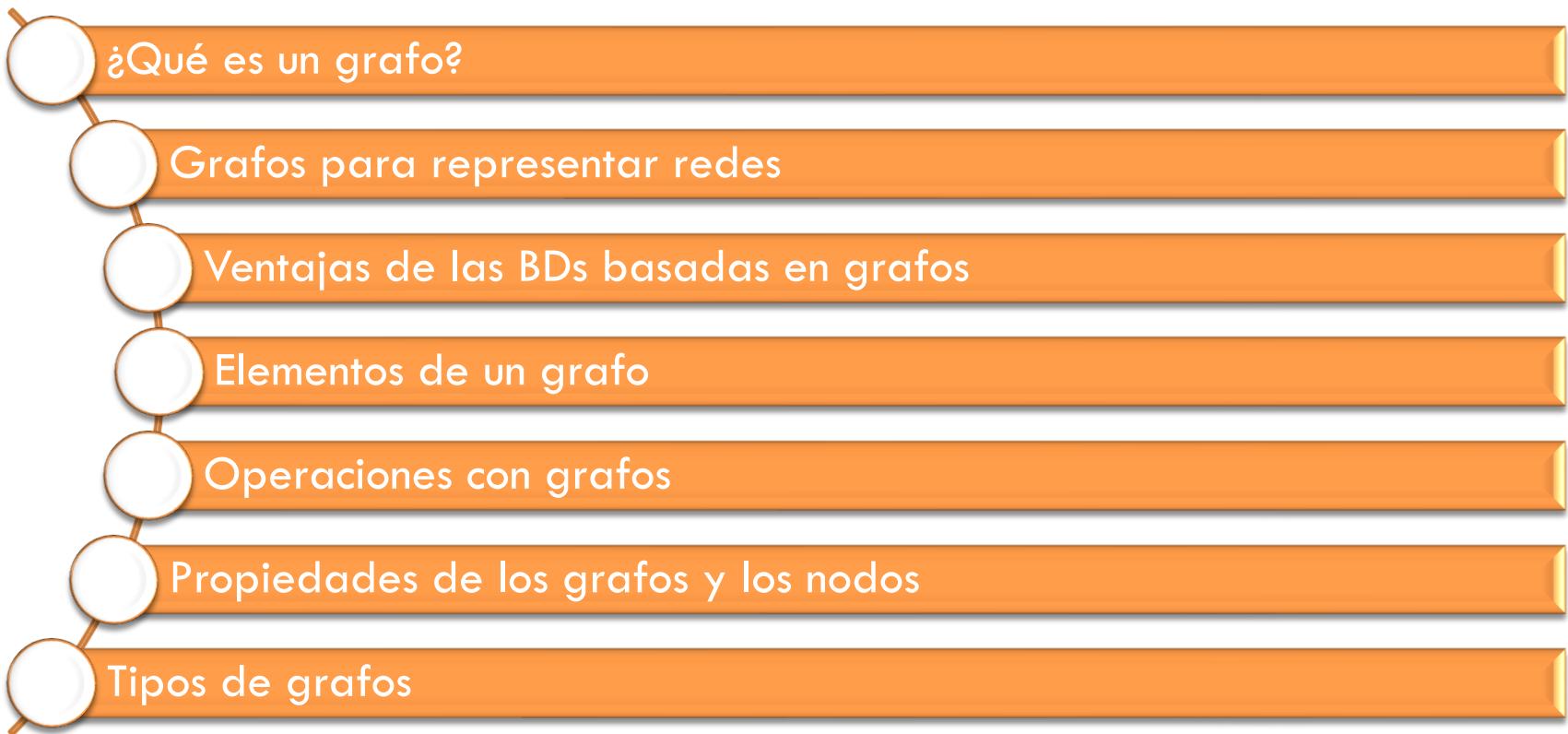
- Diseñada como una BBDD distribuida.
 - Es conveniente usarla cuando tenemos muchos datos distribuidos a través de múltiples servidores.
- El rendimiento de escritura siempre es excelente, pero el rendimiento en la lectura depende de los patrones de escritura.
 - Es importante gastar el suficiente tiempo para diseñar propiamente el esquema alrededor del patrón de consulta.

<http://cassandra.apache.org/>

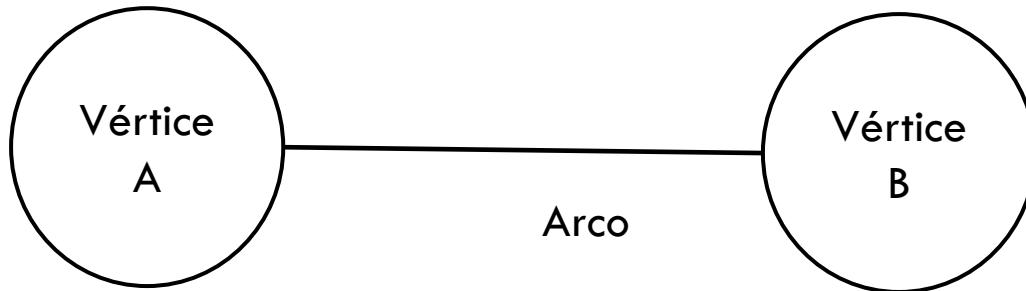
Bases de datos orientadas a grafos

TEMA 1. Análisis de grafos

Contenido

- 
- 1. ¿Qué es un grafo?
 - 2. Grafos para representar redes
 - 3. Ventajas de las BDs basadas en grafos
 - 4. Elementos de un grafo
 - 5. Operaciones con grafos
 - 6. Propiedades de los grafos y los nodos
 - 7. Tipos de grafos

¿Qué es un grafo?



Grafos para modelizar redes

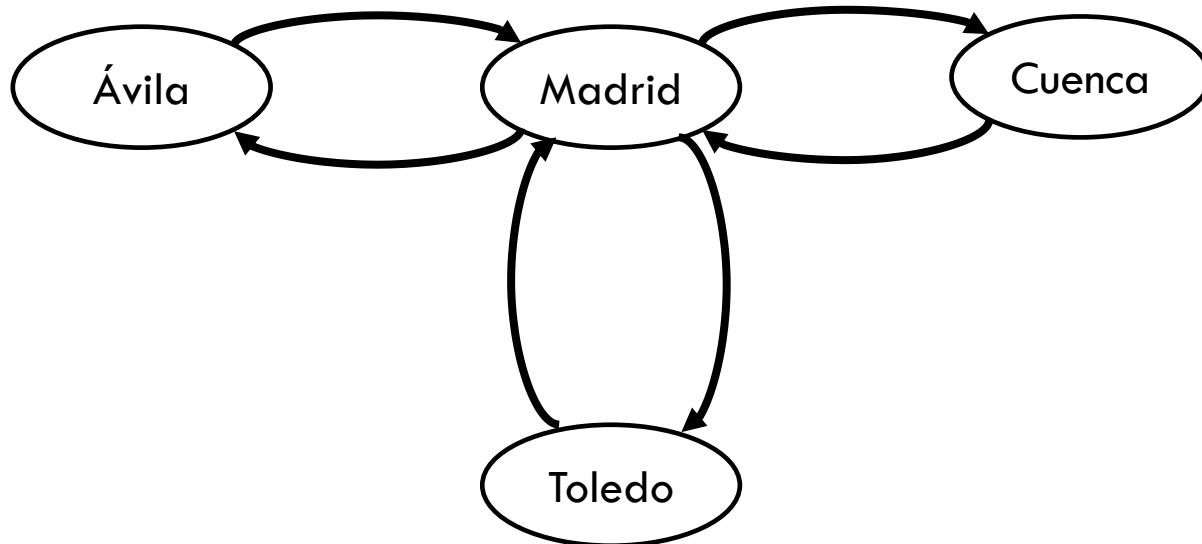
Ubicaciones geográficas

Enfermedades infecciosas

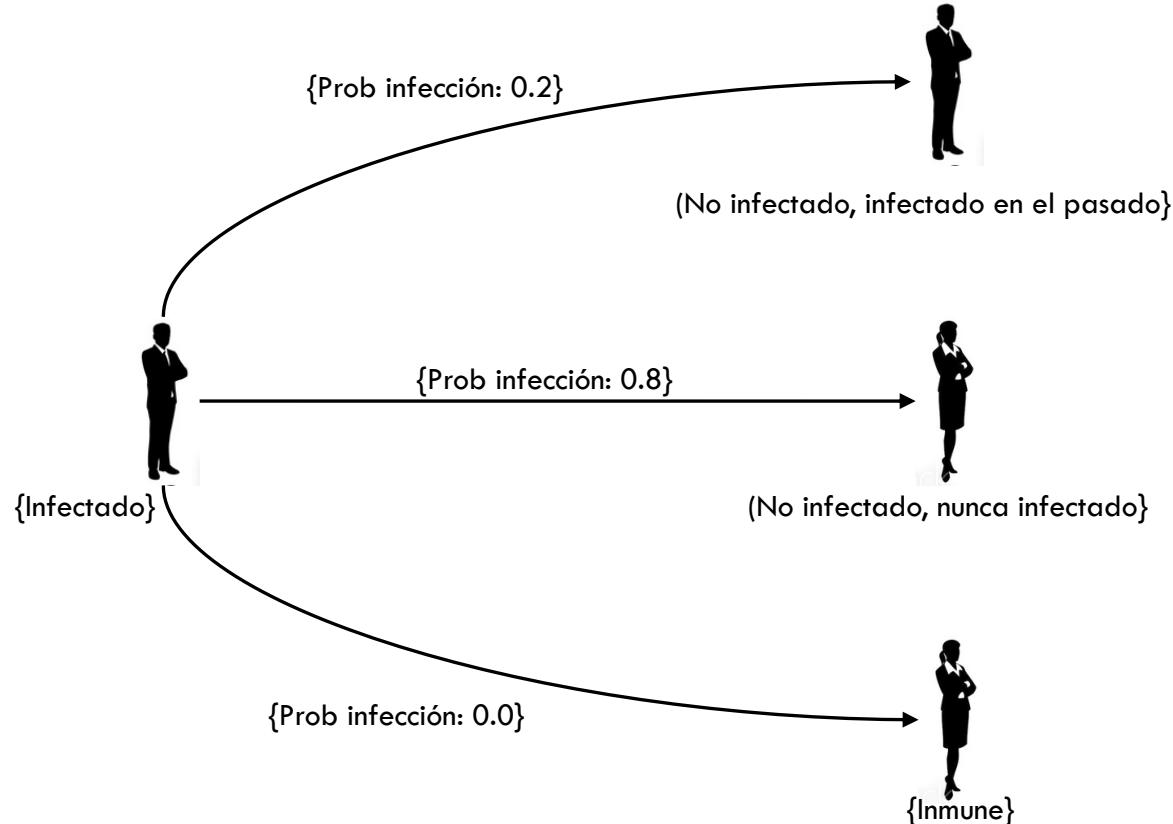
Entidades abstractas y concretas

Redes sociales

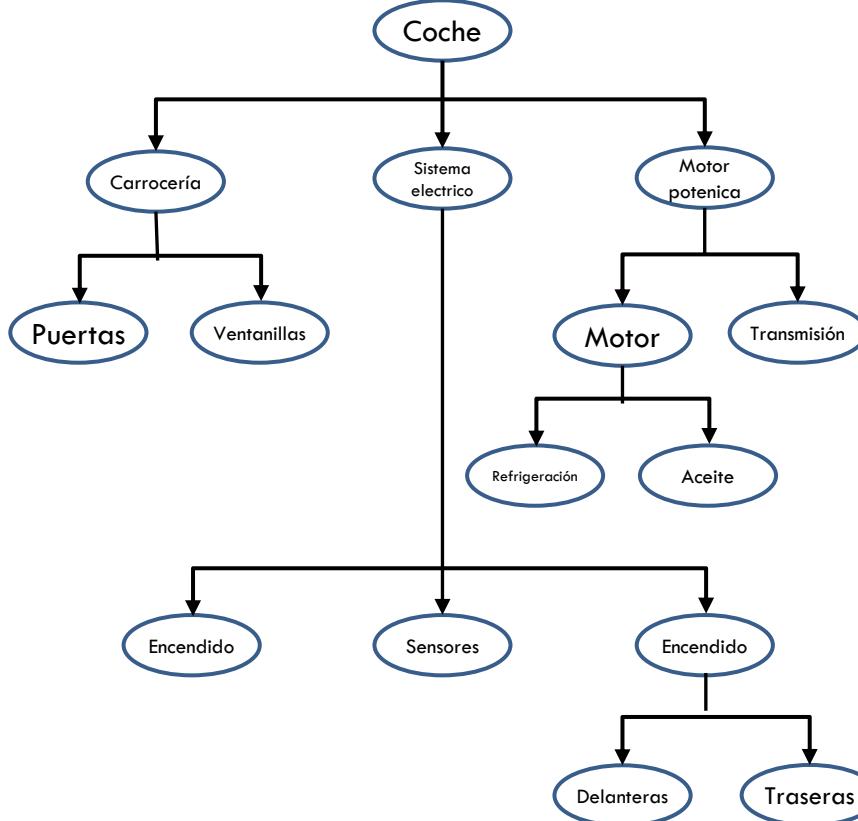
Ubicaciones geográficas



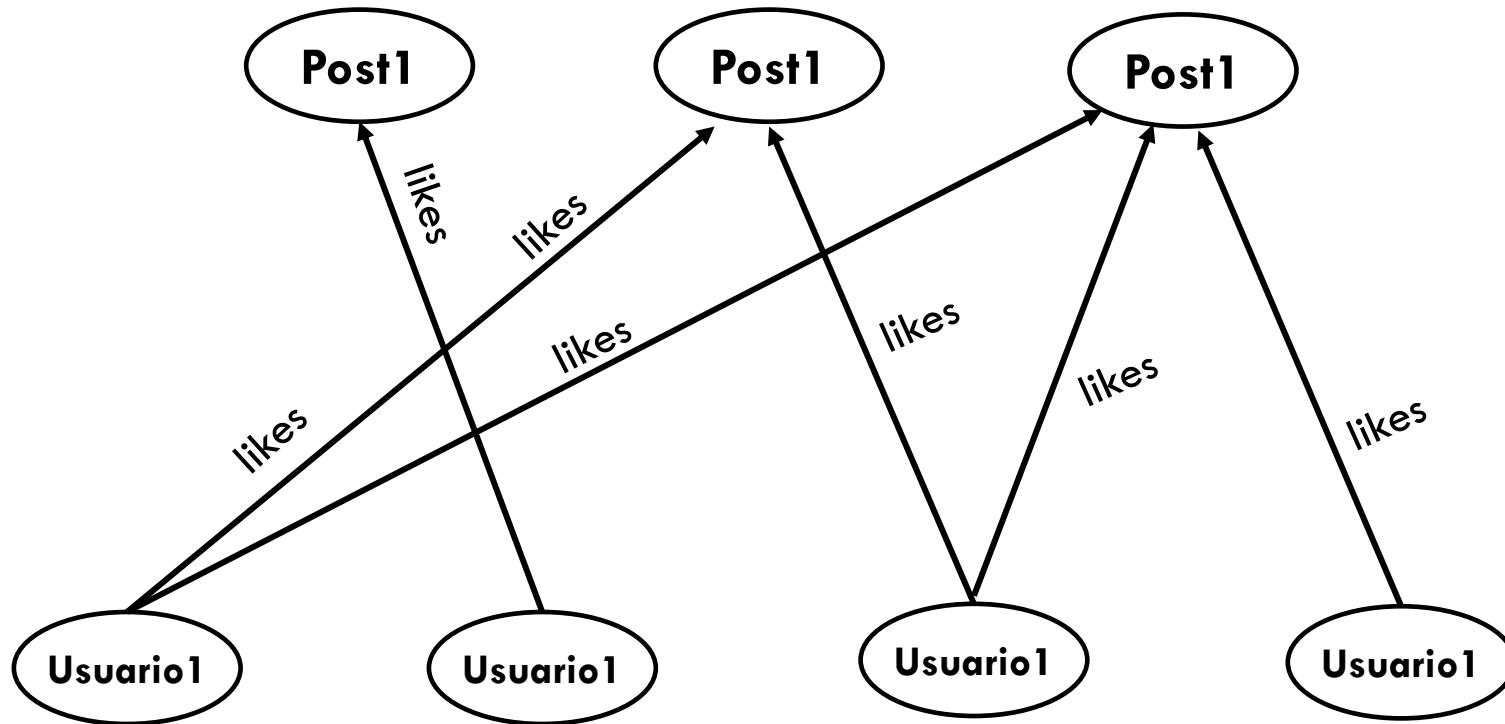
Enfermedades infecciosas



Entidades abstractas y concretas



Redes sociales



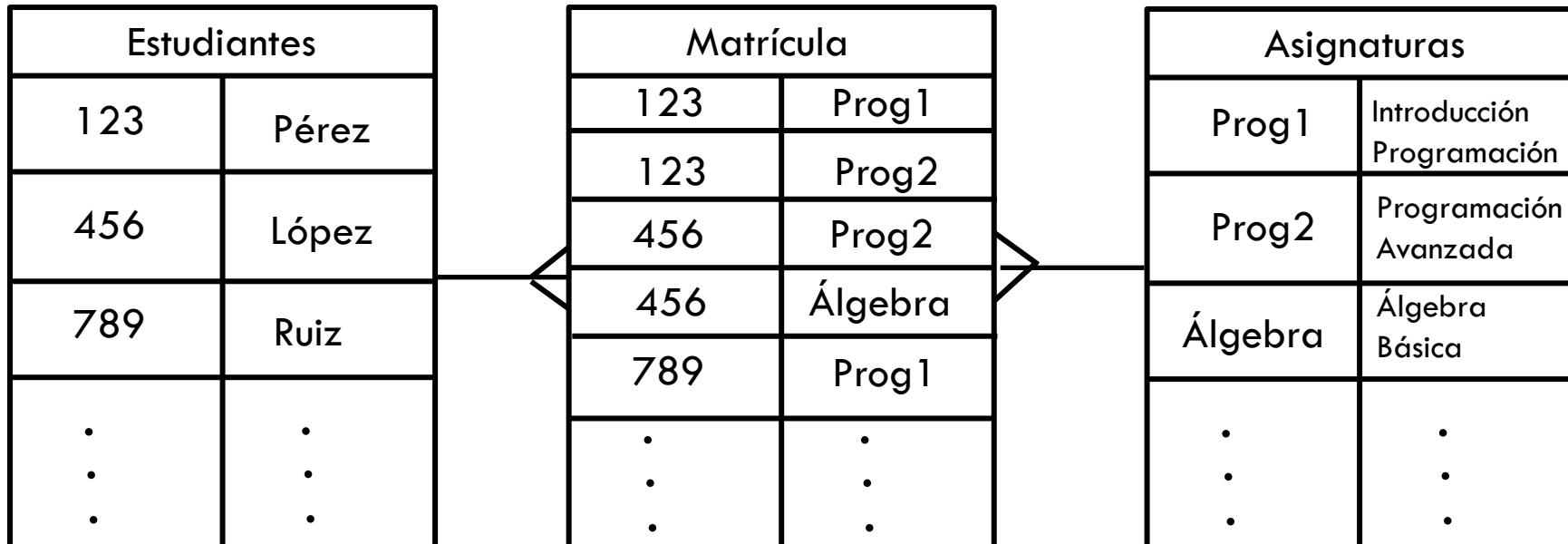
Ventajas de las BD basadas en grafos

Consultas más rápidas

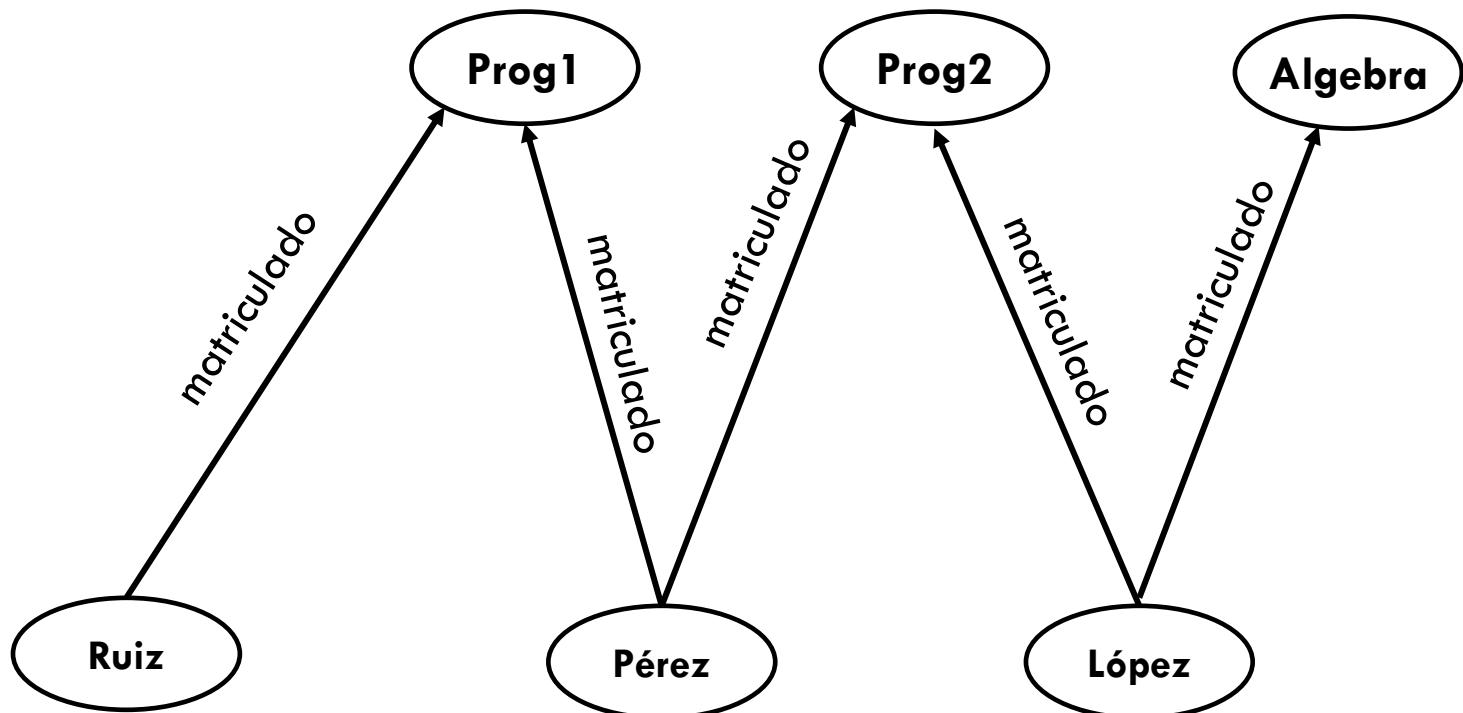
Modelado más sencillo

Múltiples relaciones entre entidades

Consultas más rápidas



Consultas más rápidas



Consultas más rápidas

Curso de la infección	
Paciente	Infectado Por
Paciente A	Paciente B
Paciente B	Paciente C
Paciente C	Paciente D
Paciente D	Paciente E
Paciente E	Paciente F
Paciente F	Paciente G
Paciente G	<Null>

← Paciente 0

Modelado más sencillo

Personas

U1	Usuario1
U2	Usuario2
U3	Usuario3
.	.
.	.
.	.
UN	UsuarioN

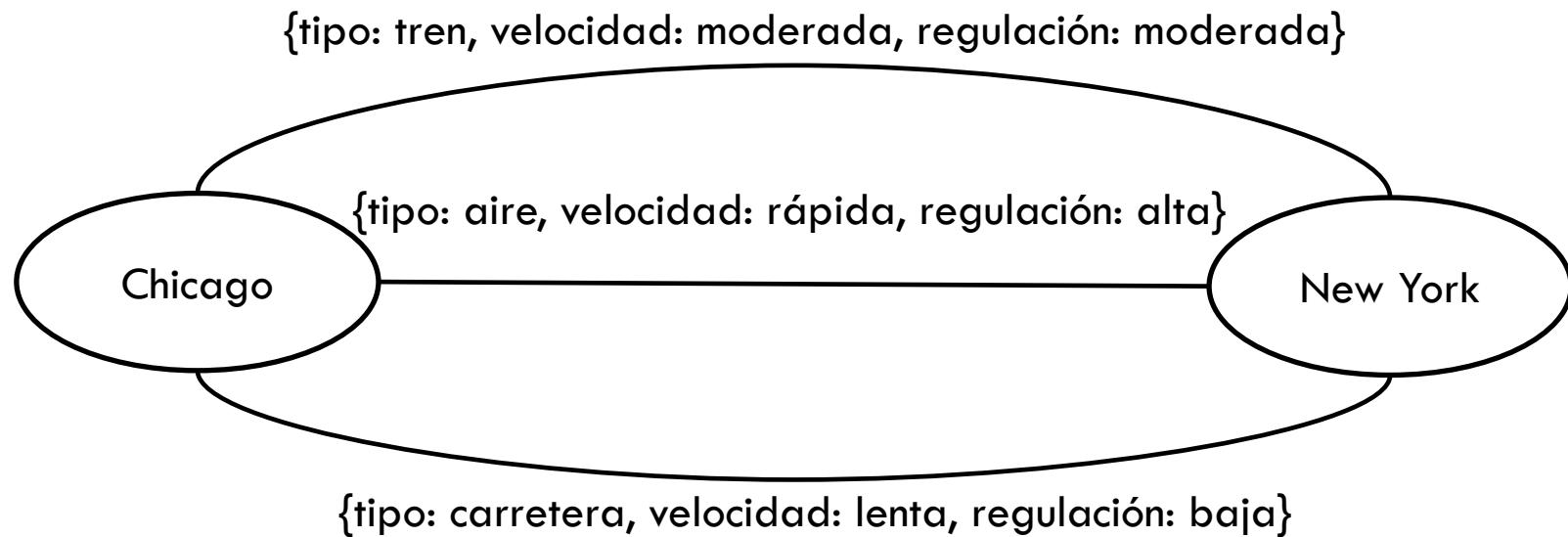
Likes

U1	P1
U1	P2
U1	P3
U2	P3
U3	PN
U3	P2
UN	P1
UN	P2

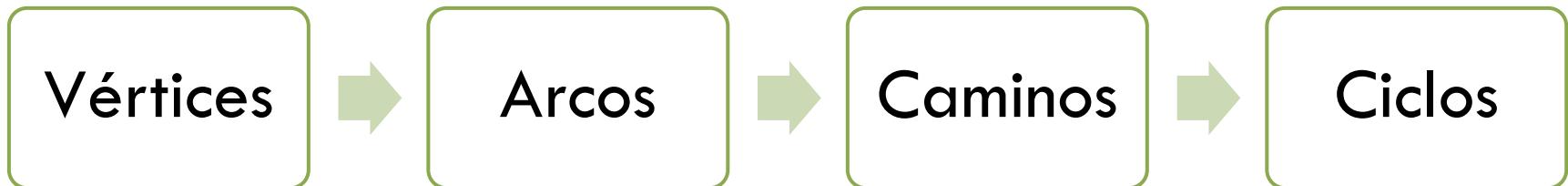
Posts

P1	Post1
P2	Post2
P3	Post3
.	.
.	.
.	.
PN	PostN

Múltiples relaciones entre entidades



Elementos de un grafo



Vértices



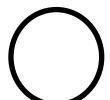
Madrid



Toledo



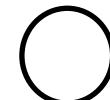
Cuenca



Barcelona



Santander

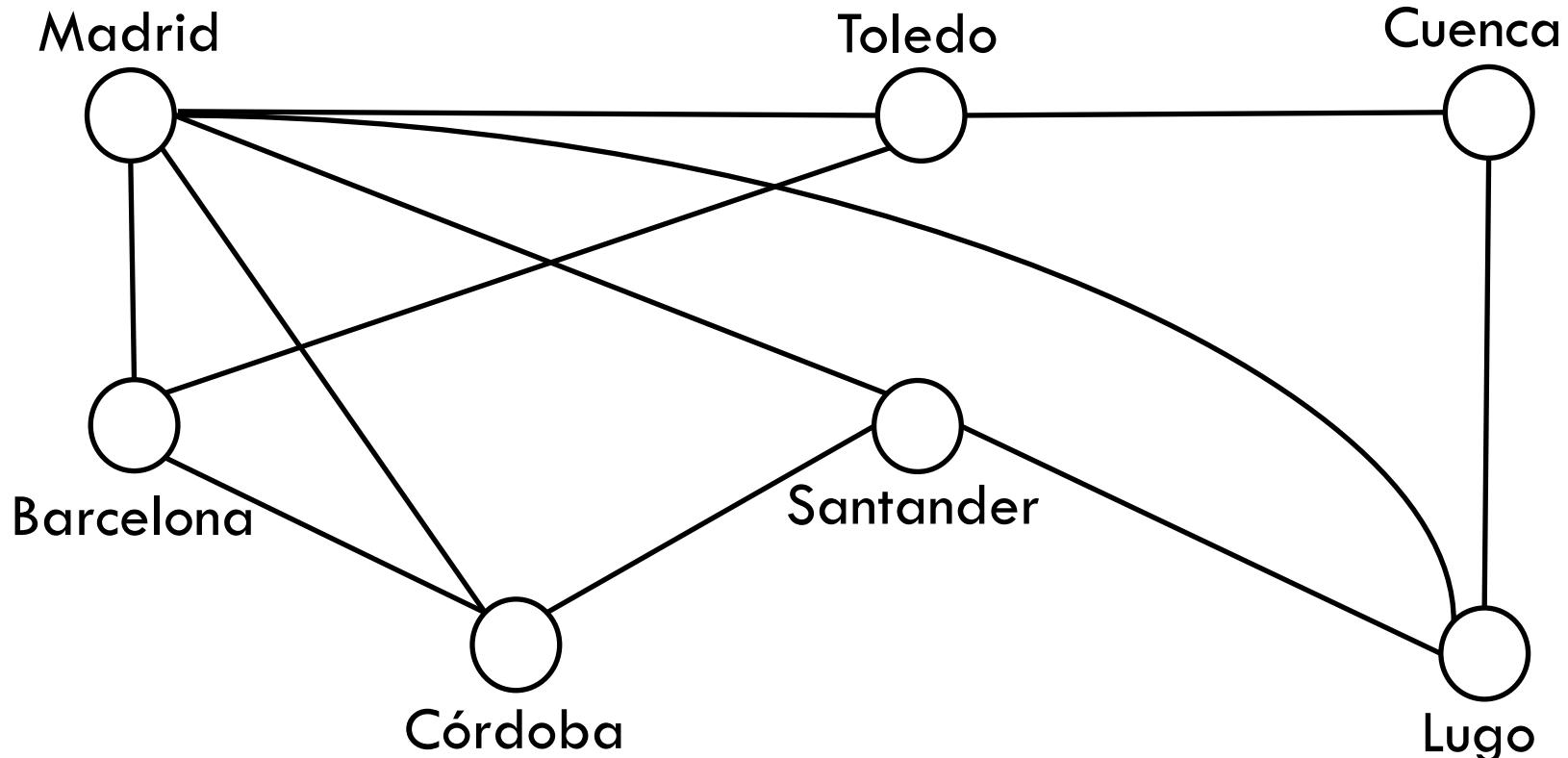


Córdoba

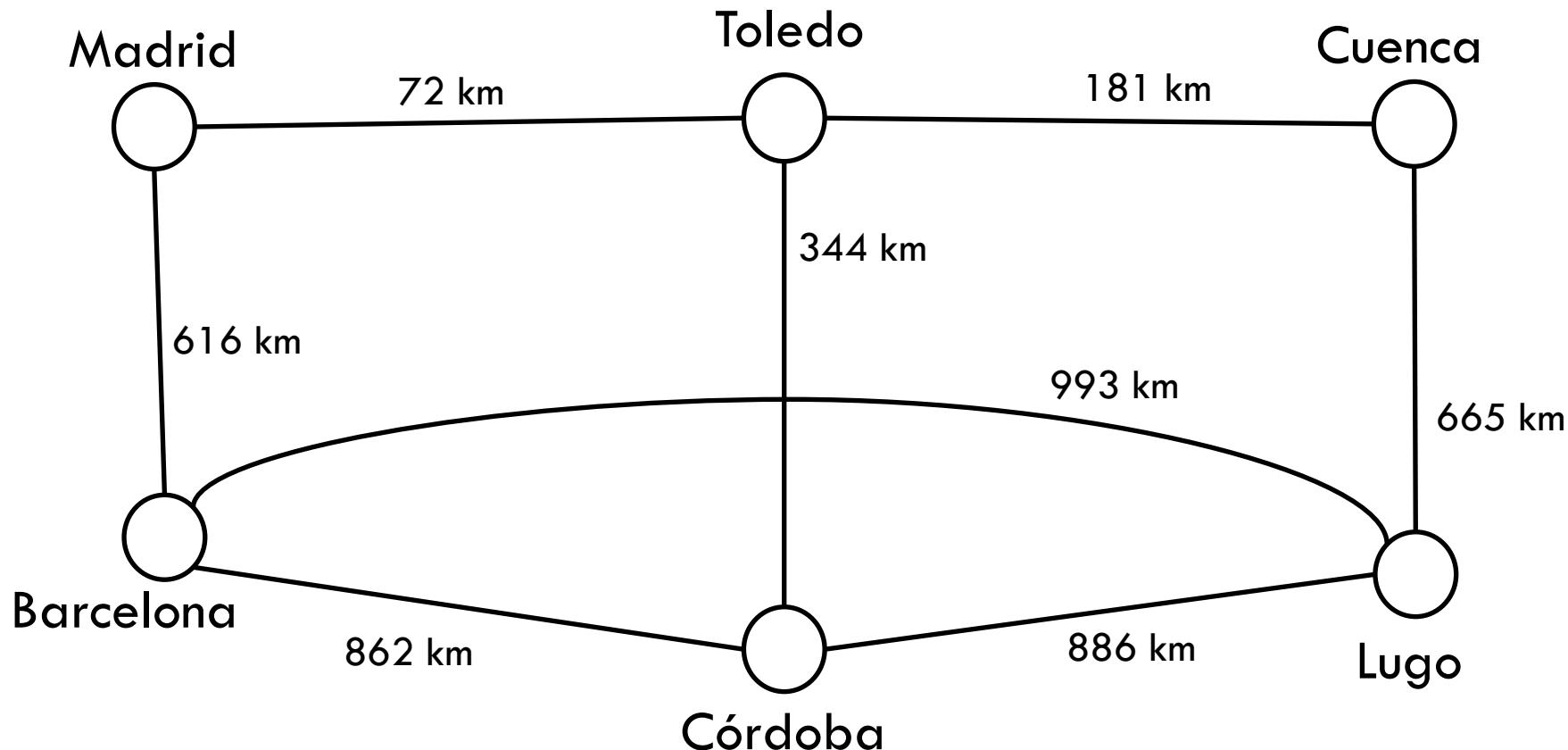


Lugo

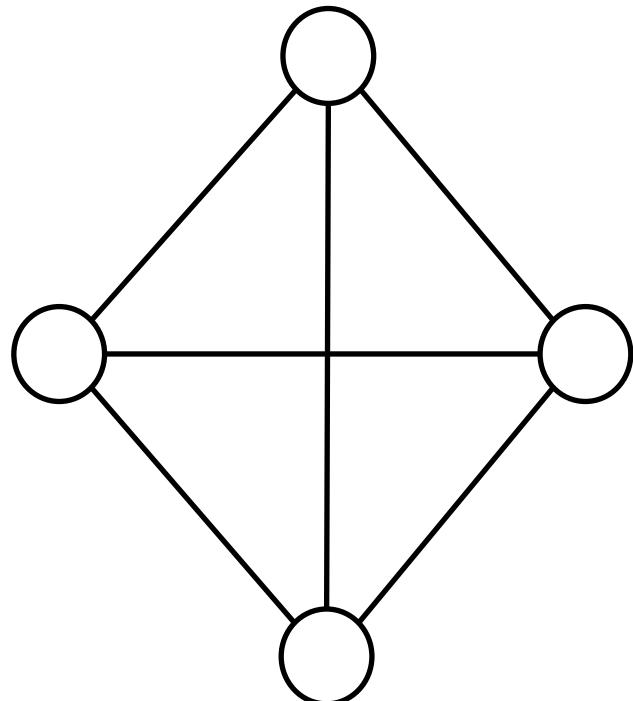
Arcos



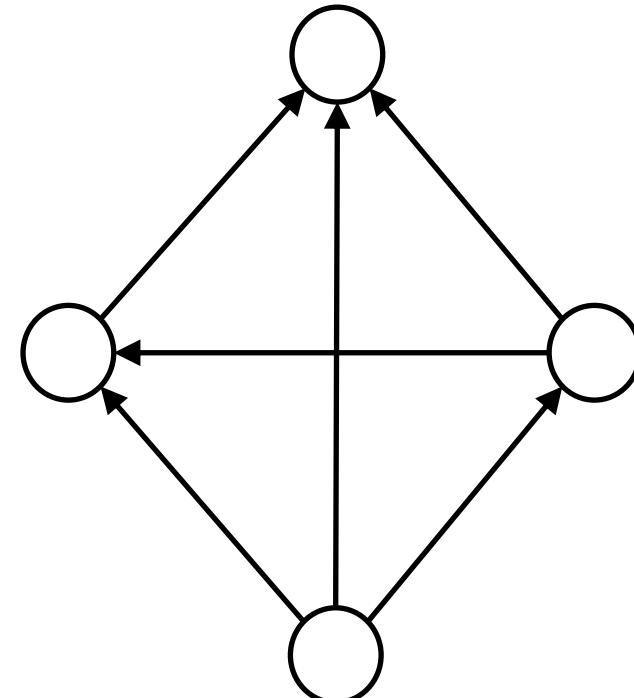
Arcos



Arcos

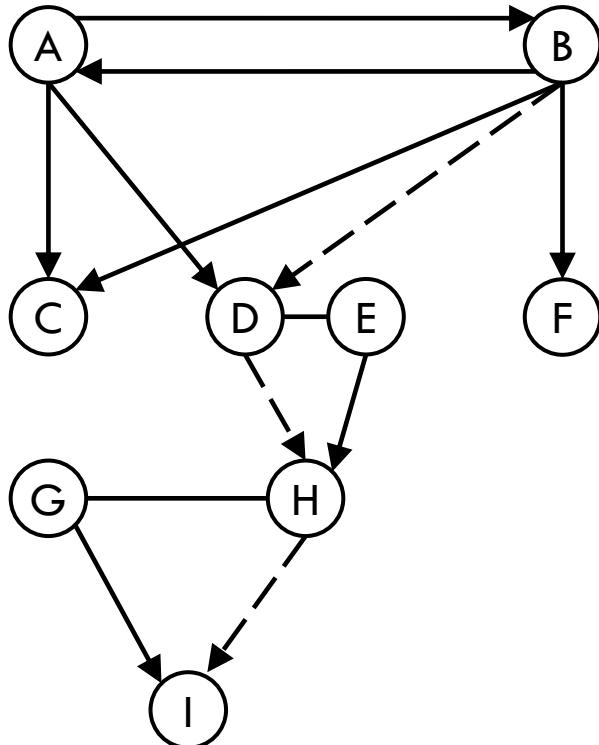


No dirigido



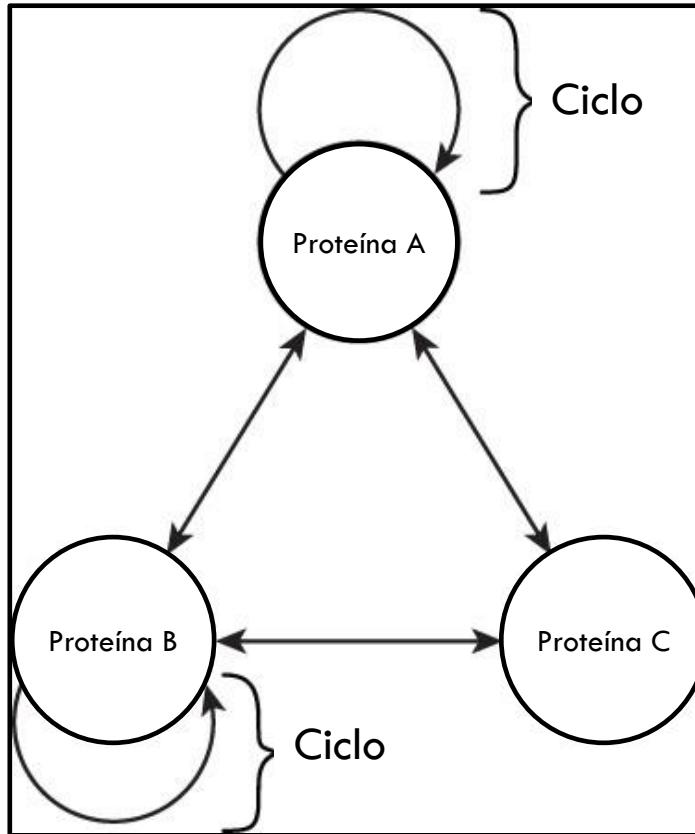
Dirigido

Caminos



Camino ancestros: I -> H -> D -> B

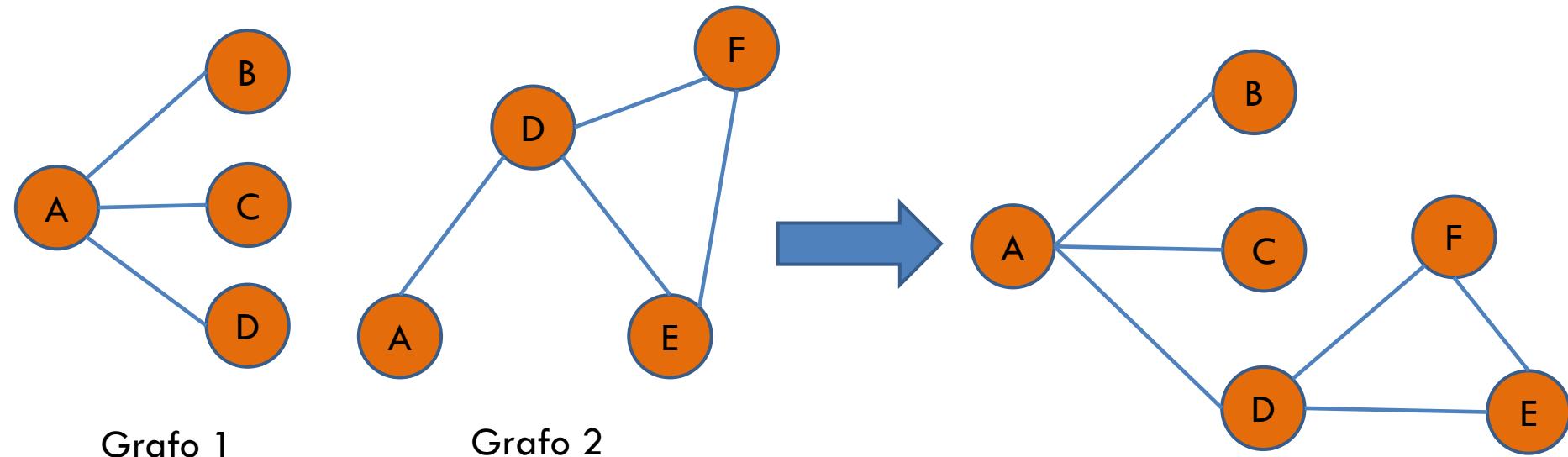
Ciclos



Operaciones con grafos



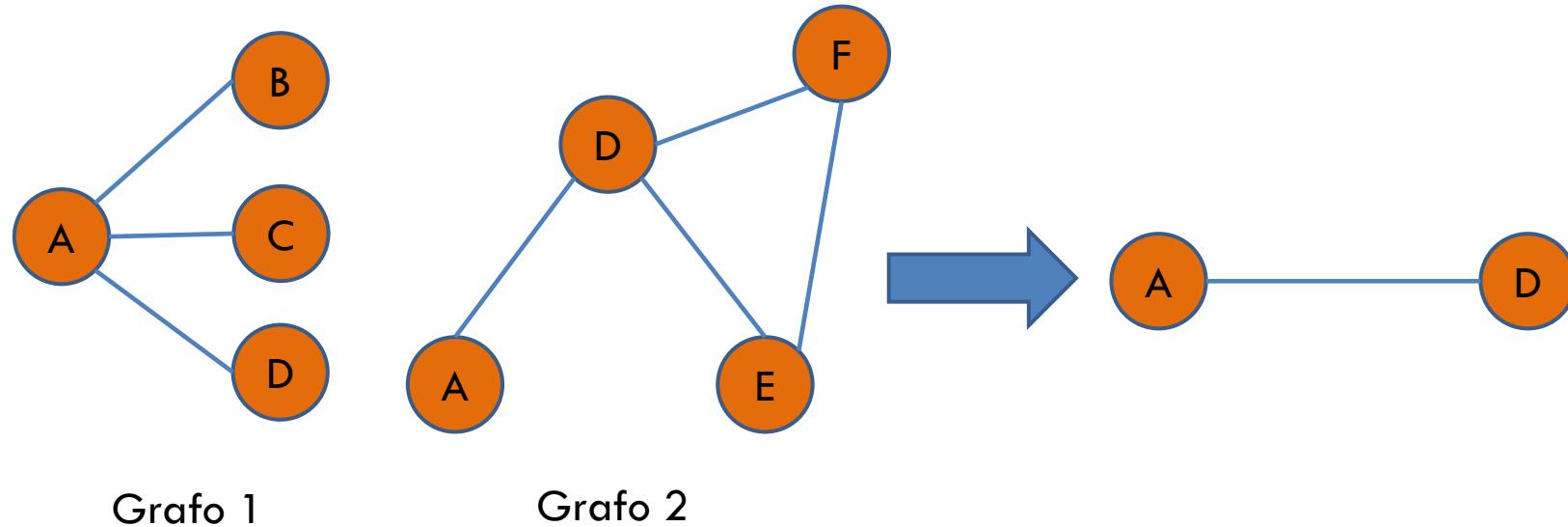
Unión



Grafo 1

Grafo 2

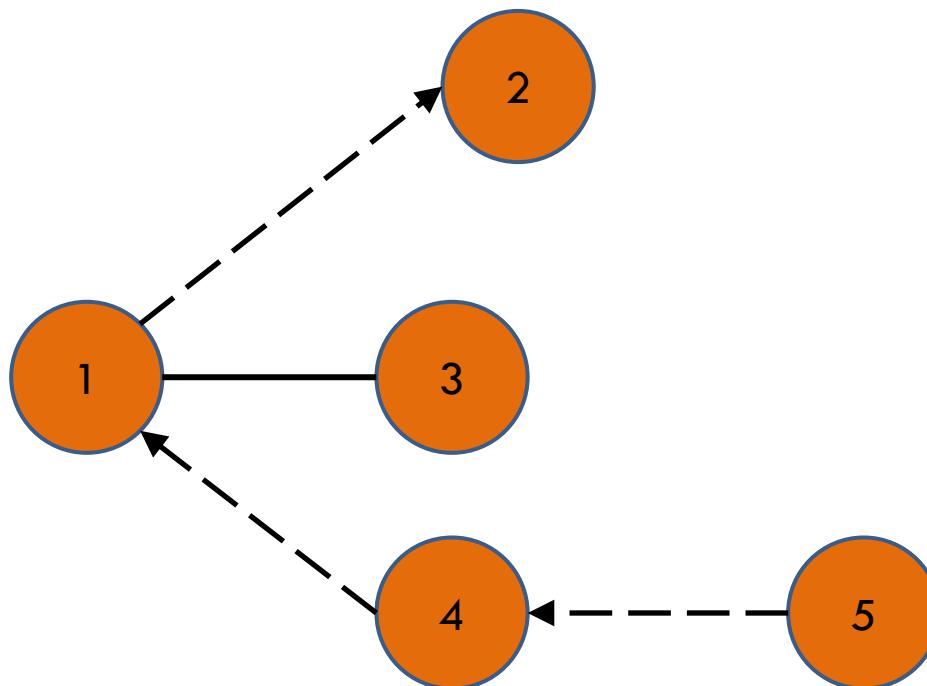
Intersección



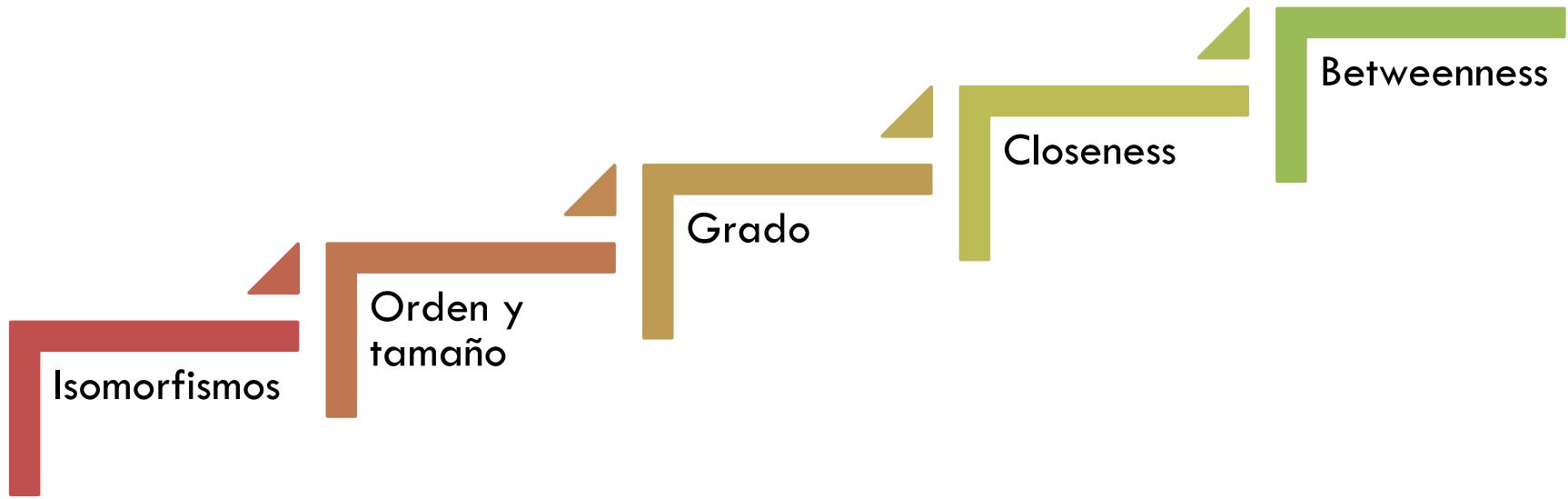
Grafo 1

Grafo 2

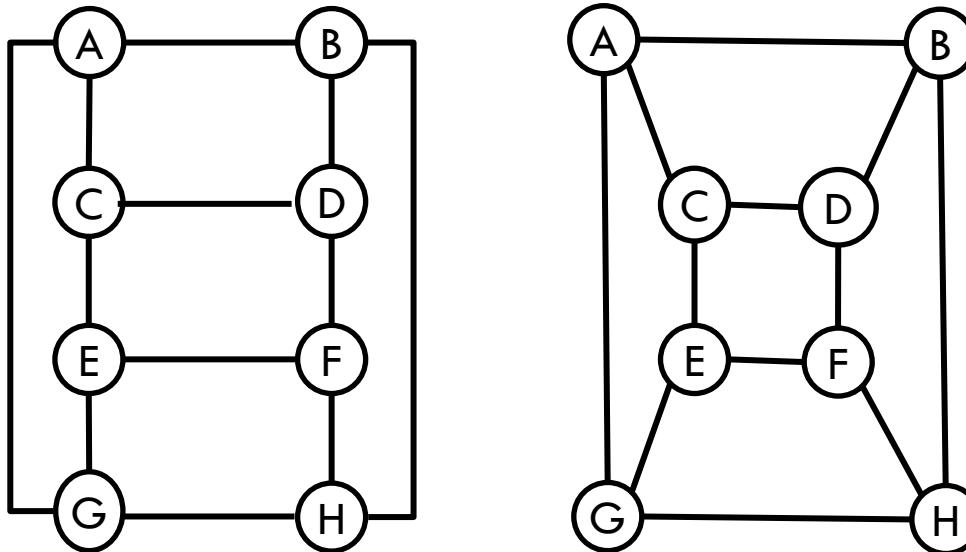
Graph traversal



Propiedades de los grafos y los nodos



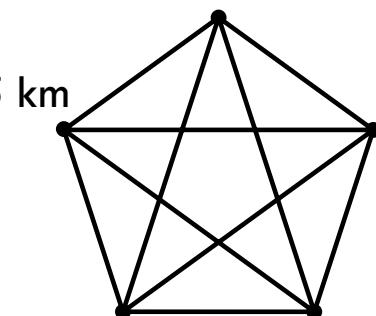
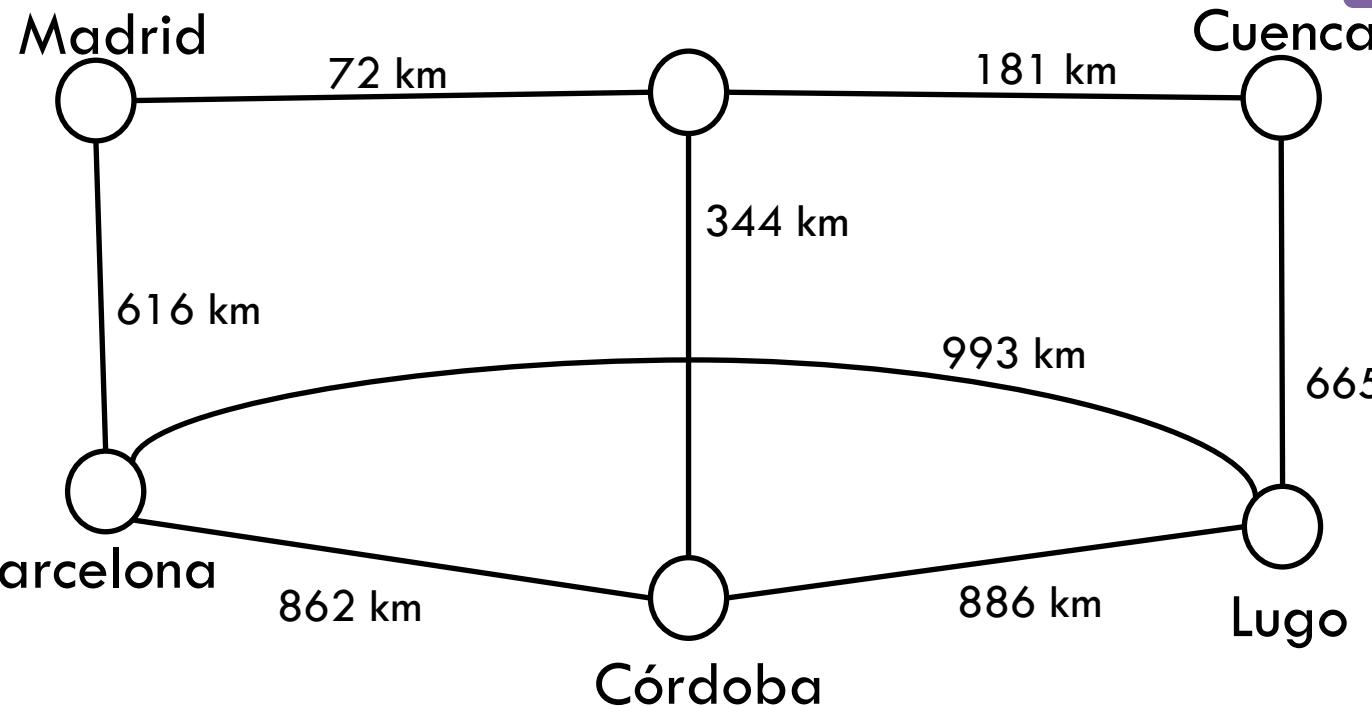
Isomorfismo



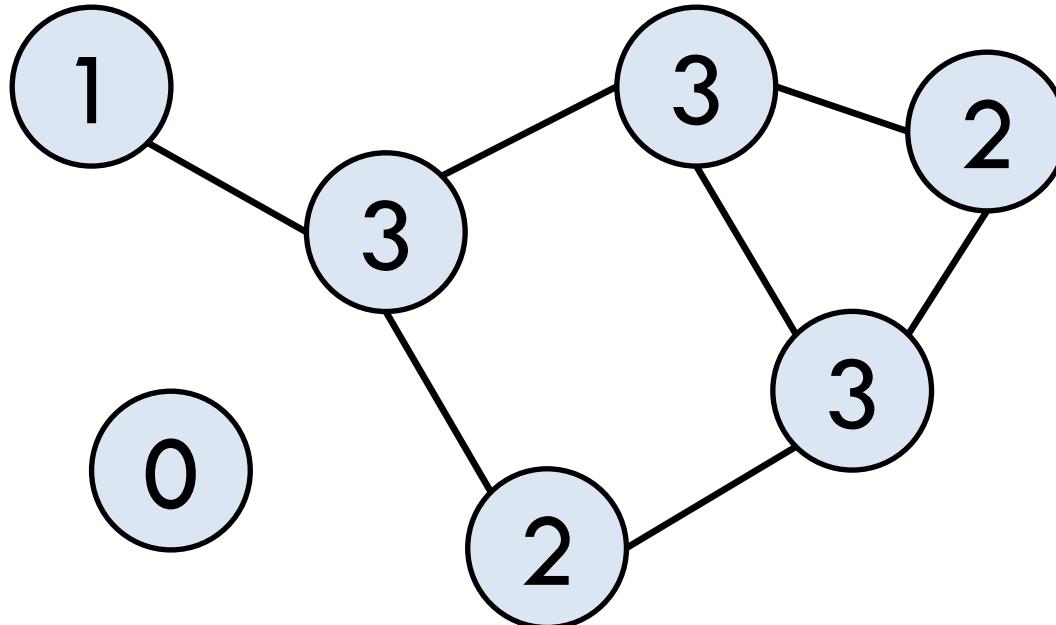
Orden y tamaño

Orden = 6

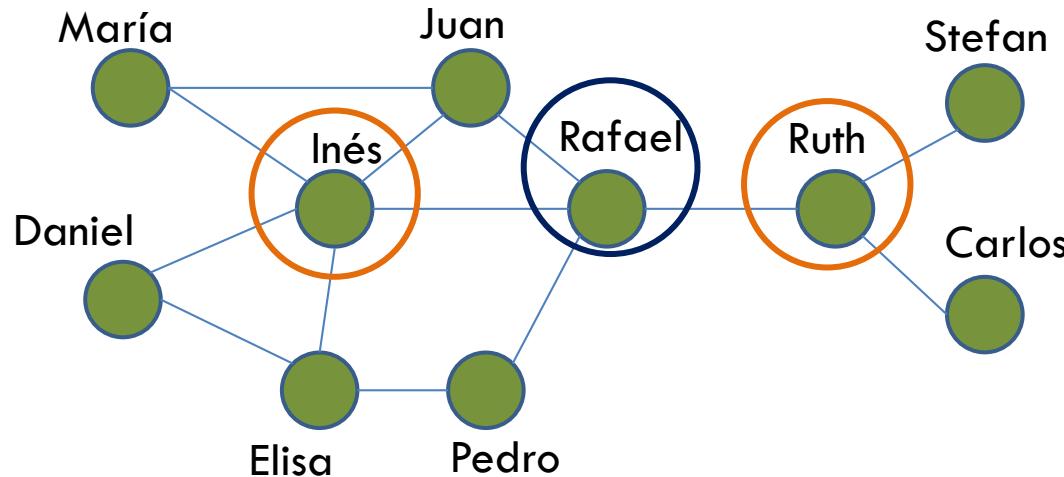
Tamaño = 8



Grado



Closeness

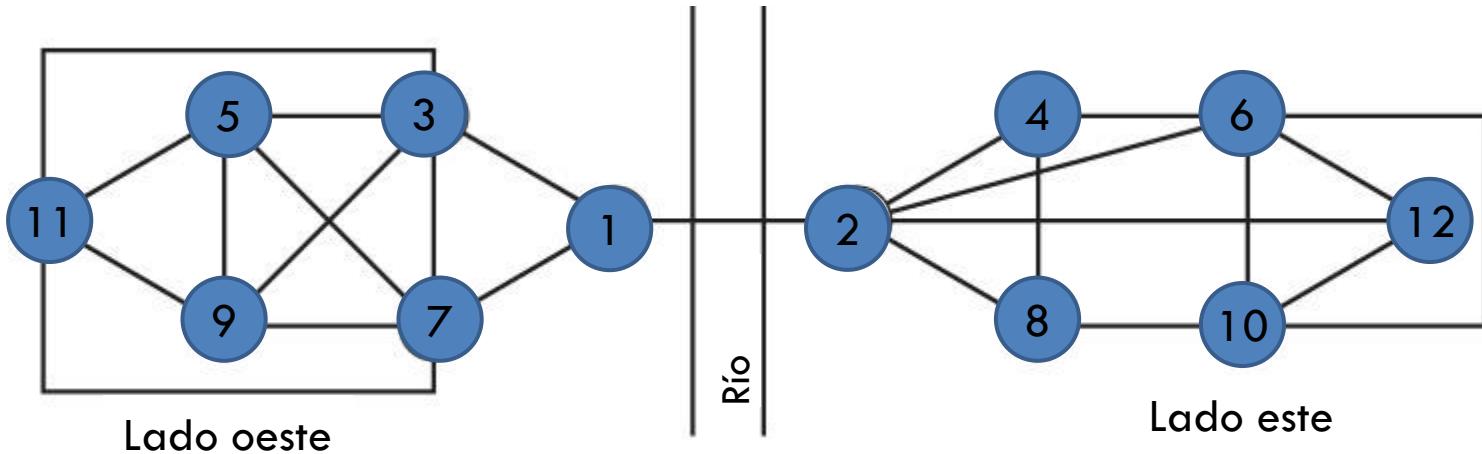


○ Closeness moderada

○ Closeness alta

$$C(x) = \frac{1}{\sum_y d(y, x)}$$

Betweenness



Tipos de grafos

Grafos
dirigidos y no
dirigidos

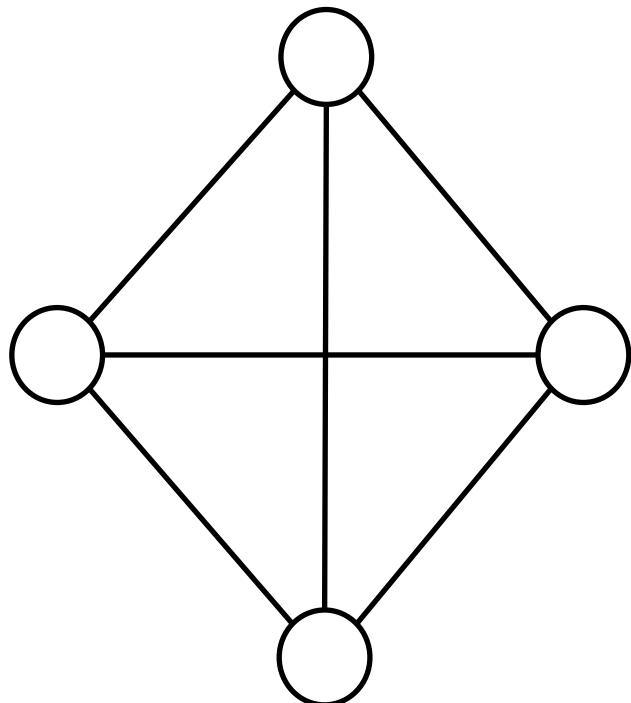
Redes de
flujo

Grafos
bipartitos

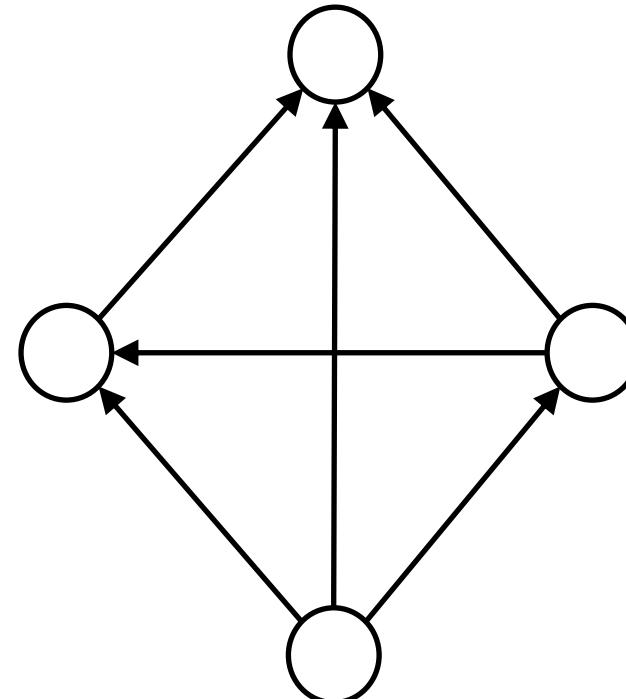
Multigrafos

Grafos
ponderados

Grafos dirigidos y no dirigidos

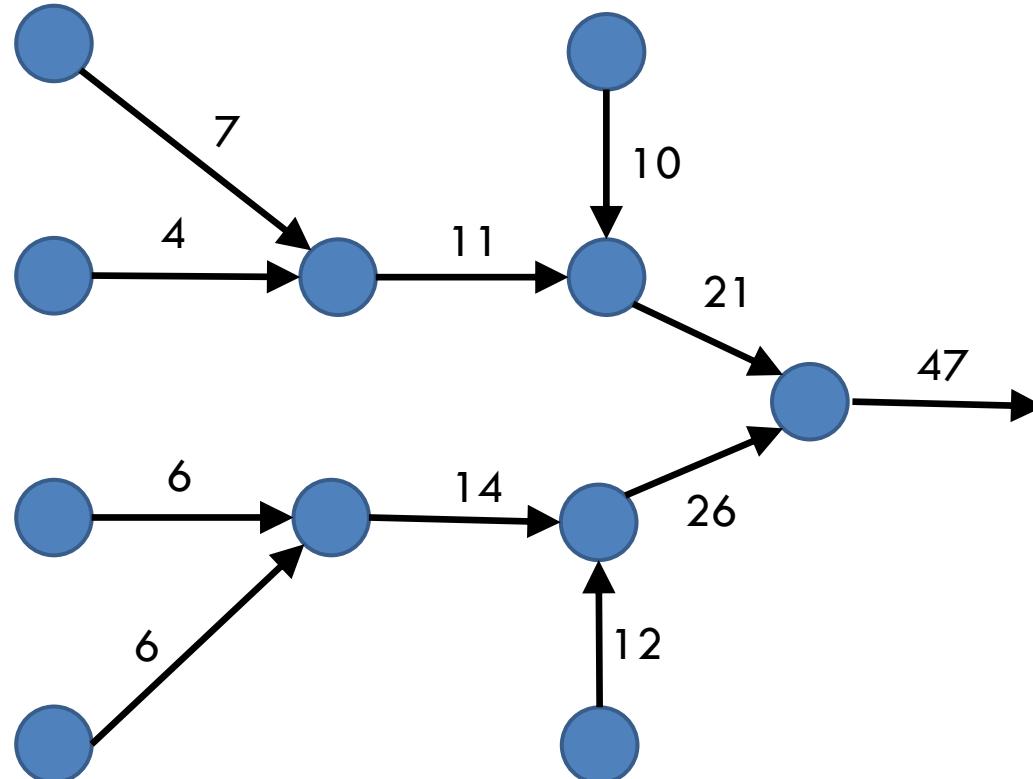


No dirigido

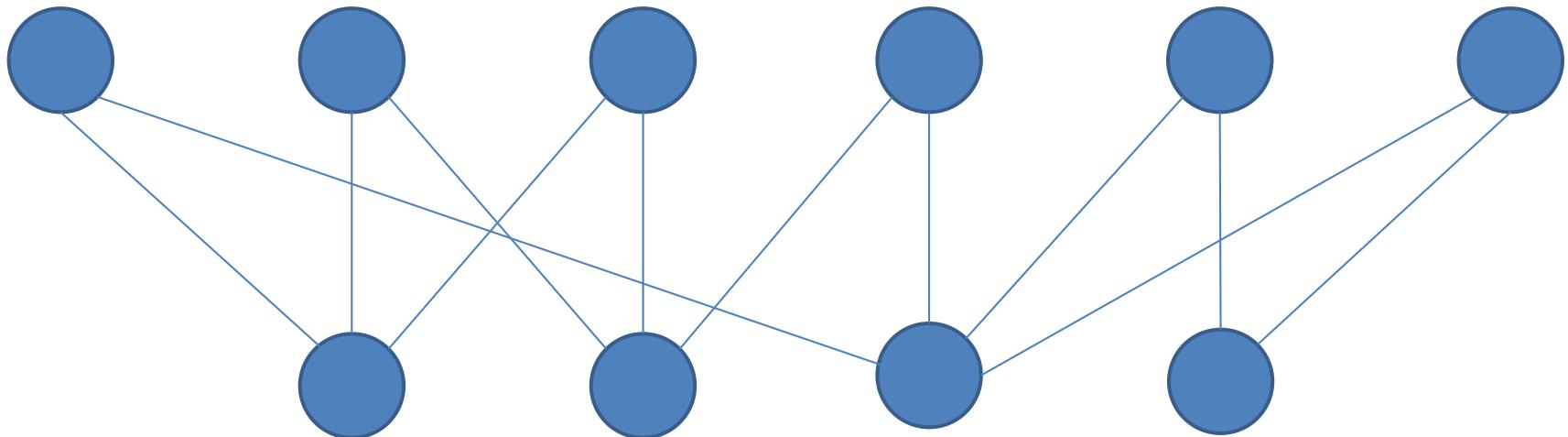


Dirigido

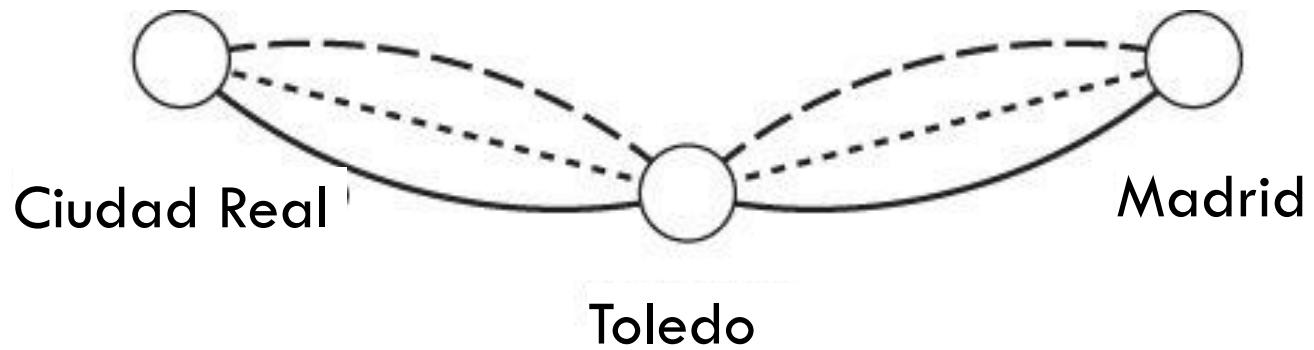
Redes de flujo



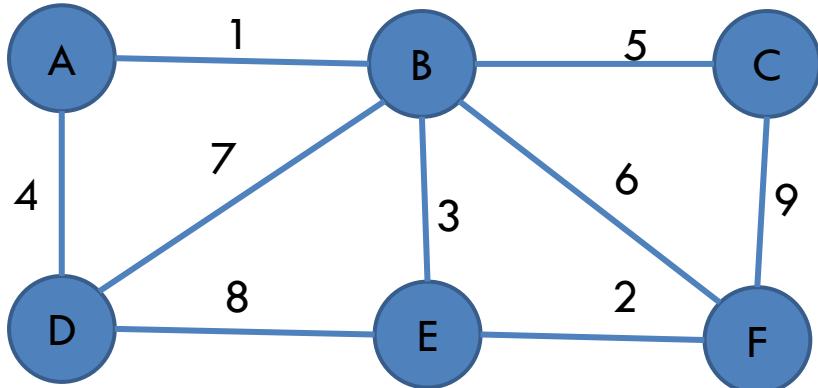
Grafos bipartitos



Multigrafos



Grafos ponderados



Número de vértices	Unidades de tiempo de ejecución
1	1
5	25
10	100
20	400
50	2500
100	10000
1000	1000000

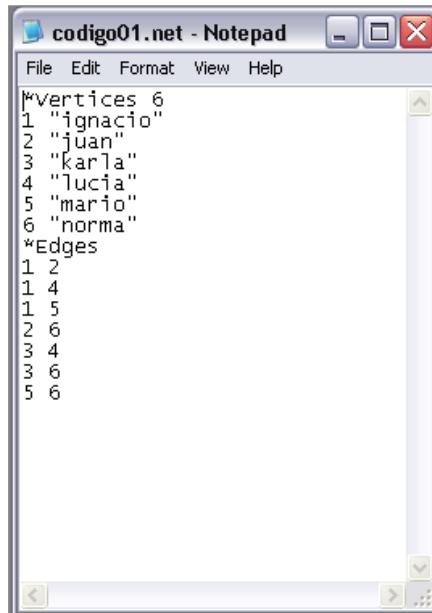
Pajek

<http://mrvar.fdv.uni-lj.si/pajek/default.htm>

Pajek

- Es un software para el análisis y visualización de redes sociales
- Desarrollado en la universidad de Ljubljana, Slovenia

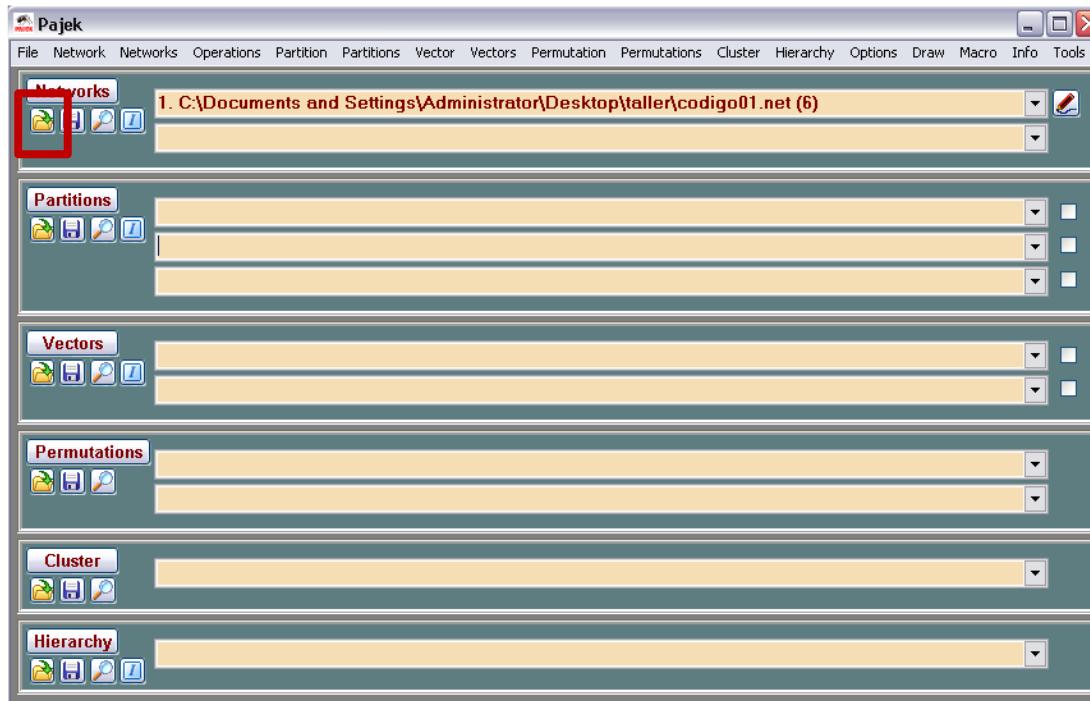
Archivo .net



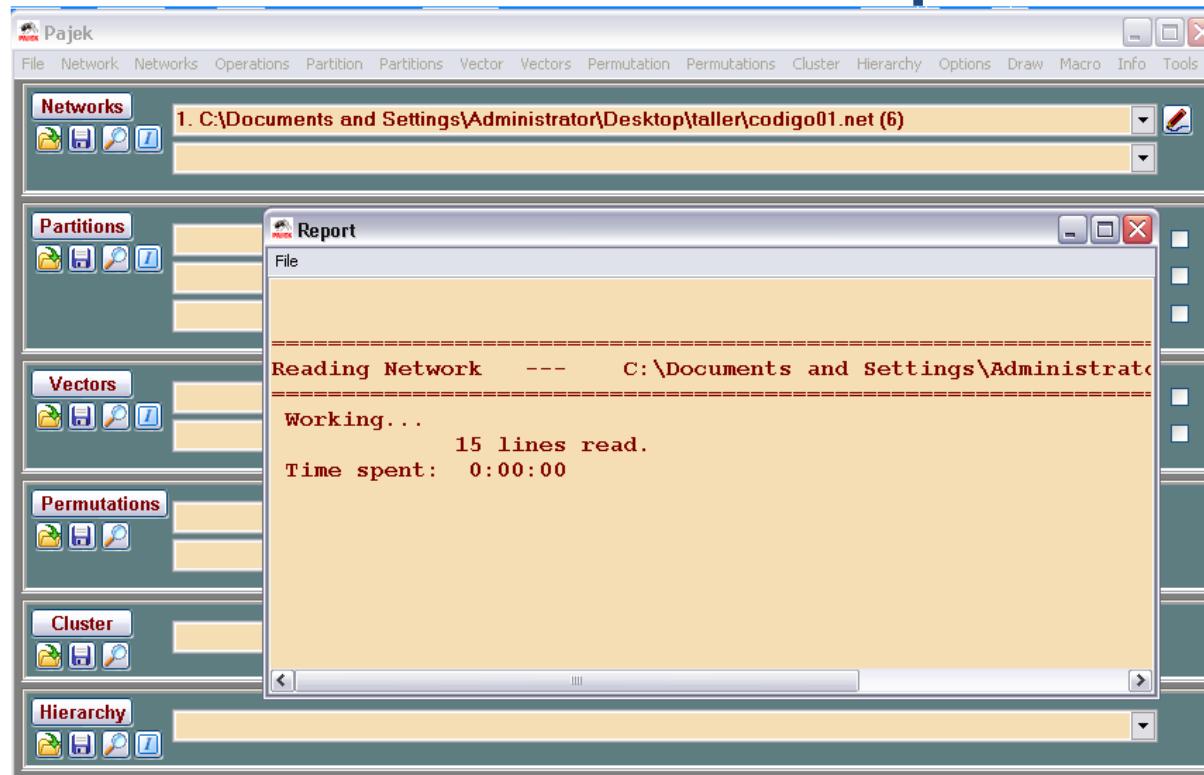
The image shows a screenshot of a Windows Notepad window titled "codigo01.net - Notepad". The window contains the following text:

```
*Vertices 6
1 "ignacio"
2 "juan"
3 "karla"
4 "lucia"
5 "mario"
6 "norma"
*Edges
1 2
1 4
1 5
2 6
3 4
3 6
5 6
```

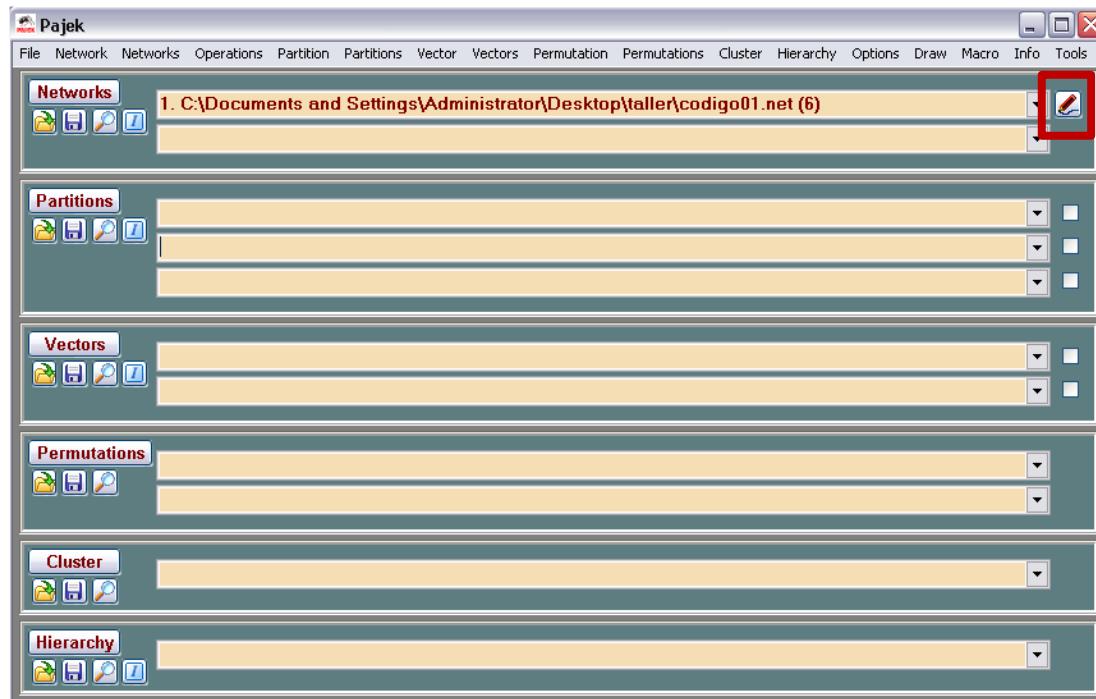
Cargar fichero .net



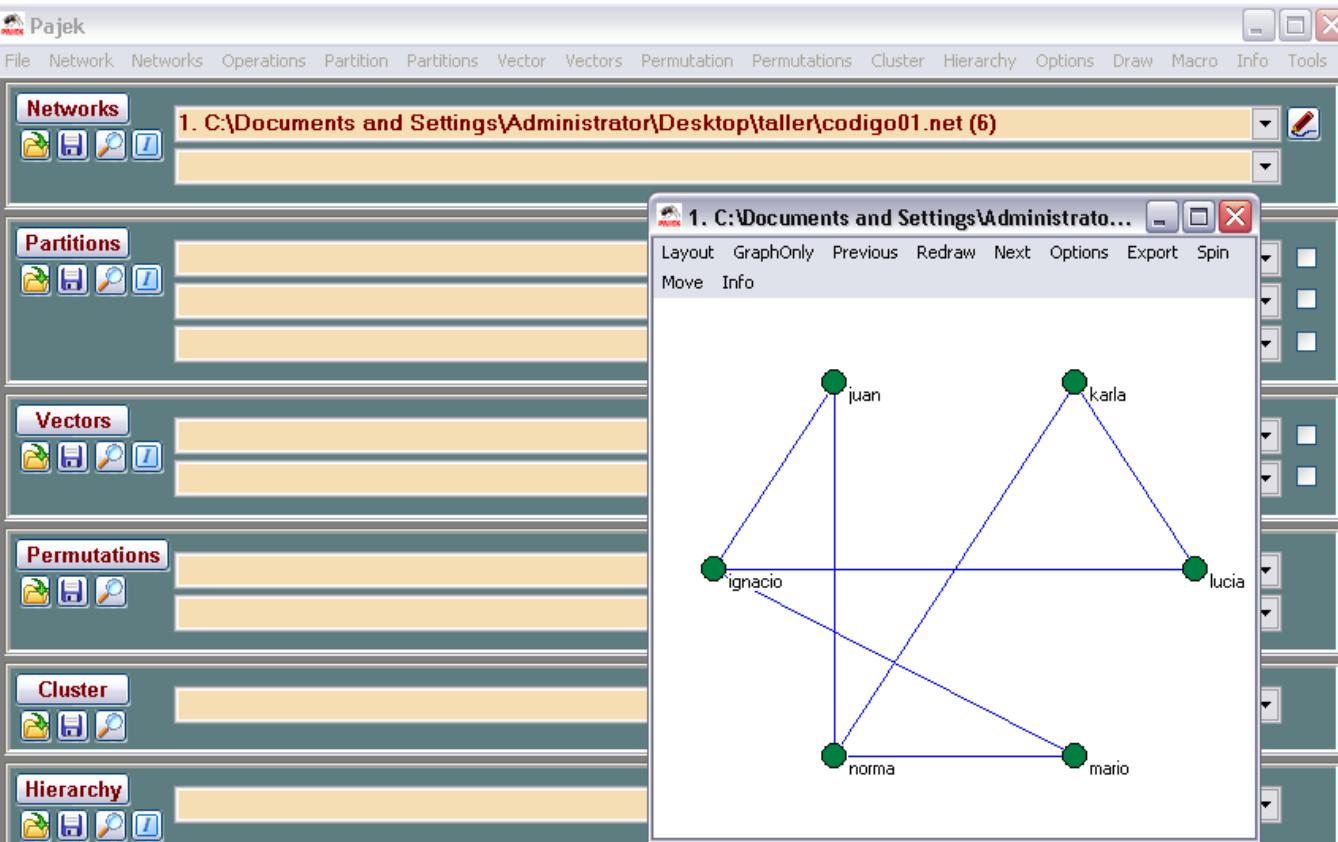
Aparece ventana de Report



Para visualizar el grafo



Aparece



Options – Mark vertices using - Labels

Relaciones

Parentesco

	Ignacio	Juan	Karla	Lucia	Mario	Norma
Ignacio				1	1	
Juan						1
Karla						
Lucia	1				1	
Mario	1			1		
Norma		1				

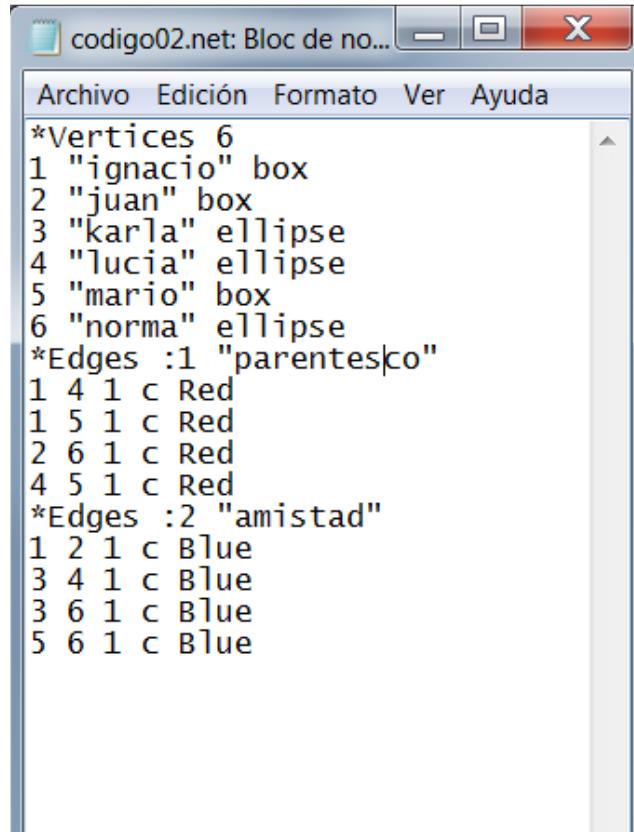
Amistad

	Ignacio	Juan	Karla	Lucia	Mario	Norma
Ignacio			1			
Juan	1					
Karla					1	
Lucia				1		
Mario						1
Norma				1		1

En Pajek

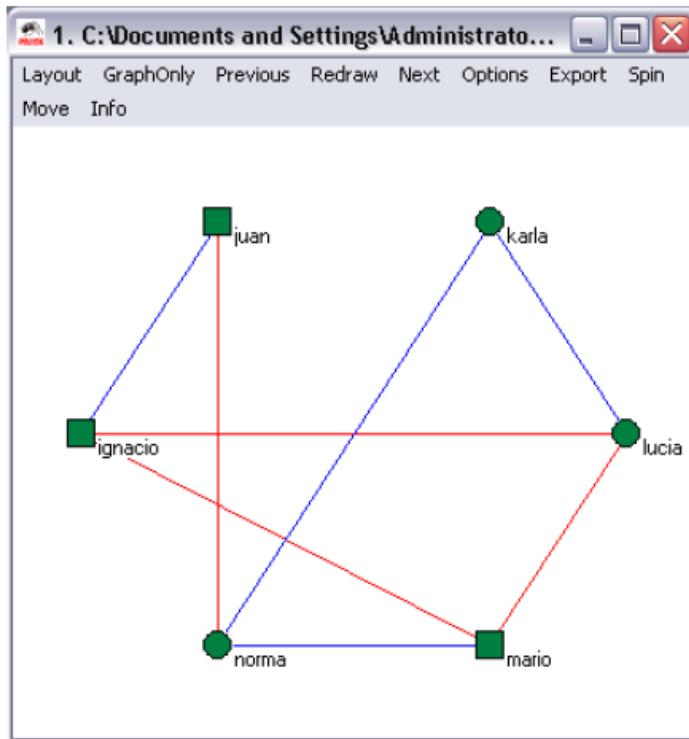
- Para visualizar:
comprobar en ventana
de visualización:

Options -> Colors
-> Edges -> As Defined
on Input File

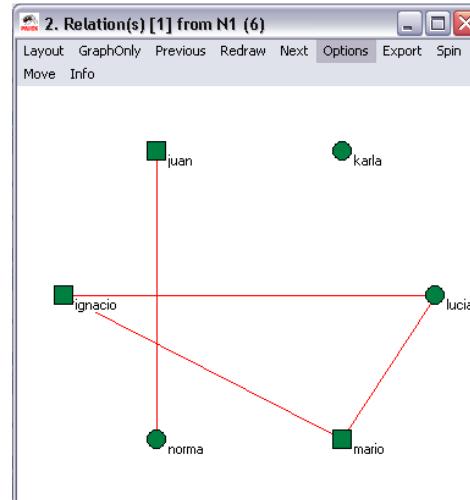
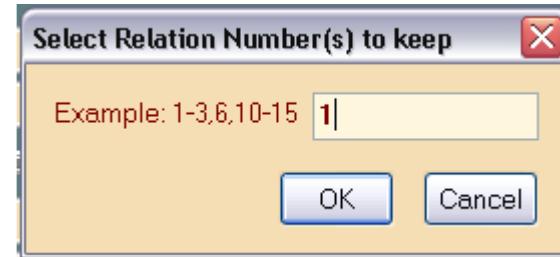
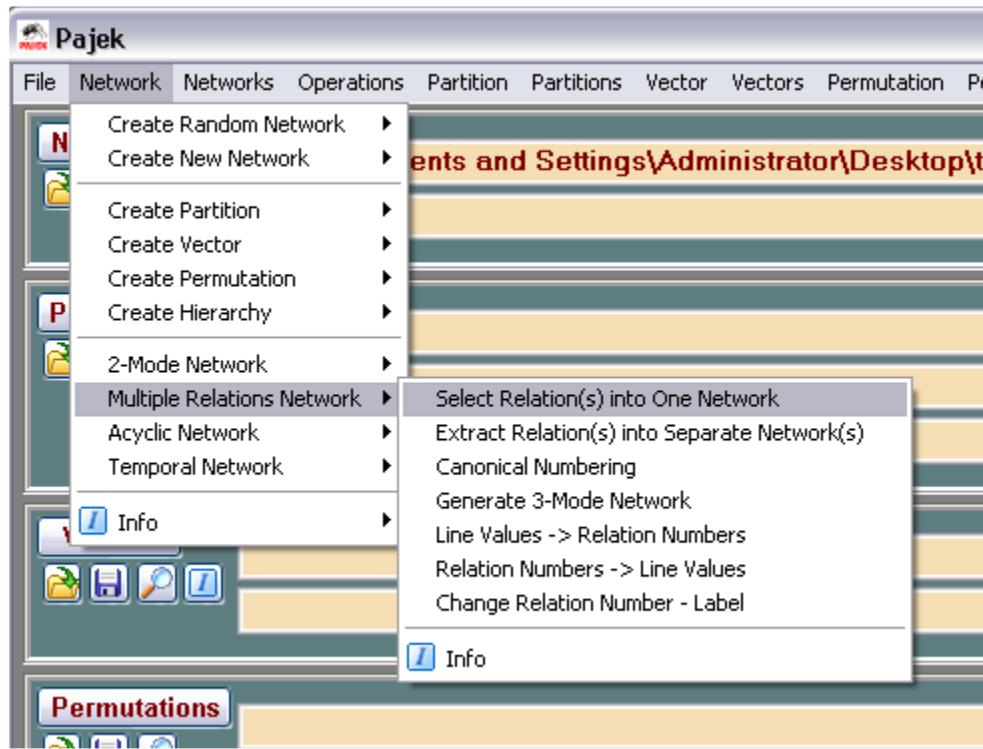


```
*Vertices 6
1 "ignacio" box
2 "juan" box
3 "karla" ellipse
4 "lucia" ellipse
5 "mario" box
6 "norma" ellipse
*Edges :1 "parentesco"
1 4 1 c Red
1 5 1 c Red
2 6 1 c Red
4 5 1 c Red
*Edges :2 "amistad"
1 2 1 c Blue
3 4 1 c Blue
3 6 1 c Blue
5 6 1 c Blue
```

- M
 - F
- parentesco
— amistad



Mostrar solo un tipo de relación



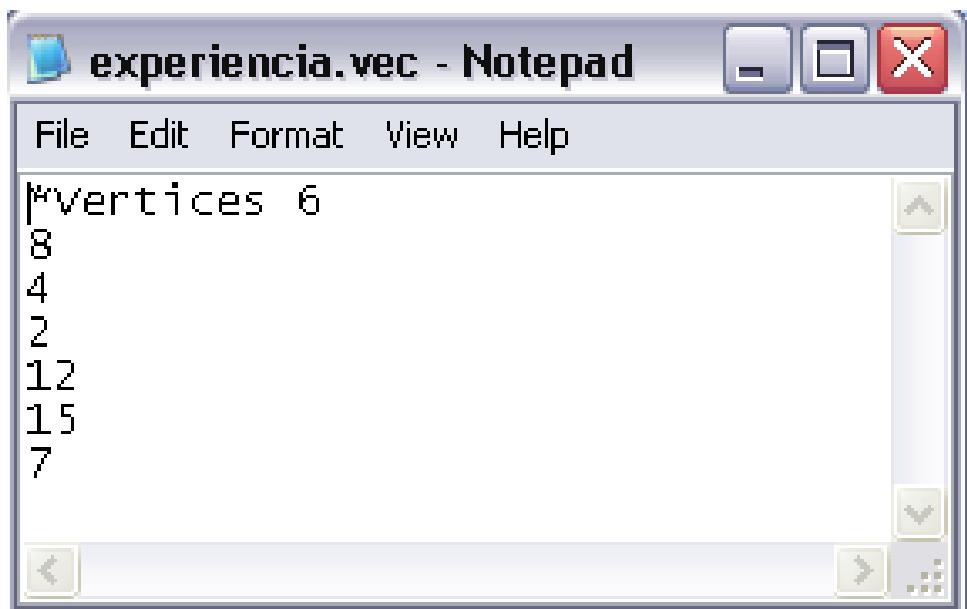
Atributos

Casos
(individuos)

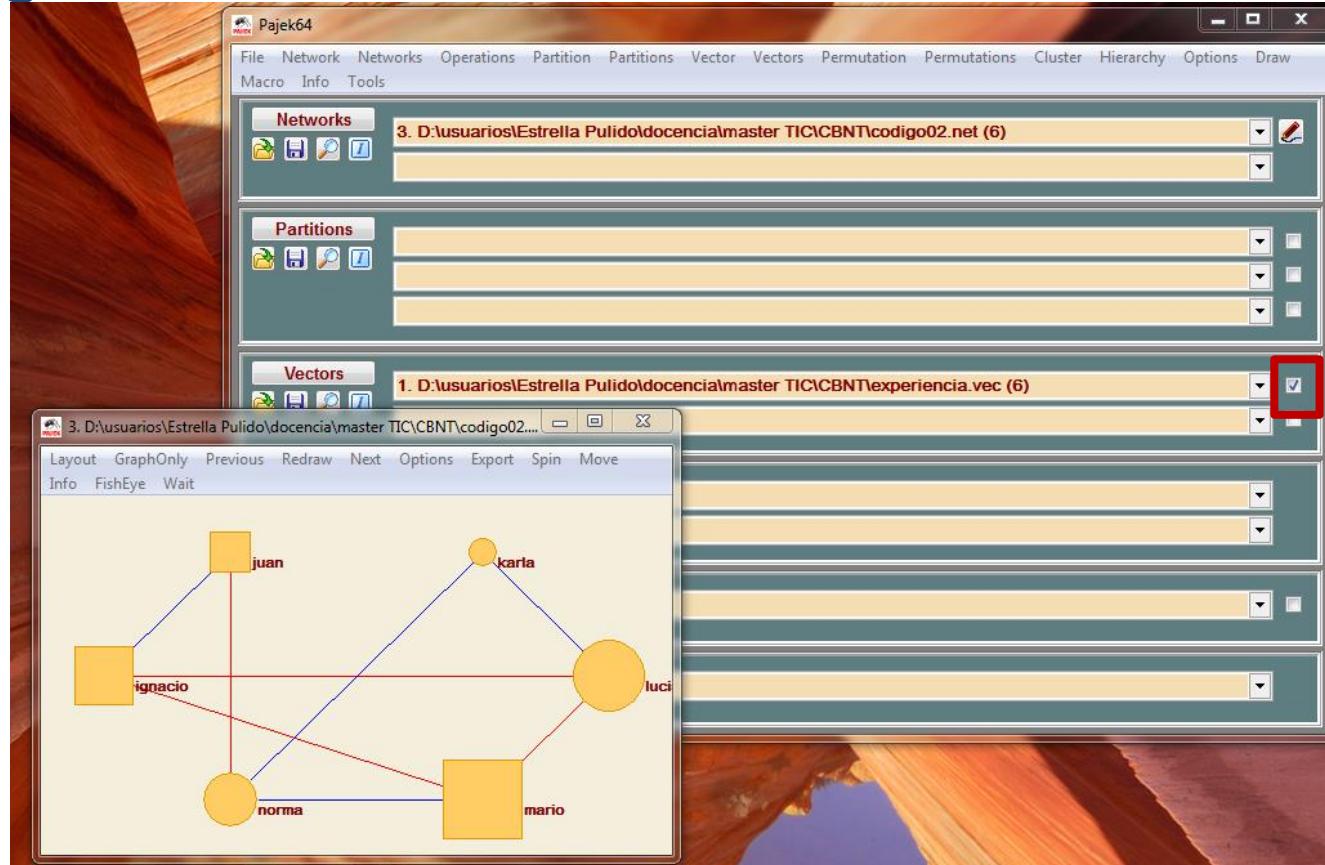
Variables

	sexo	Años	Lugar de trabajo
Ignacio	M	8	UNAM
Juan	M	4	IPN
Karla	F	2	UAM
Lucia	F	12	UNAM
Mario	M	15	UNAM
Norma	F	7	UAM

Archivo .vec

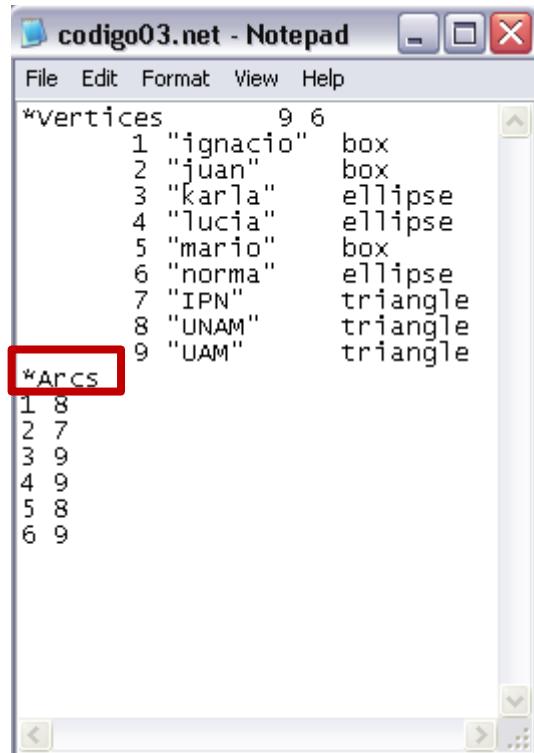


Cargar el fichero



Grafos multimodales

- En la definición de los vértices se considerará el número total de individuos e instituciones (9) y a continuación se indica en qué número termina la definición de los individuos (6).

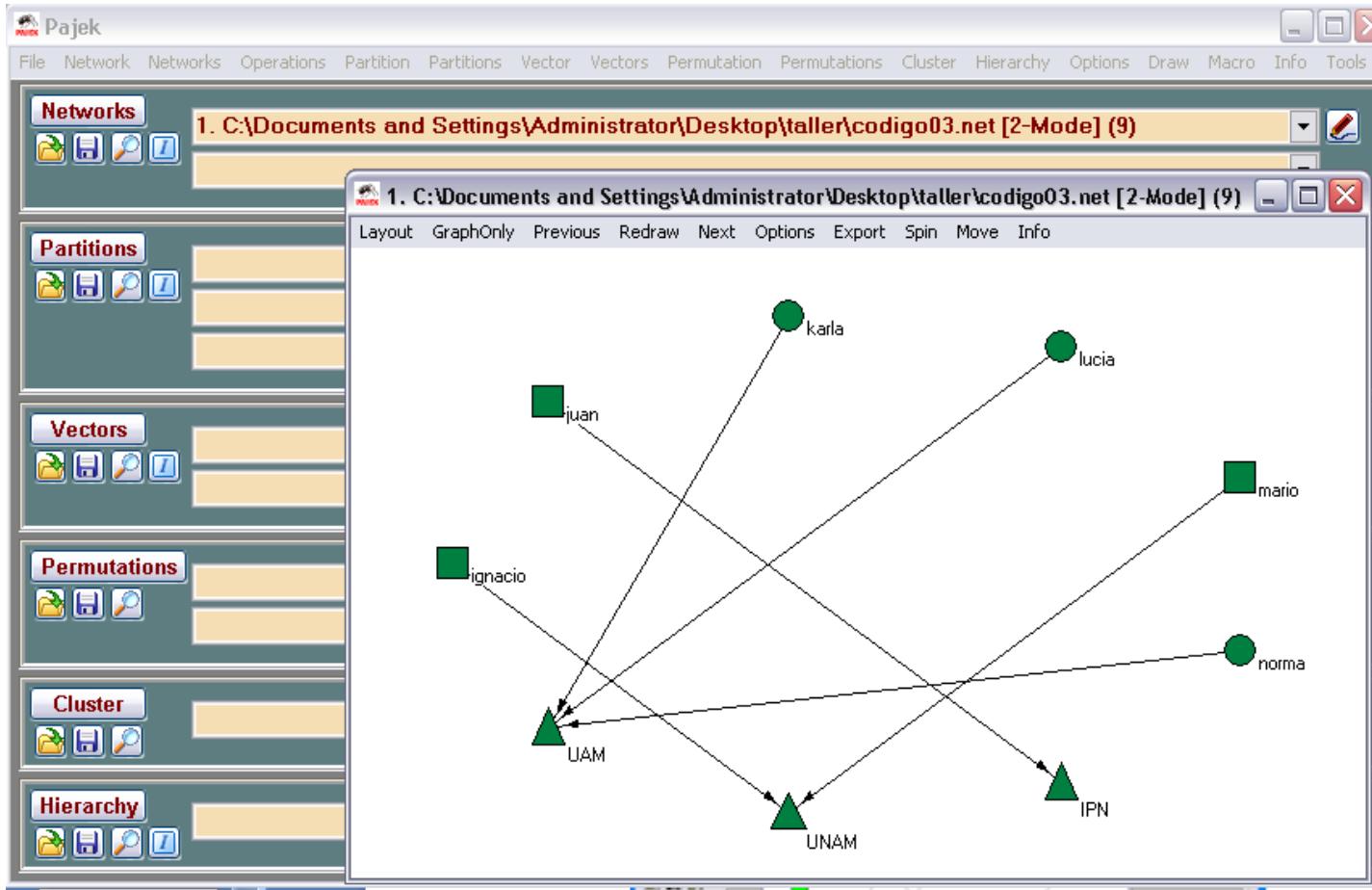


The screenshot shows a Windows Notepad window titled "codigo03.net - Notepad". The content is a text-based definition of a multimodal graph. It starts with a section for vertices, followed by a section for arcs.

```
*vertices 9 6
1 "ignacio" box
2 "juan" box
3 "karla" ellipse
4 "lucia" ellipse
5 "mario" box
6 "norma" ellipse
7 "IPN" triangle
8 "UNAM" triangle
9 "UAM" triangle

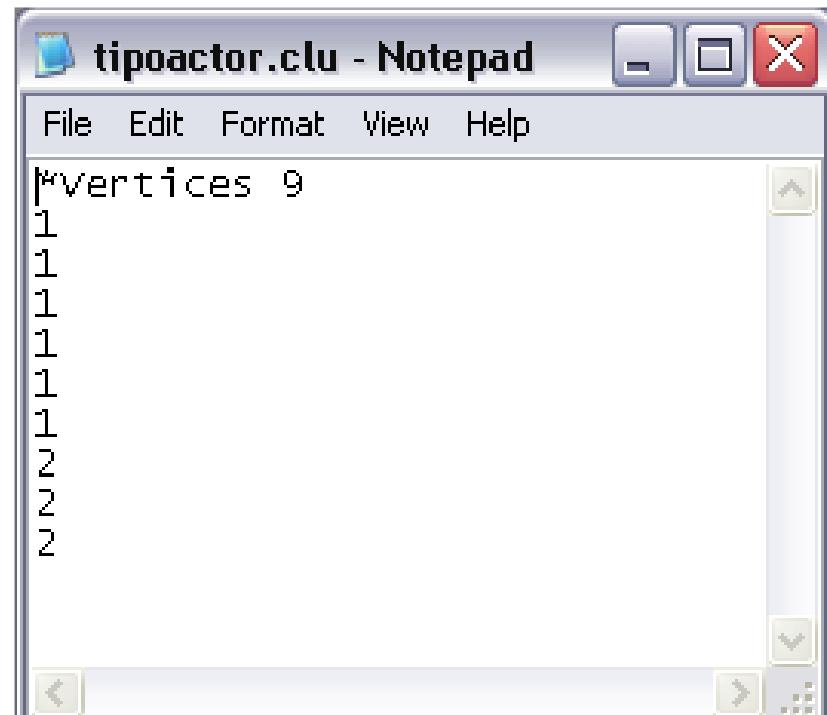
*Arcs
1 8
2 7
3 9
4 9
5 8
6 9
```

A red rectangular box highlights the word "Arcs" in the second section of the text.



Particiones: fichero .clu

- Podemos hacer lo mismo usando particiones
- Asignamos un tipo a cada vértice conservando el orden en el que aparecen en el fichero .net

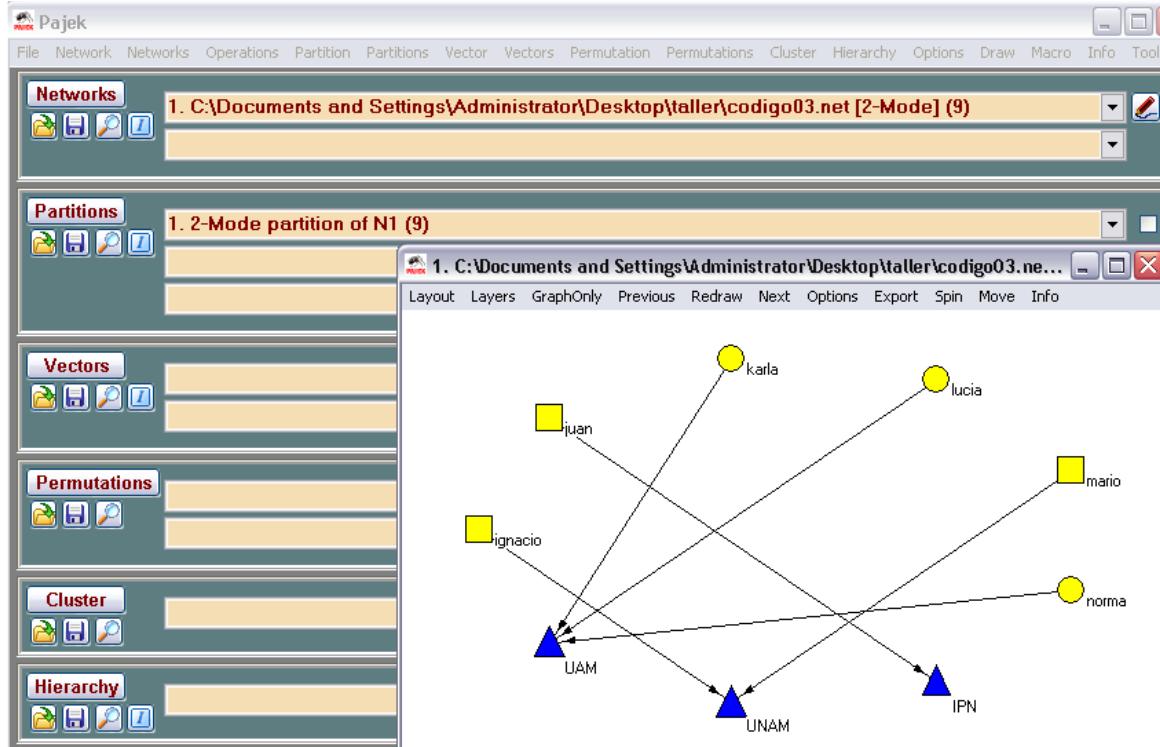


The screenshot shows a Windows Notepad window titled "tipoactor.clu - Notepad". The menu bar includes File, Edit, Format, View, and Help. The main text area contains the following content:

```
*Vertices 9
1
1
1
1
1
1
2
2
2
```

Para visualizar :

Draw->Network + First Partition



```

*Vertices      9 6
  1 "ignacio"      0.1299    0.0845    0.5000 box ic Green
  2 "juan"         0.8164    0.1026    0.5000 box ic Green
  3 "karla"        0.5253    0.9386    0.5000 ellipse ic Red
  4 "lucia"        0.0664    0.7379    0.5000 ellipse ic Red
  5 "mario"        0.4189    0.5303    0.5000 box ic Green
  6 "norma"        0.8184    0.5708    0.5000 ellipse ic Red
  7 "IPN"          0.5873    0.3139    0.5000 triangle ic Yellow
  8 "UNAM"         0.1735    0.4652    0.5000 triangle ic Yellow
  9 "UAM"          0.4834    0.7139    0.5000 triangle ic Yellow

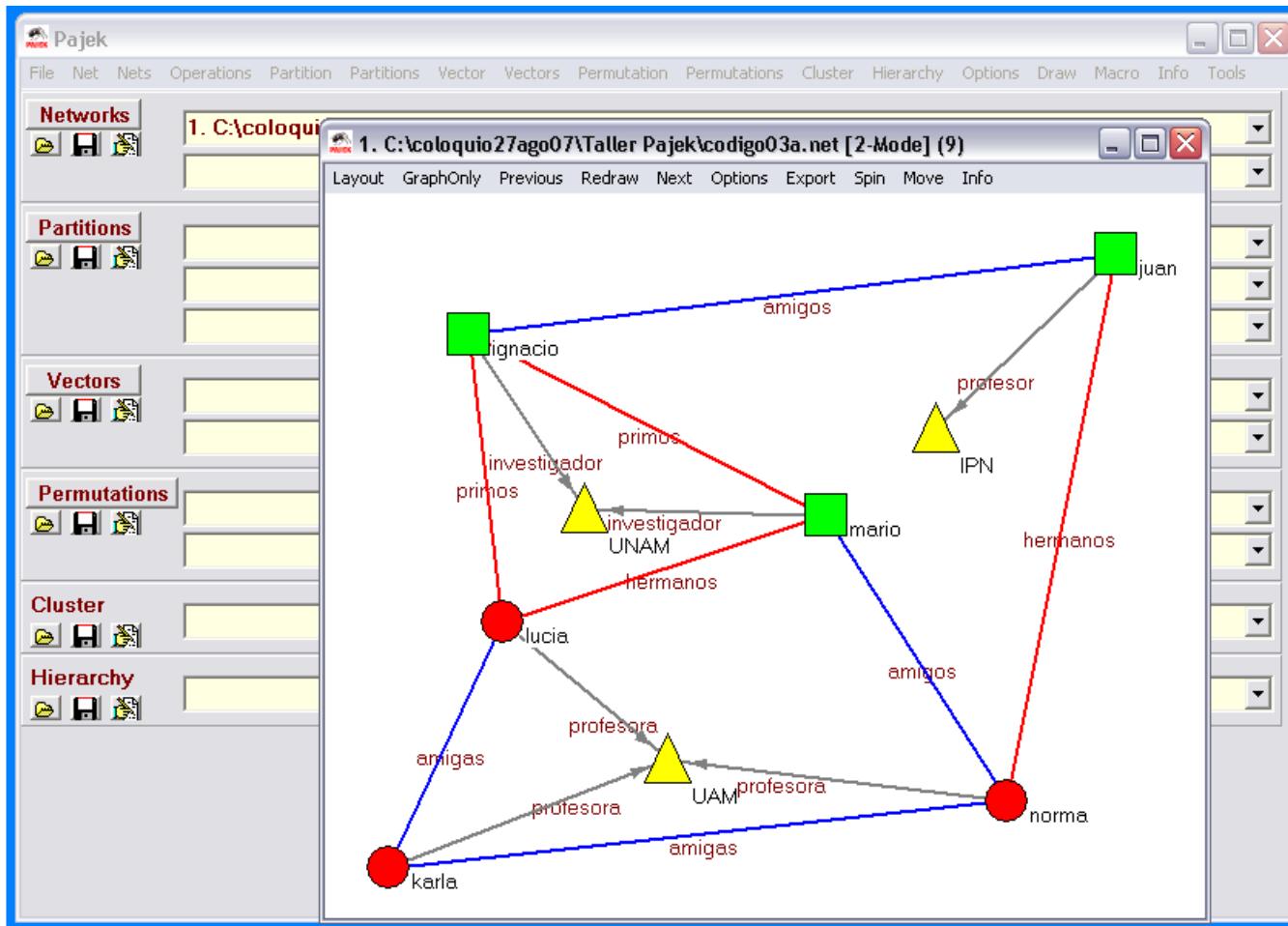
*Arcs :1 "afiliacion"
  1      8 1 1 "investigador"
  2      7 1 1 "profesor"
  3      9 1 1 "profesora"
  4      9 1 1 "profesora"
  5      8 1 1 "investigador"
  6      9 1 1 "profesora"

*Edges :2 "parentesco"
  1      4 1 c Red 1 "primos"
  1      5 1 c Red 1 "primos"
  2      6 1 c Red 1 "hermanos"
  4      5 1 c Red 1 "hermanos"

*Edges :3 "amistad"
  1      2 1 c Blue 1 "amigos"
  3      4 1 c Blue 1 "amigas"
  3      6 1 c Blue 1 "amigas"
  5      6 1 c Blue 1 "amigos"

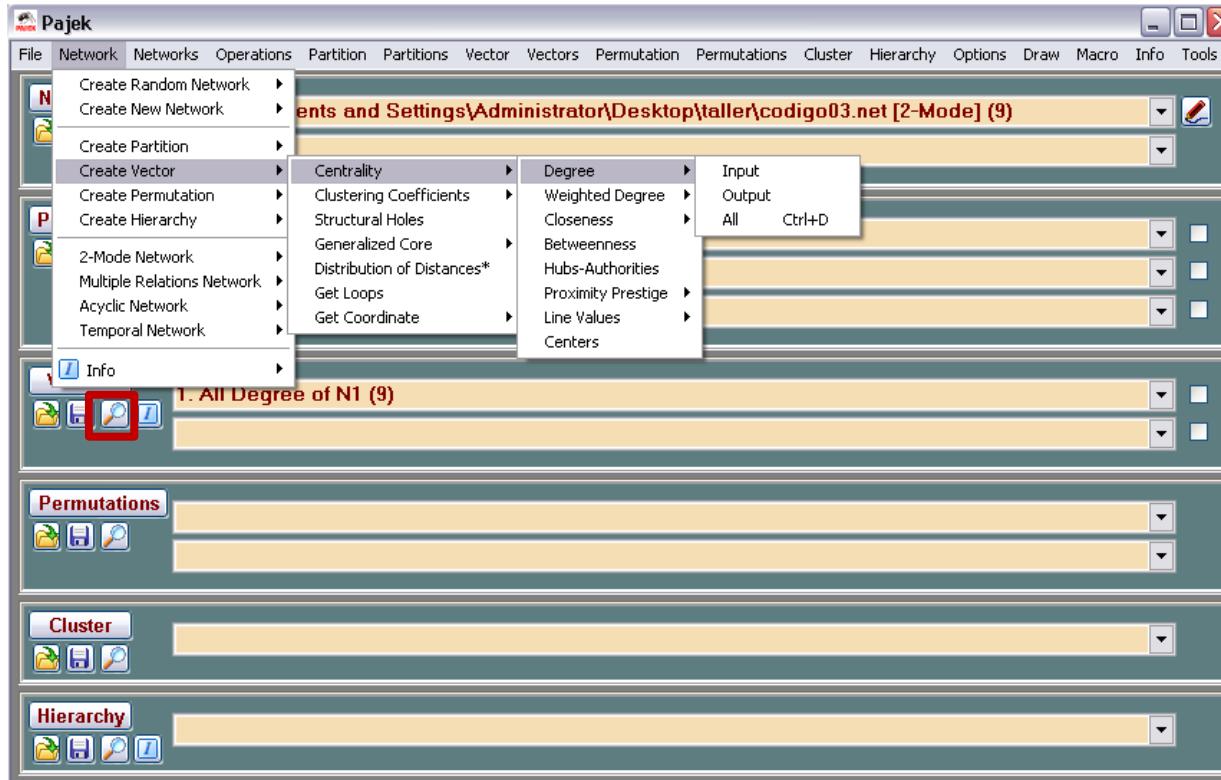
```

Pajek permite visualizar varias características de la red en una sola imagen, tanto relacionales como atributivas.



*Vertices n m *Vertices 9 6	n corresponde al número de nodos m corresponde donde empiezan la otra categoría cuando se trata de redes de afiliación (opción Draw-Partition para visualizar)
n₁ "Nombre" N1 N2 N3 forma ic Color 1 "ignacio" 0.1299 0.0845 0.5000 box ic Green 2 "juan" 0.8164 0.1026 0.5000 box ic Green 3 "karla" 0.5253 0.9386 0.5000 ellipse ic Red 4 "lucia" 0.0664 0.7379 0.5000 ellipse ic Red 5 "mario" 0.4189 0.5303 0.5000 box ic Green 6 "norma" 0.8184 0.5708 0.5000 ellipse ic Red 7 "IPN" 0.5873 0.3139 0.5000 triangle ic Yellow 8 "UNAM" 0.1735 0.4652 0.5000 triangle ic Yellow 9 "UAM" 0.4834 0.7139 0.5000 triangle ic Yellow	n₁ corresponde al número del nodo "Nombre" corresponde a la etiqueta del nodo N1 N2 N3 corresponden a las coordenadas forma (<i>ellipse, box, diamond, cross, empty</i>) ic indica que lo que sigue es la definición del color interno del nodo Color (<i>GreenYellow, Yellow ...</i>) un listado completo se muestra en el archivo crayola.pdf
*Arcs *Arcs :1 "afiliacion" i j val_arco c color p tipo_linea l "etiqueta" 1 8 1 1 "investigador" 2 7 1 1 "profesor" 3 9 1 1 "profesora" 4 9 1 1 "profesora" 5 8 1 1 "investigador" 6 9 1 1 "profesora"	Encabezado de la declaración de los arcos i número de nodo inicial j número de nodo final val_arco valor numérico del arco c indica que lo que sigue es el color del arco color del arco (crayola.pdf) p indica que lo que sigue es el tipo de línea tipo_linea (Solid, Dots) l (<i>label</i>) indica que lo que sigue es la etiqueta del arco "etiqueta" del arco
*Edges *Edges :2 "parentezco" i j val_arista c color p tipo_linea l "etiqueta" 1 4 1 c Red 1 "primos" 1 5 1 c Red 1 "primos" 2 6 1 c Red 1 "hermanos" 4 5 1 c Red 1 "hermanos"	Encabezado de la declaración de las aristas i número de nodo inicial j número de nodo final val_arista valor numérico de la arista c indica que lo que sigue es el color de la arista color de la arista (crayola.pdf) l (<i>label</i>) indica que lo que sigue es la etiqueta de la arista "etiqueta" de la arista

Centralidad



Bibliografía

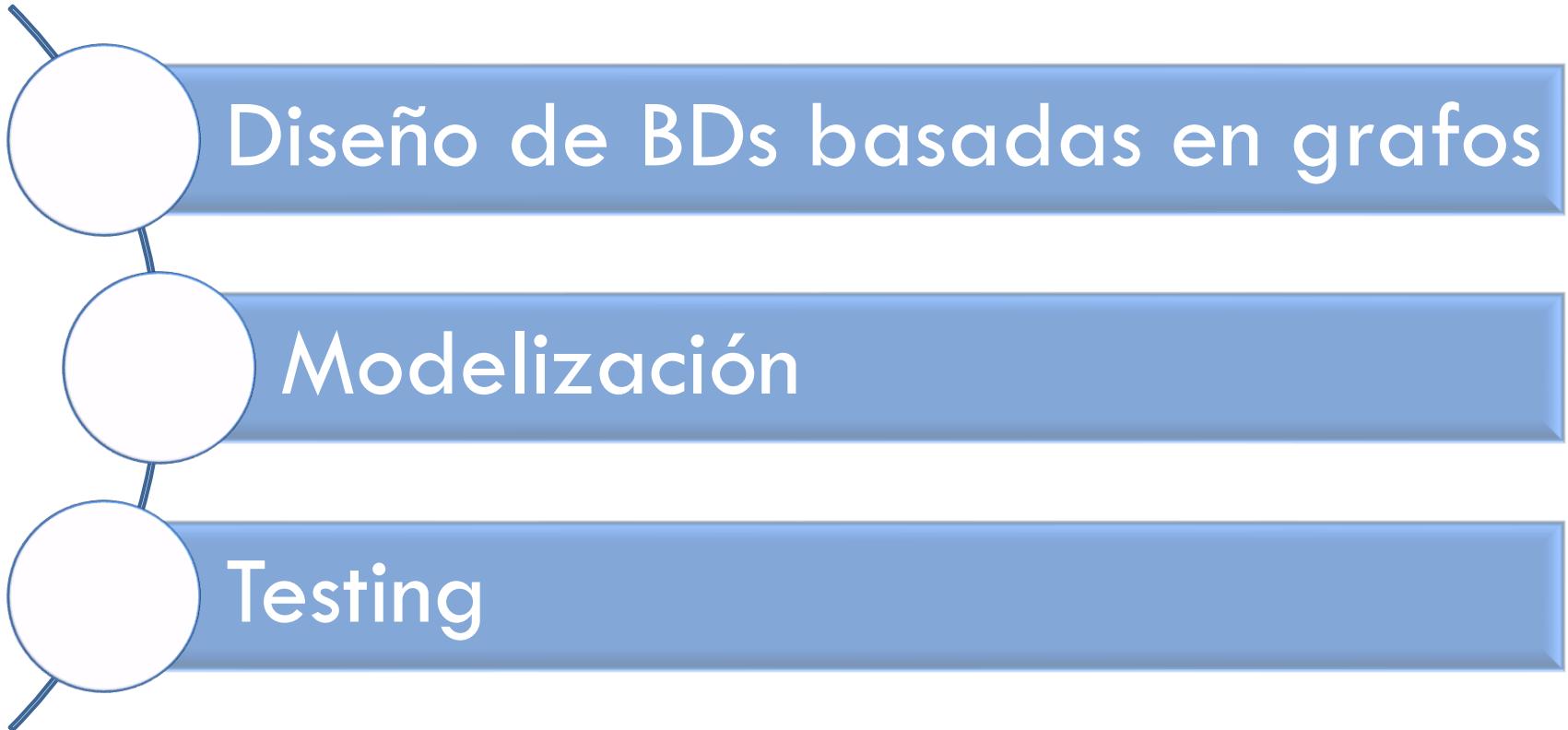
- NoSQL for mere mortals. Dan Sullivan. Pearson Education 2015.
- Visualización con Pajek. Alejandro A. Ruiz León. Nina Ines Jung. UNAM.

<http://harary.iimas.unam.mx/TallerPajek.pdf>

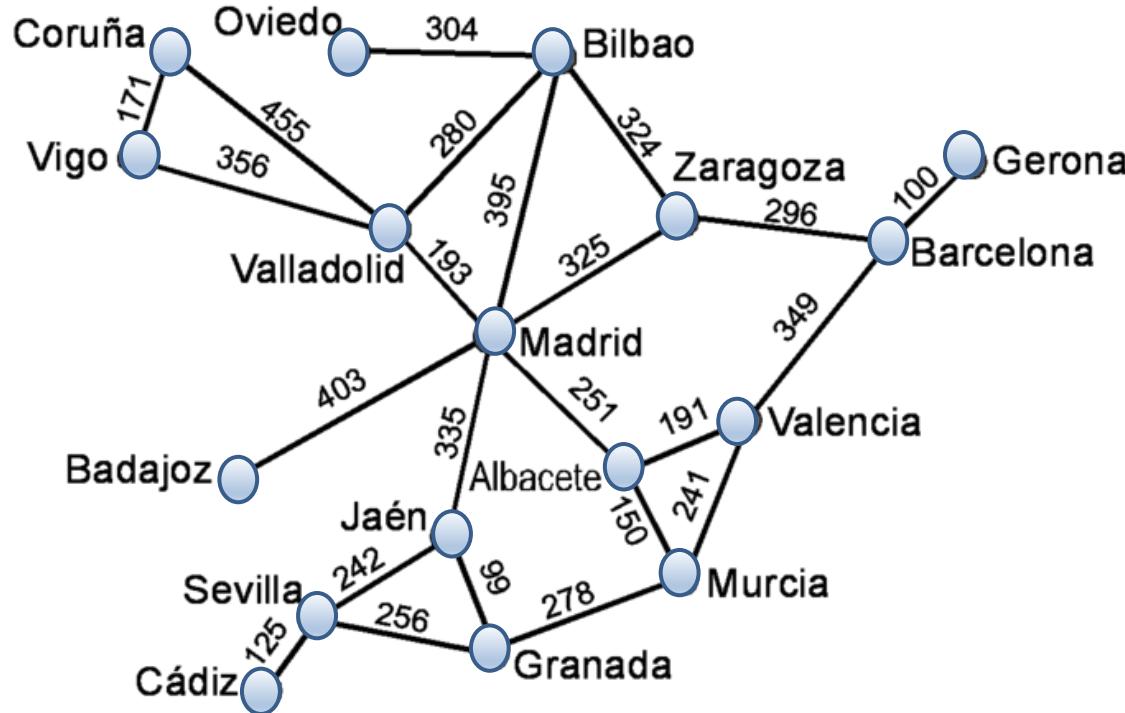
Bases de datos orientadas a grafos

TEMA 2. Diseño

Contenido



Diseño de BDs basadas en grafos



Ejemplo: red social de desarrolladores NoSQL



Dos entidades

Desarrolladores

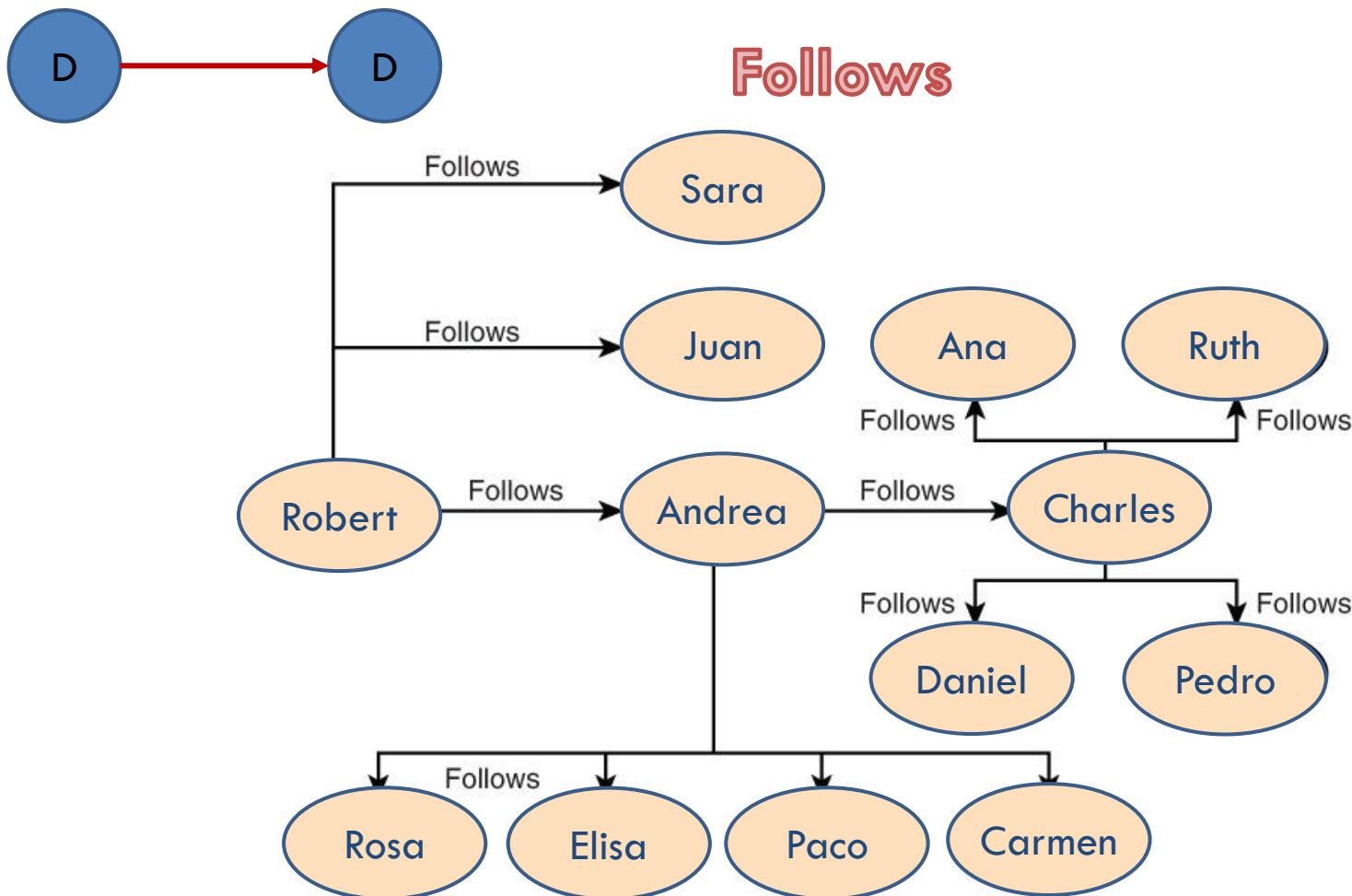
- Nombre
- Ciudad
- BD NoSQL usadas
- Años de experiencia
- Áreas de interés

Posts

- Fecha de creación
- Topics
- Tipo de post (pregunta, tip, noticia)
- Título
- Texto del post

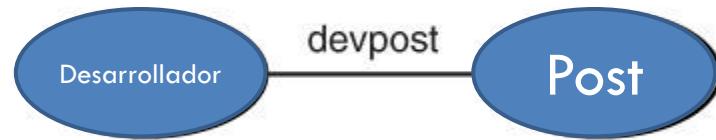
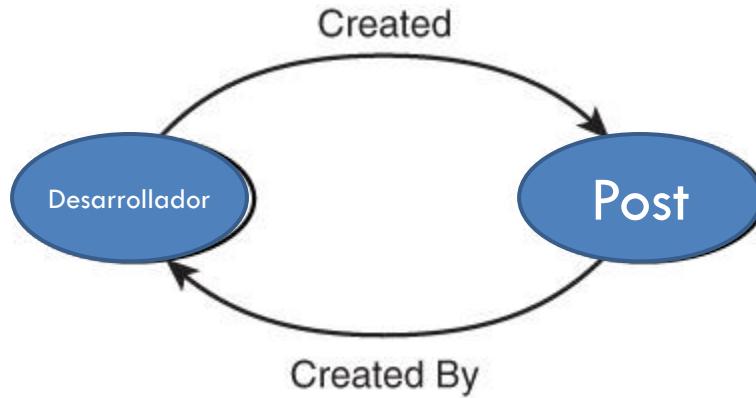
Posibles relaciones





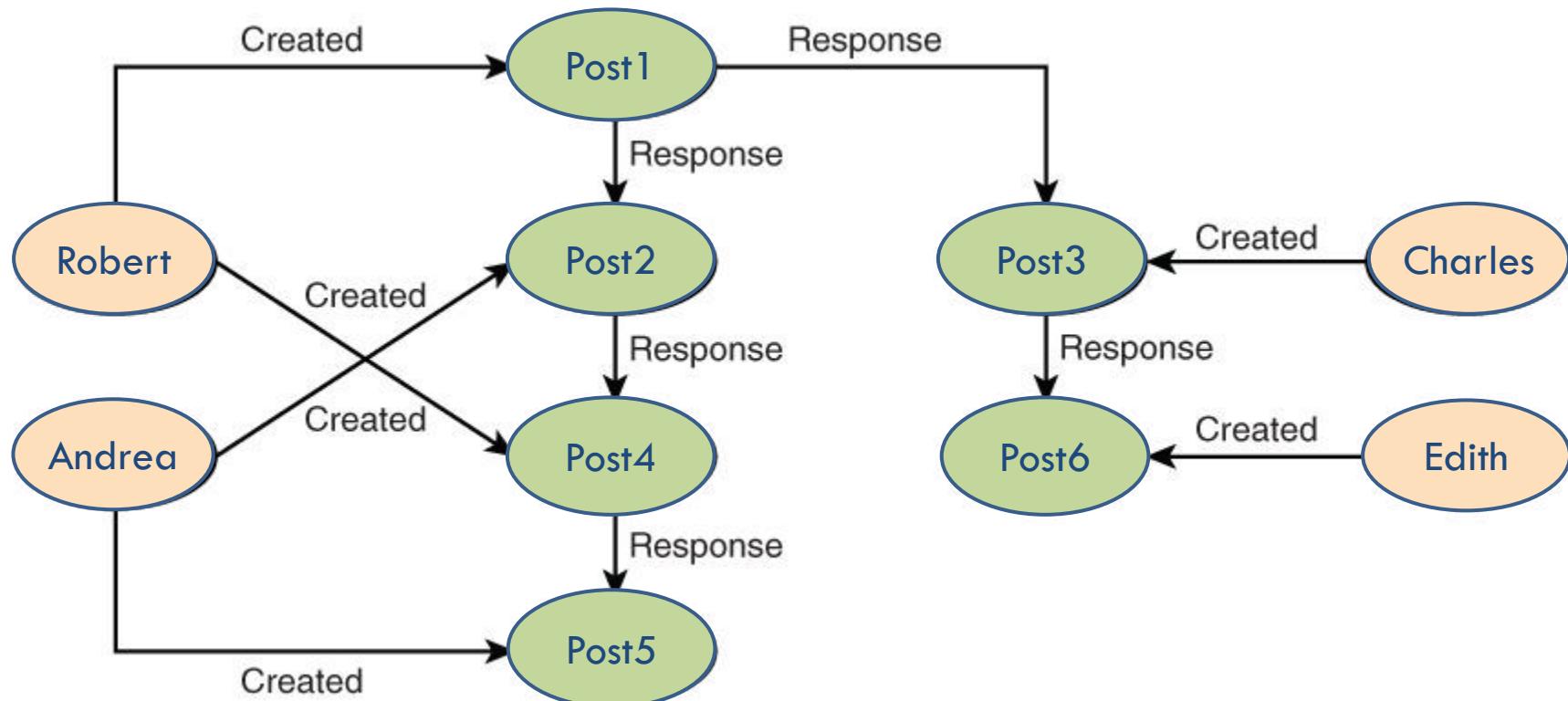


Created





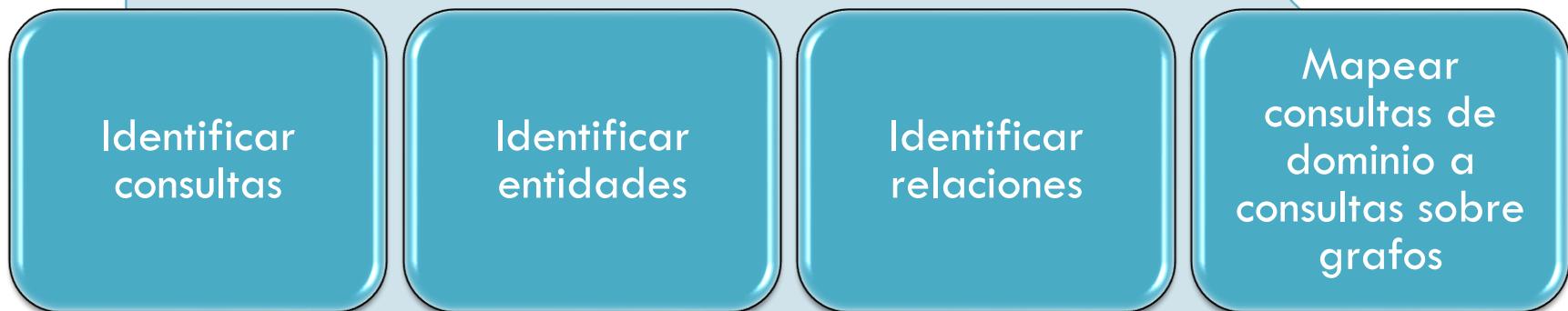
Response

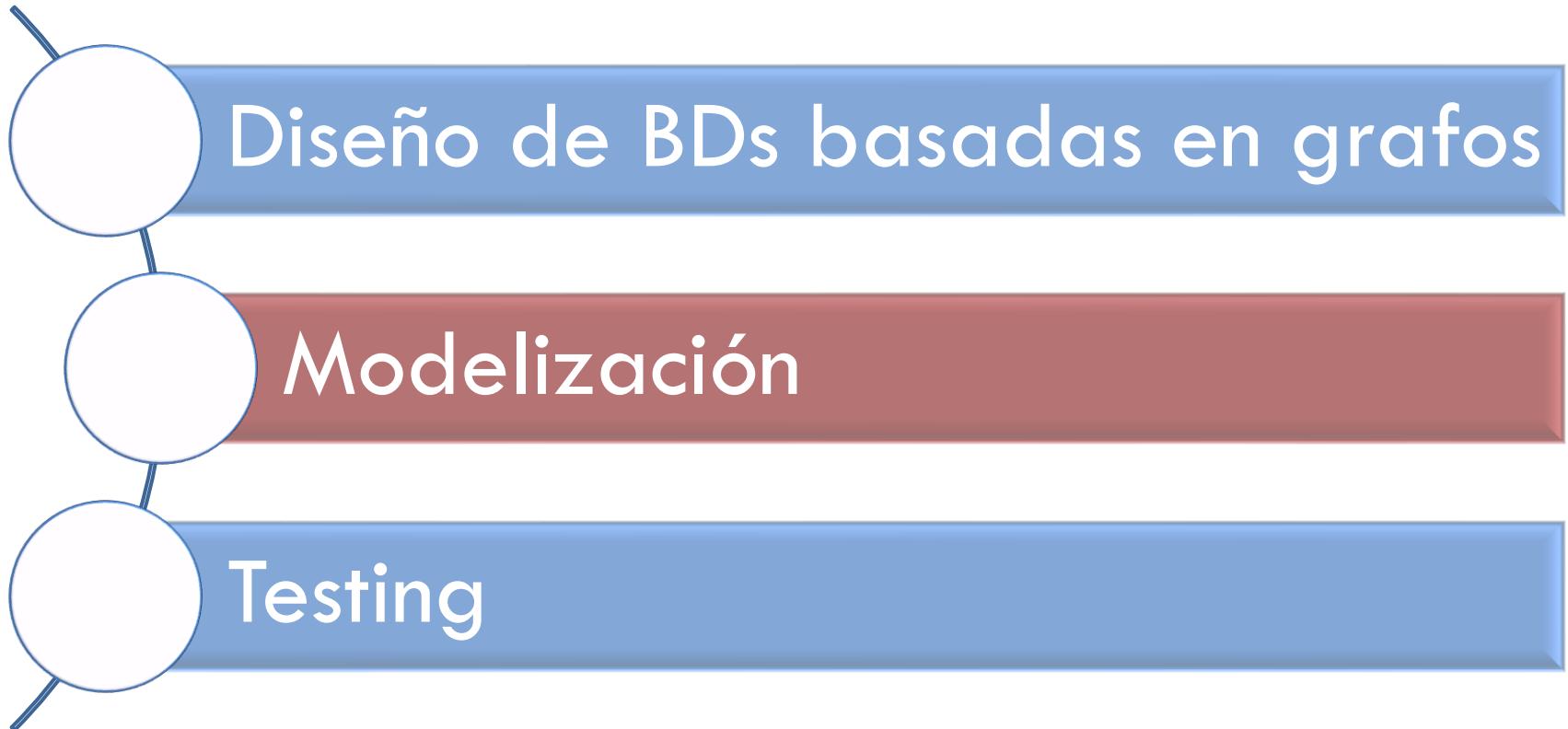


Consultas

Consultas grafos	Consultas específicas dominio
¿Cuántos saltos (arcos) hacen falta para ir del vértice A al vértice B	¿Cuántas relaciones “follows” hay entre el desarrollador A y el desarrollador B?
¿Cuántos arcos entrantes inciden en el vértice A?	¿Cuántos desarrolladores siguen a Andrea?
¿Cuál es la centralidad del vértice B?	¿Cómo de bien conectada está Edith en la red social?
¿Es C un cuello de botella? Es decir, si se elimina el nodo C ¿qué partes del grafo se desconectan?	Si un desarrollador abandona la red social ¿habría grupos de desarrolladores desconectados?

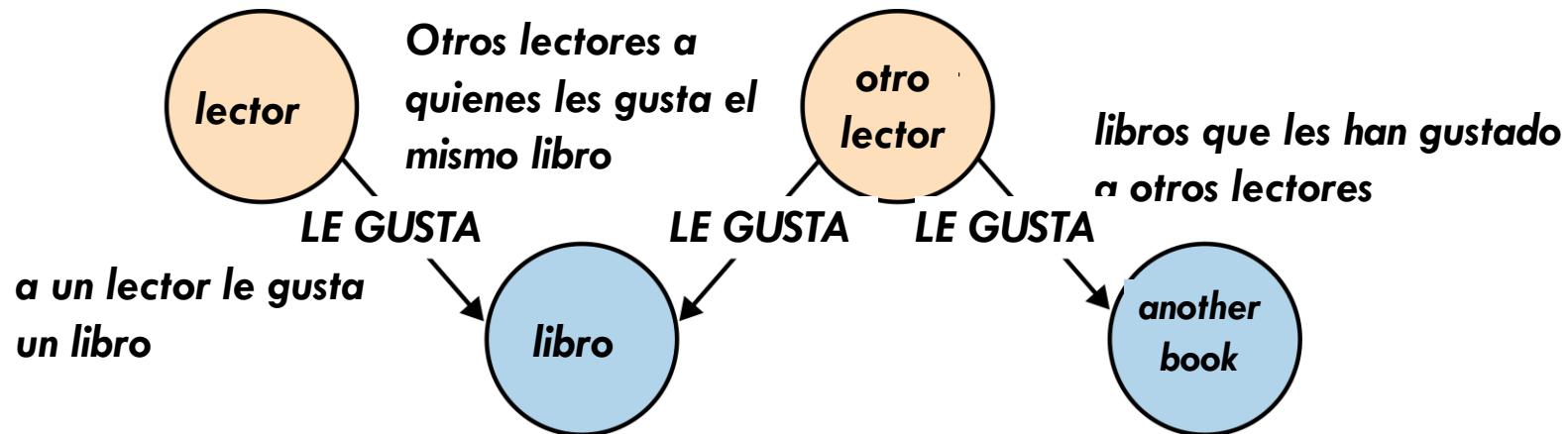
Pasos





Describir el modelo en términos de las necesidades de la aplicación

“COMO lector al que le gusta un libro, QUIERO saber a qué otros lectores les gusta el mismo libro PARA poder encontrar otros libros que leer”



Guidelines

Nodos



Entidades

Relaciones



Conexiones

Dirección de una relación



Clarificar semántica

Propiedades de los nodos



Atributos de las entidades

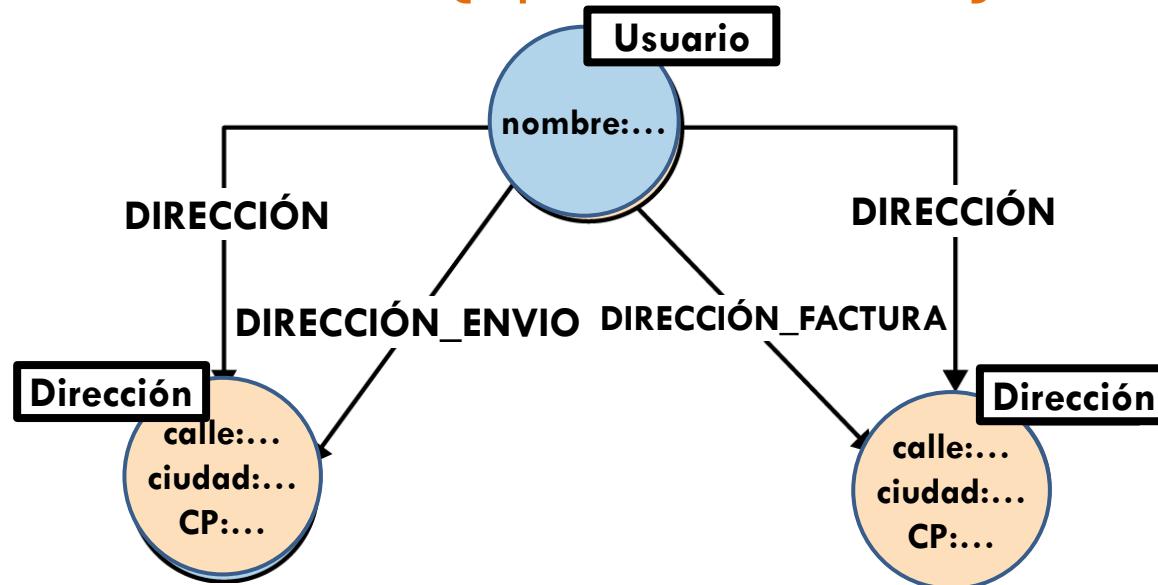
Propiedades de las relaciones



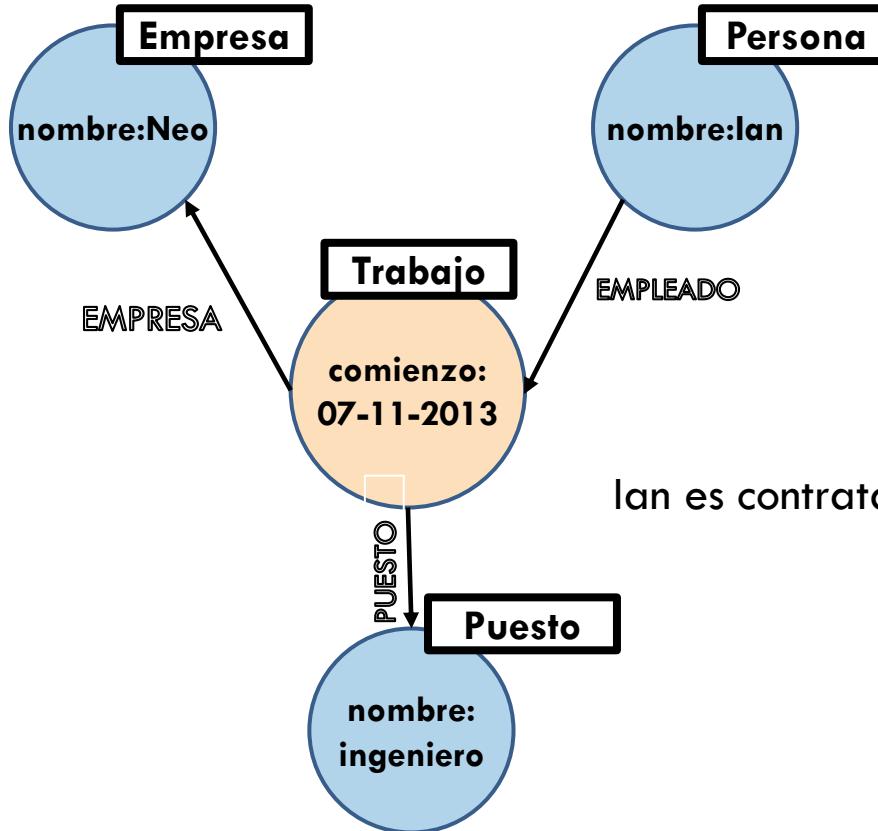
Fuerza, peso de las relaciones

Relaciones detalladas o genéricas

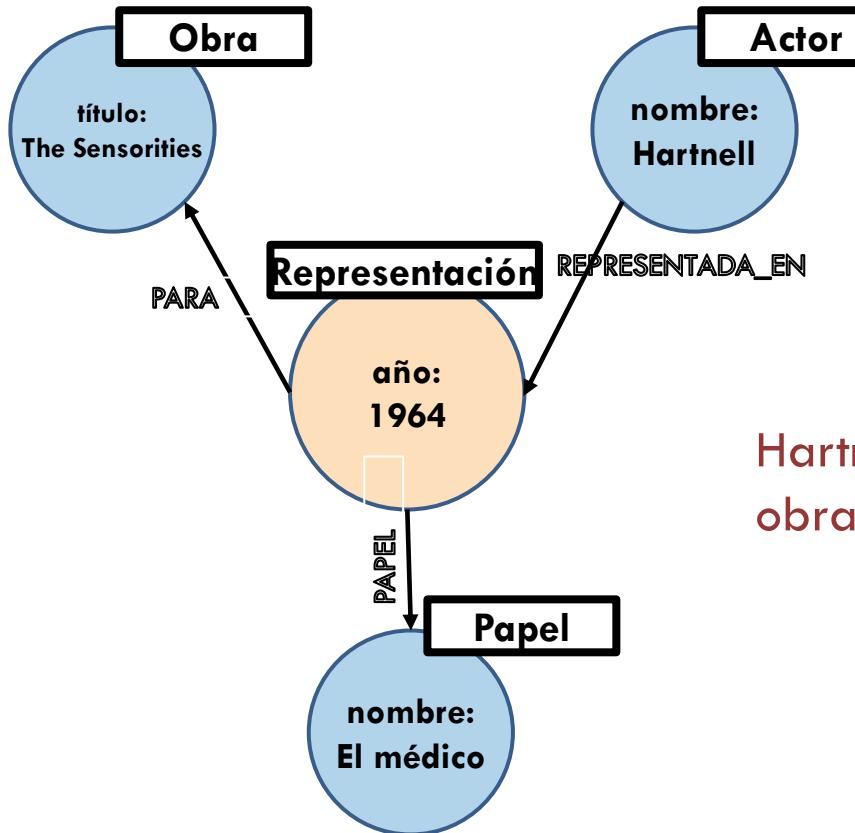
- Opción 1: DIRECCION_ENVIO y DIRECCION_FACTURACION
- Opción 2: DIRECCION {tipo:'envio'} y
DIRECCION {tipo:'facturacion'}



Modelizar hechos

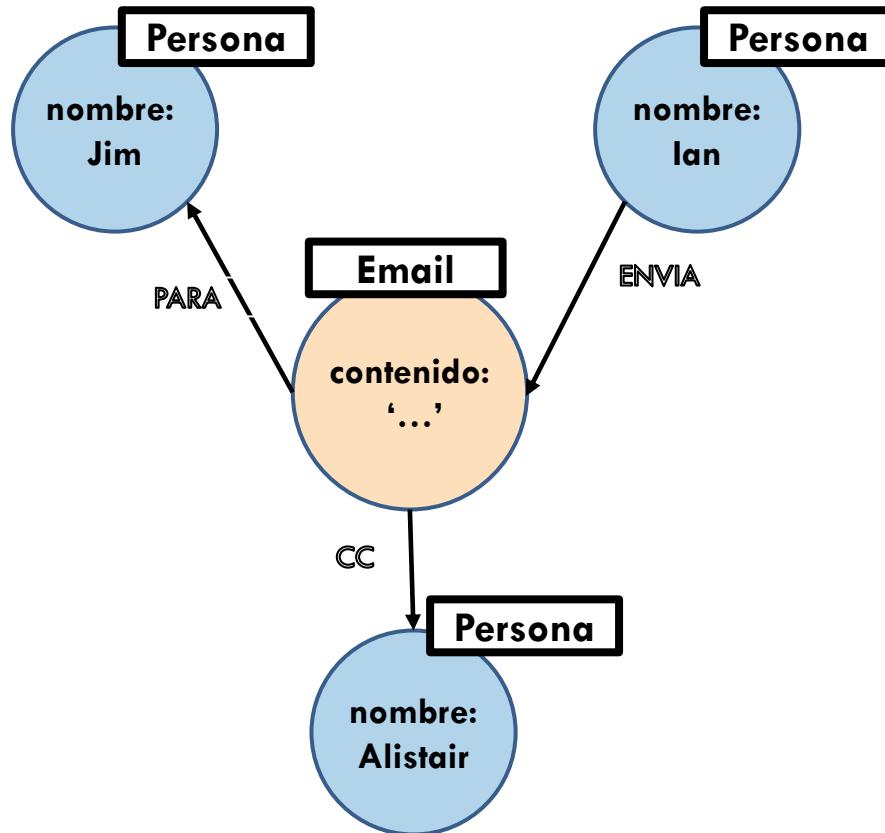


Modelizar hechos (JJ)



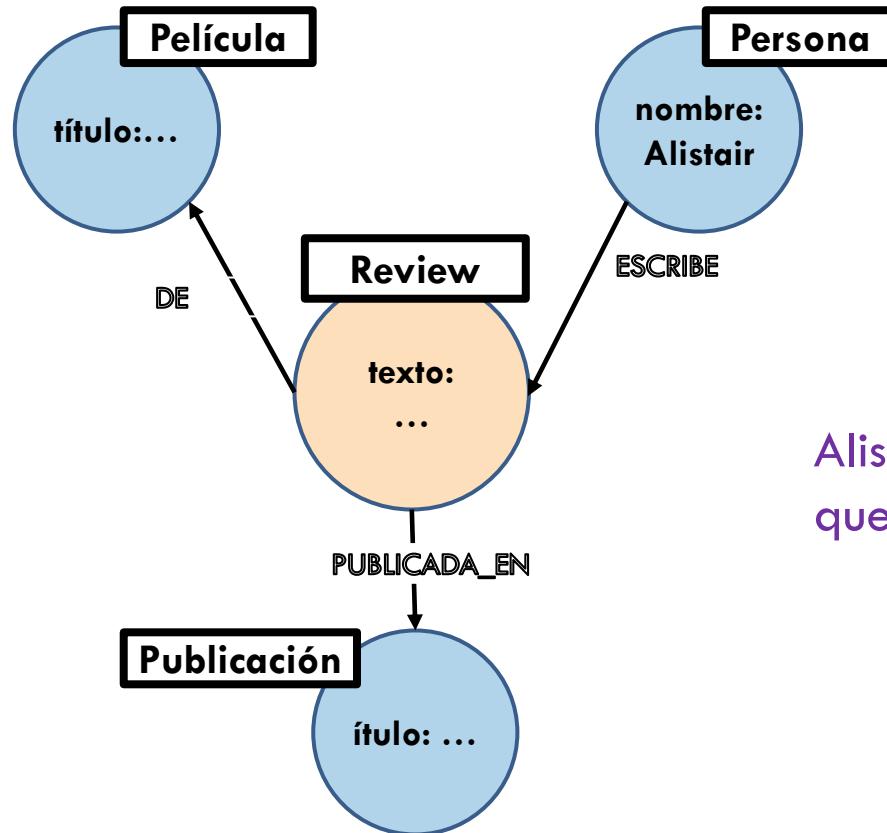
Hartnell representa el papel de doctor en la obra The Sensorities

Modelizar hechos (III)



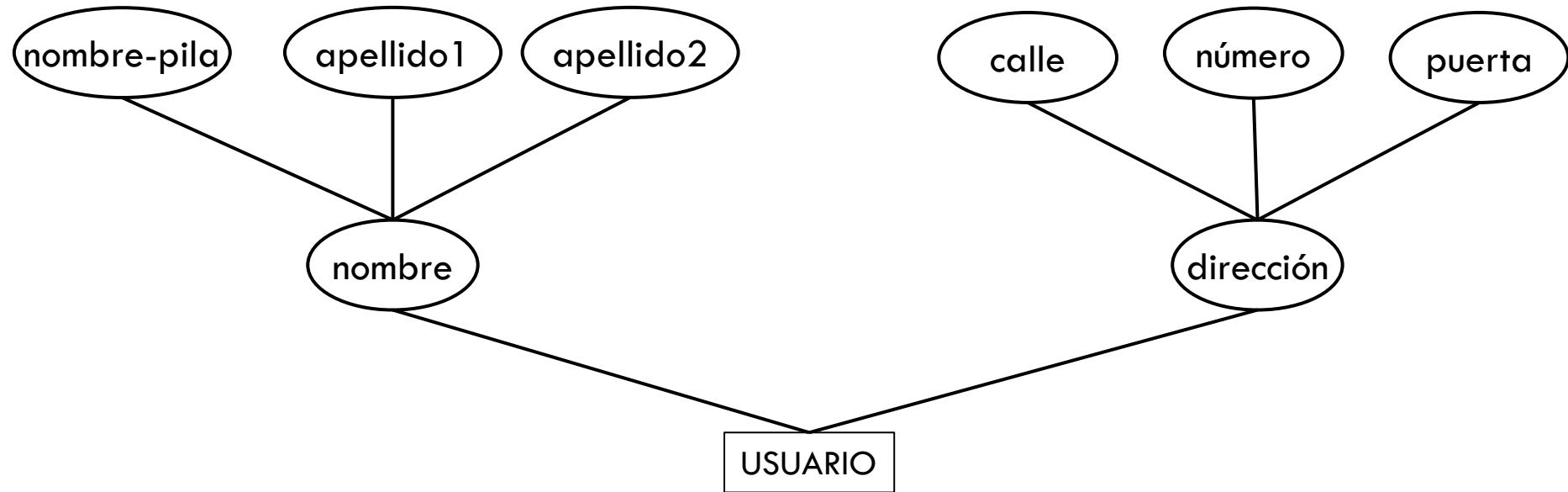
Ian envía un correo a Jim y copia a Alistair

Modelizar hechos (JV)

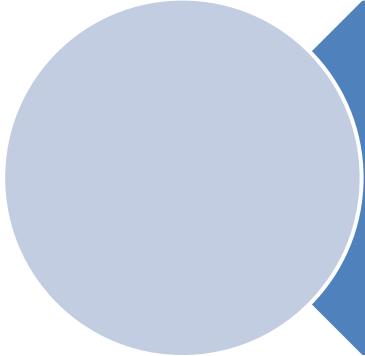


Alistair escribió review de una película que fue publicada por una revista

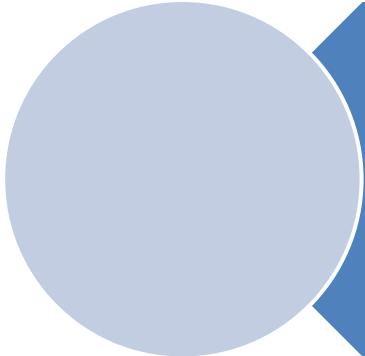
Representar atributos compuestos como nodos



Cómo representar el tiempo

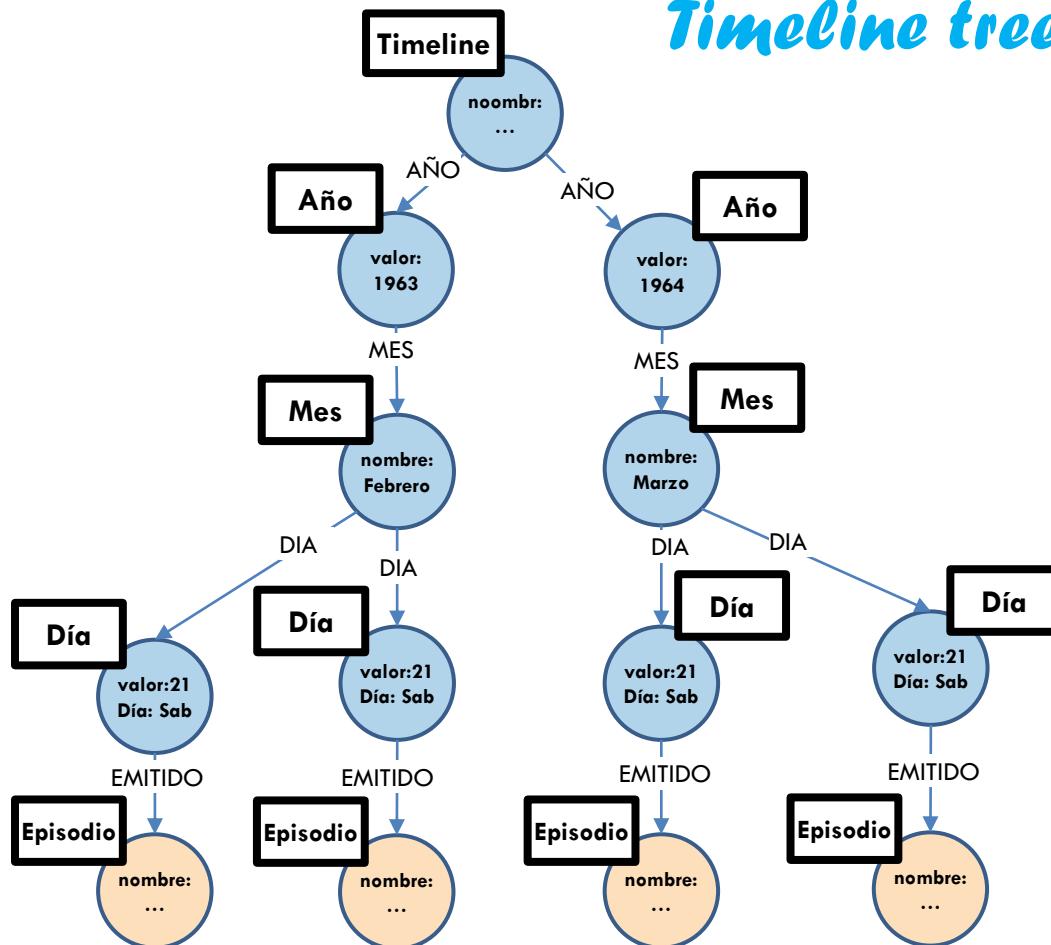


Timeline trees



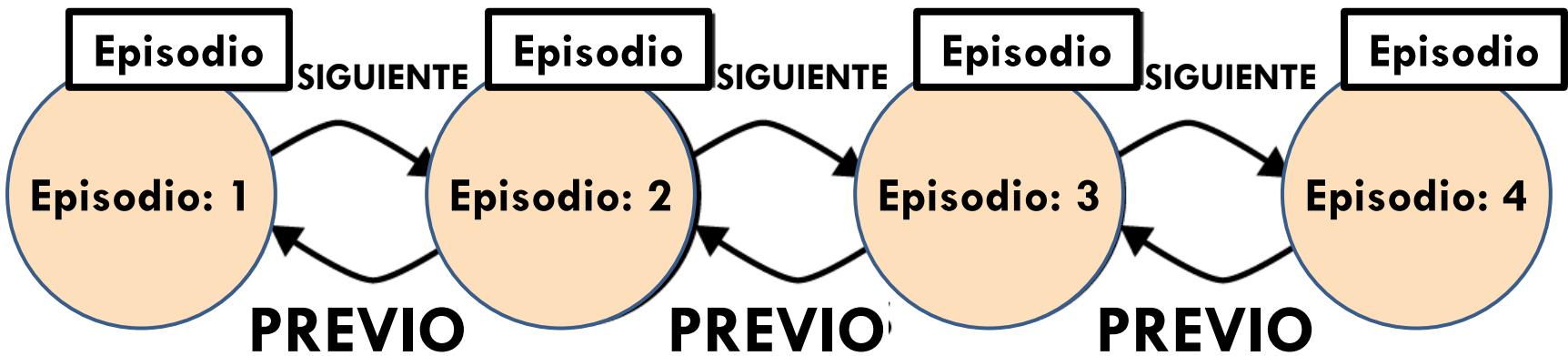
Listas
enlazadas

Timeline trees

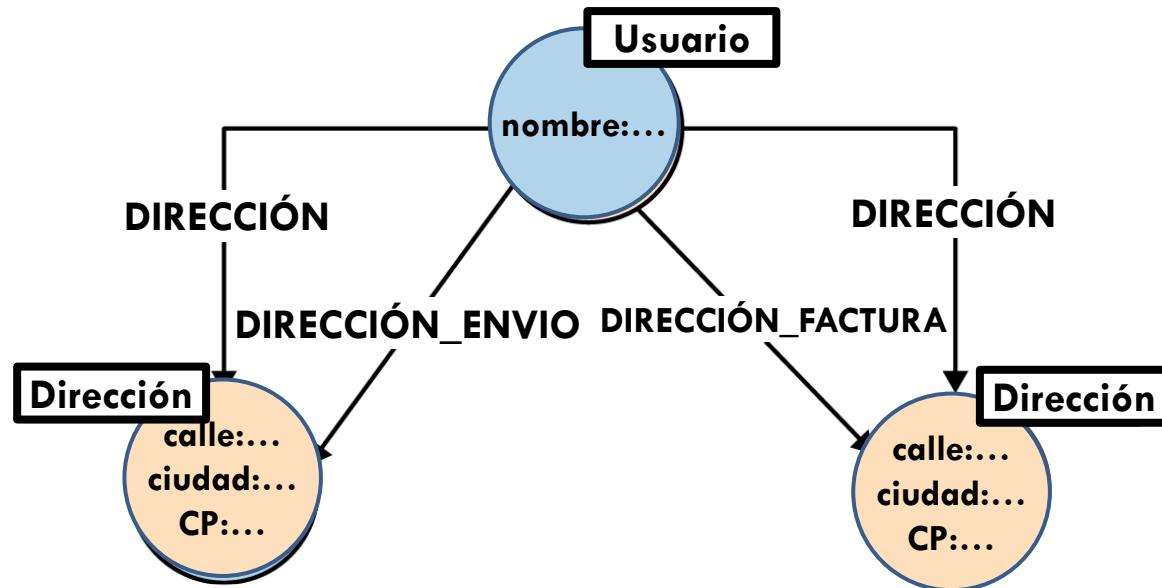


Fechas de emisión de cuatro episodios de un programa de televisión

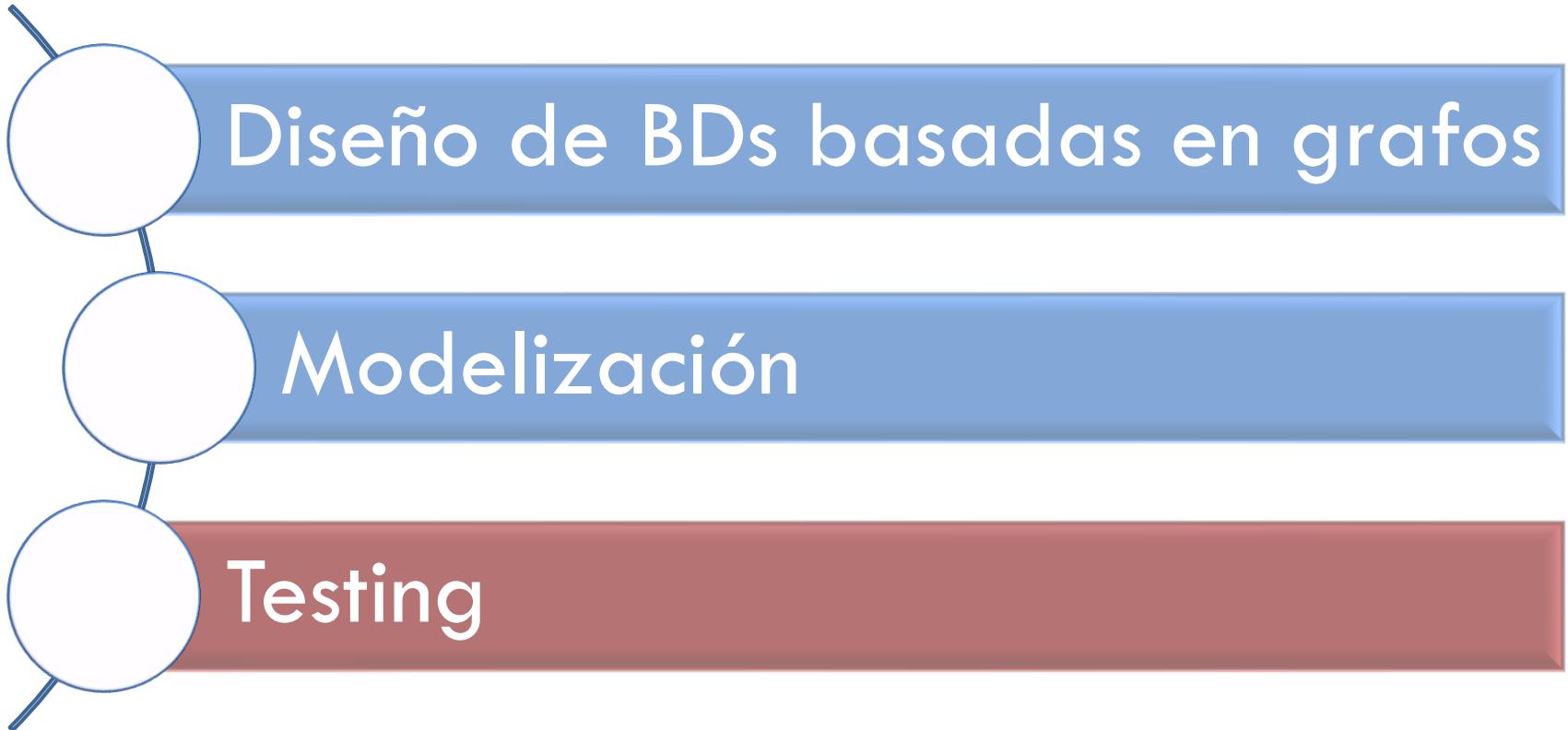
Listas enlazadas



Desarrollo iterativo e incremental



Contenido



Testing

1. Desarrollo del modelo de datos basado en el testing

2. Test de rendimiento

- Test de consultas
- Test de aplicación

Desarrollo del modelo de datos basado en el testing

Test unitarios

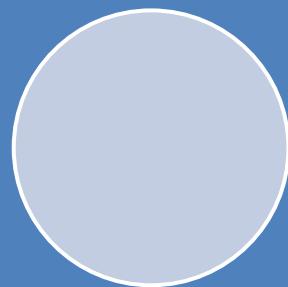
Normal  Excepcional

Documentación

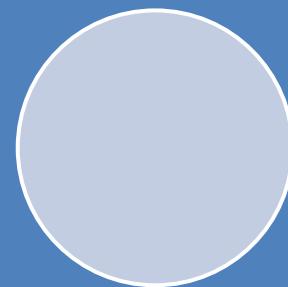
Un test = Un aspecto del dominio

Nuevas funcionalidades

Test de rendimiento



De consultas



De aplicación



Test de rendimiento de consultas

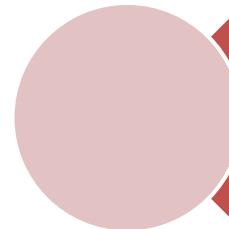


Crear tests y
guardar datos
de rendimiento

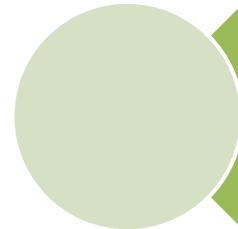
Ejecutar tests
con frecuencia

Ejecutar una
consulta
muchas veces

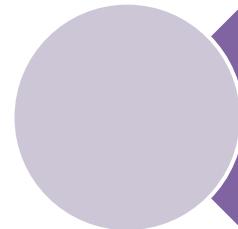
Test de rendimiento de aplicación



Aplicación completa



Escenarios de uso
representativos



En paralelo con desarrollo
de características

Testing con datos representativos

De un tercero o
adaptar existente =>
importar

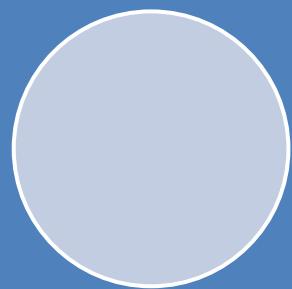
Partir de cero =>
constructor de datasets

- Incremental
- Reproducir todas las características identificadas
- No excesivamente grandes

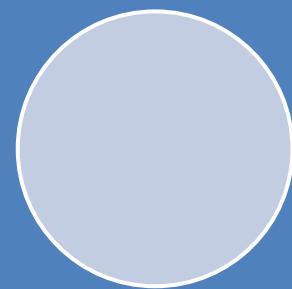
Neo4j

Estrella Pulido

Ejemplos de uso



Ebay



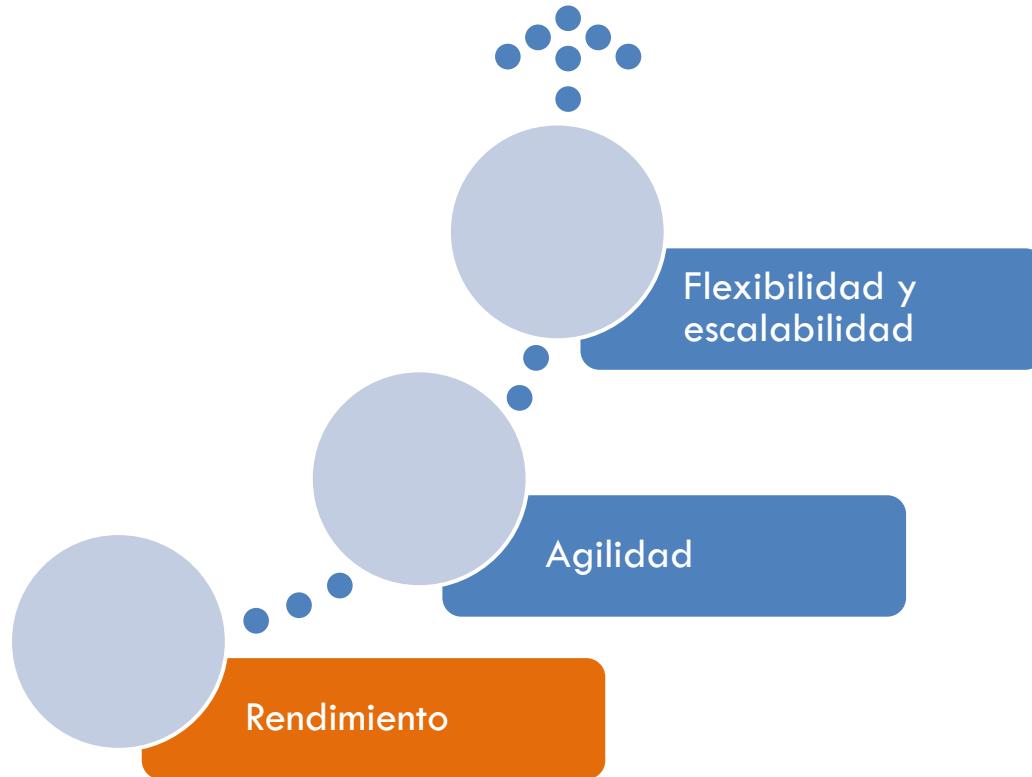
Walmart



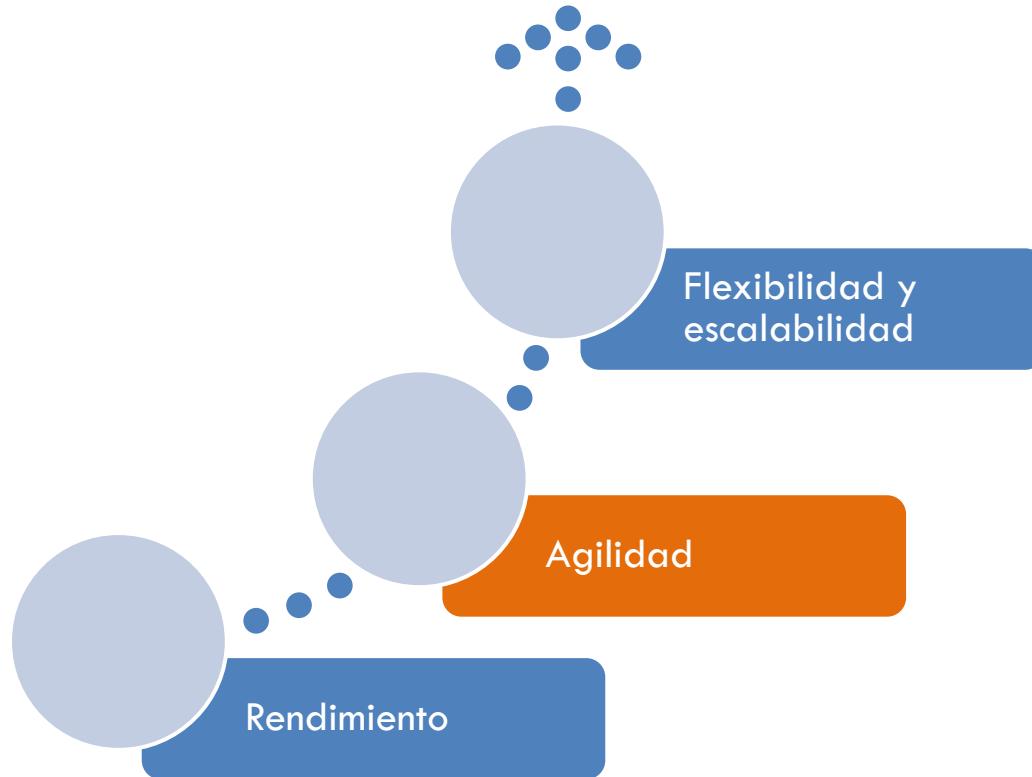
¿Qué grafos utiliza?

- Grafos dirigidos
- Grafos no dirigidos
- Grafos ponderados
- Grafos con etiquetas
- **Grafos de propiedad**

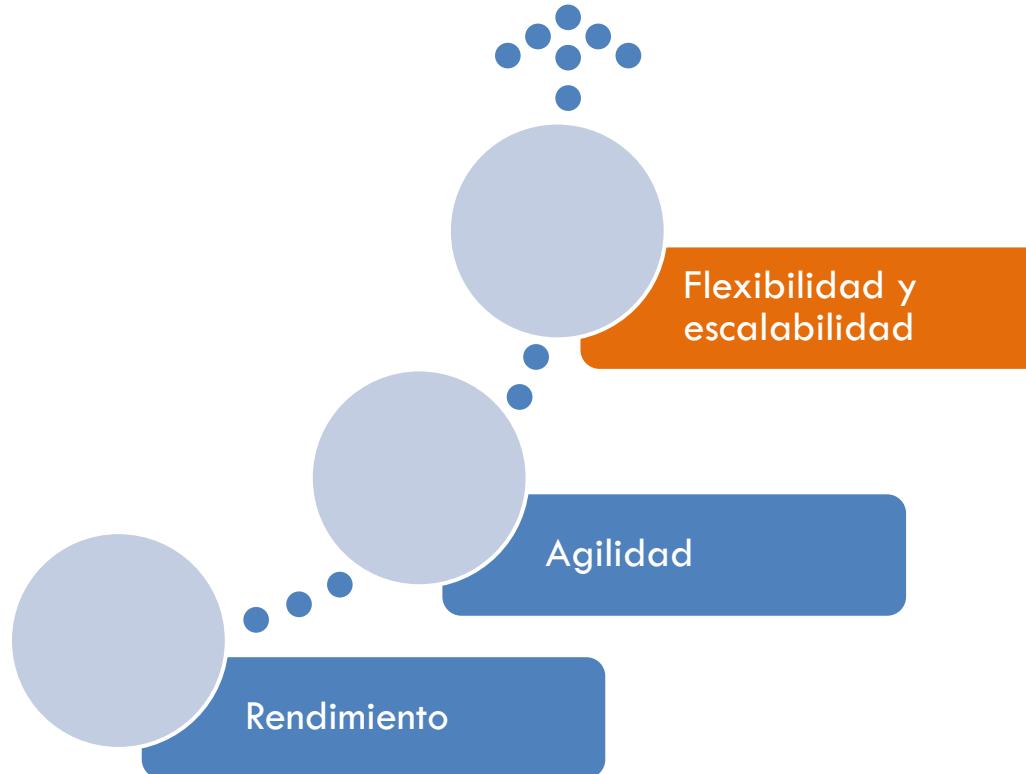
Ventajas



Ventajas



Ventajas



Dos casos de uso

Detección de fraude

Recomendaciones en tiempo real y redes sociales

Dos casos de uso

Detección de fraude

Recomendaciones en tiempo real y redes sociales

BD relacional <-> BD basada en grafos

BD relacional	BD basada en grafos
Tablas	Grafos
Filas	Nodos
Columnas y datos	Propiedades y valores
Restricciones	Relaciones
Joins	Recorrido de grafos

Capacidad

Versión 2.1.3

Elemento de Neo4j	Capacidad
Nodos	Alrededor de 35 billones
Relaciones	Alrededor de 35 billones
Etiquetas	Alrededor de 275 billones

La versión 3.0 admite grafos de cualquier tamaño

Características

- Lenguaje de consulta CQL
- Modelo de datos: grafo de propiedades
- Indices (Apache Lucene)
- Restricciones UNIQUE
- Interfaz para ejecutar comandos CQL
- ACID
- Almacenamiento de grafos de forma nativa
- Accesible desde Java, Spring, Scala, ...

Ventajas de CQL

Fácil representación de datos

Recuperación / Traversal /Navegación: fácil y rápido

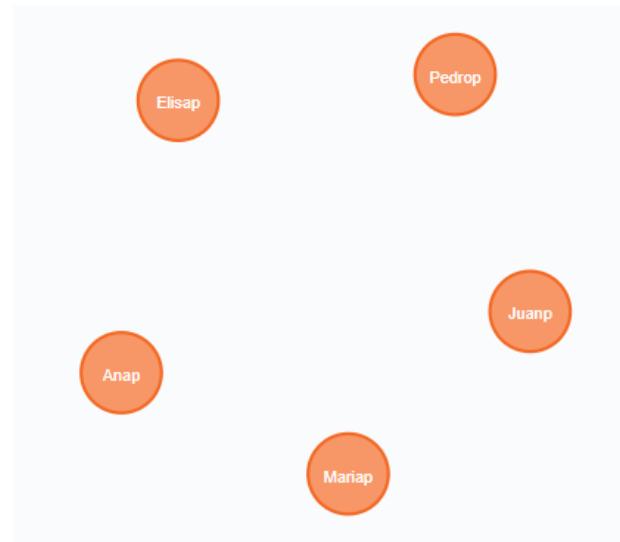
CQL: fácil de aprender

Modelo de datos simple y potente

No requiere joins

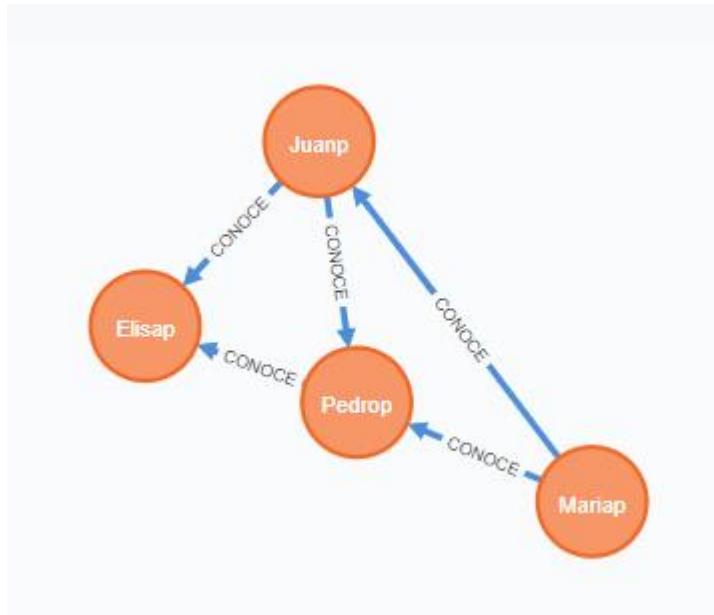
Nodos, propiedades, etiquetas

- Nodos ≈ registros
- Propiedades ≈ pares nombre/valor
 - Cadenas, números o booleanos
- Etiquetas
 - Forma de agrupar nodos
 - Ejemplo: Persona

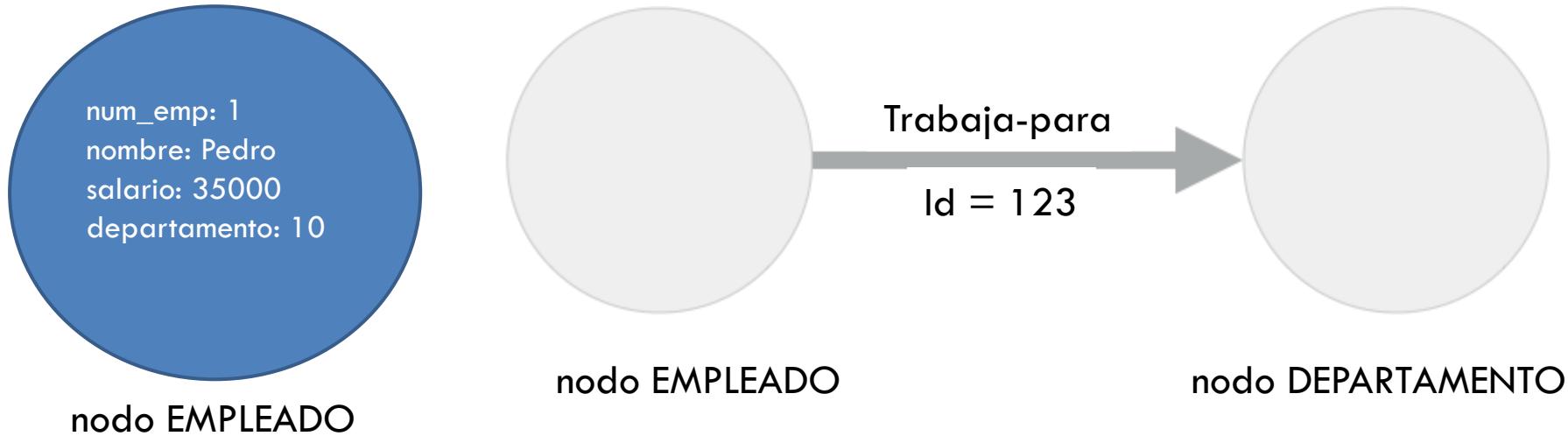


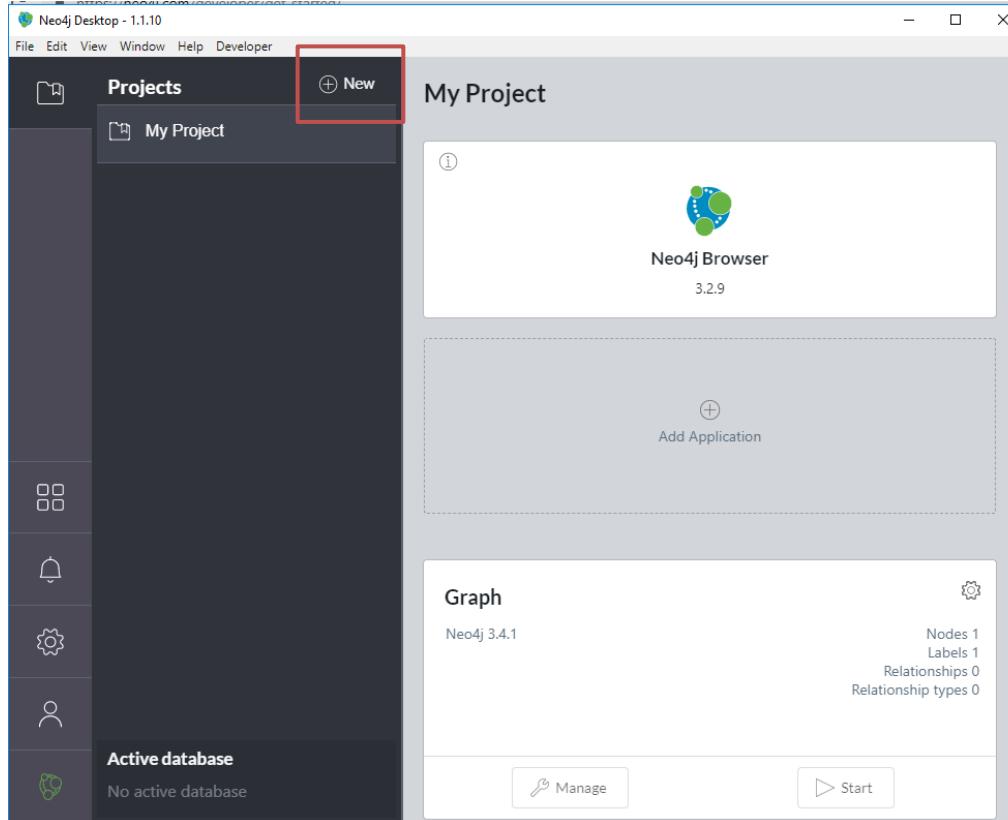
Relaciones

- Tienen dirección
- Tienen tipo
- Tienen propiedades

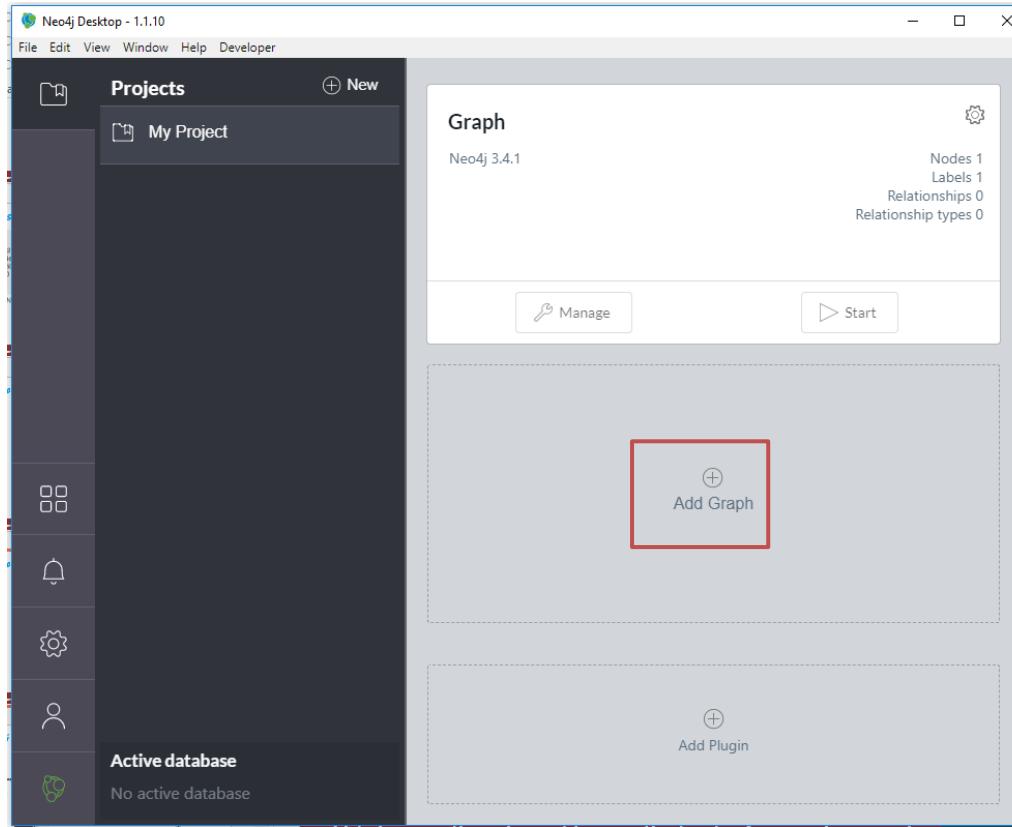


Nodos, propiedades, relaciones y etiquetas

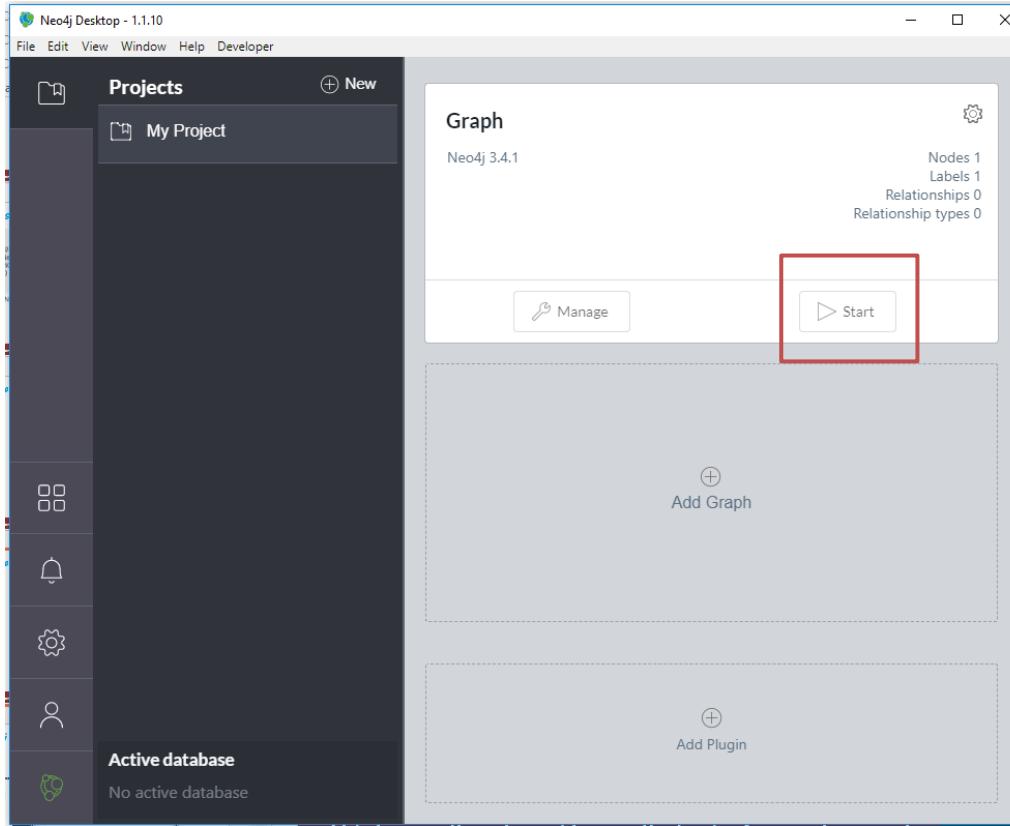




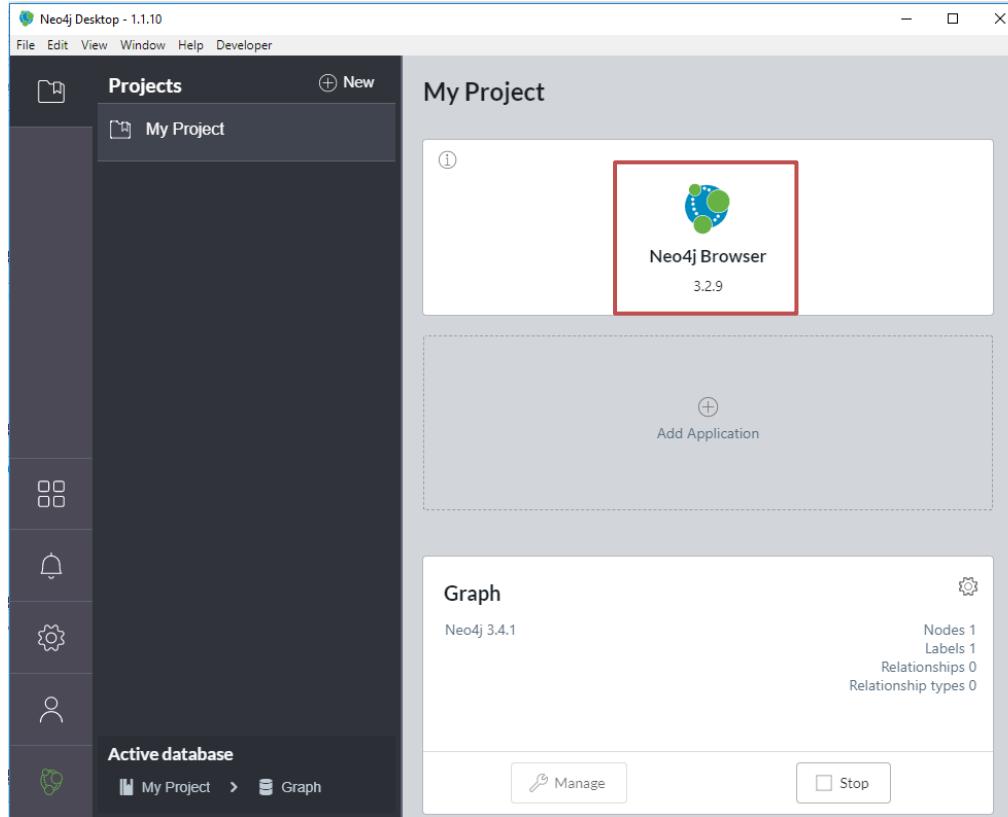
Crear base de datos



Arrancar la base de datos



Arrancar Neo4j Browser



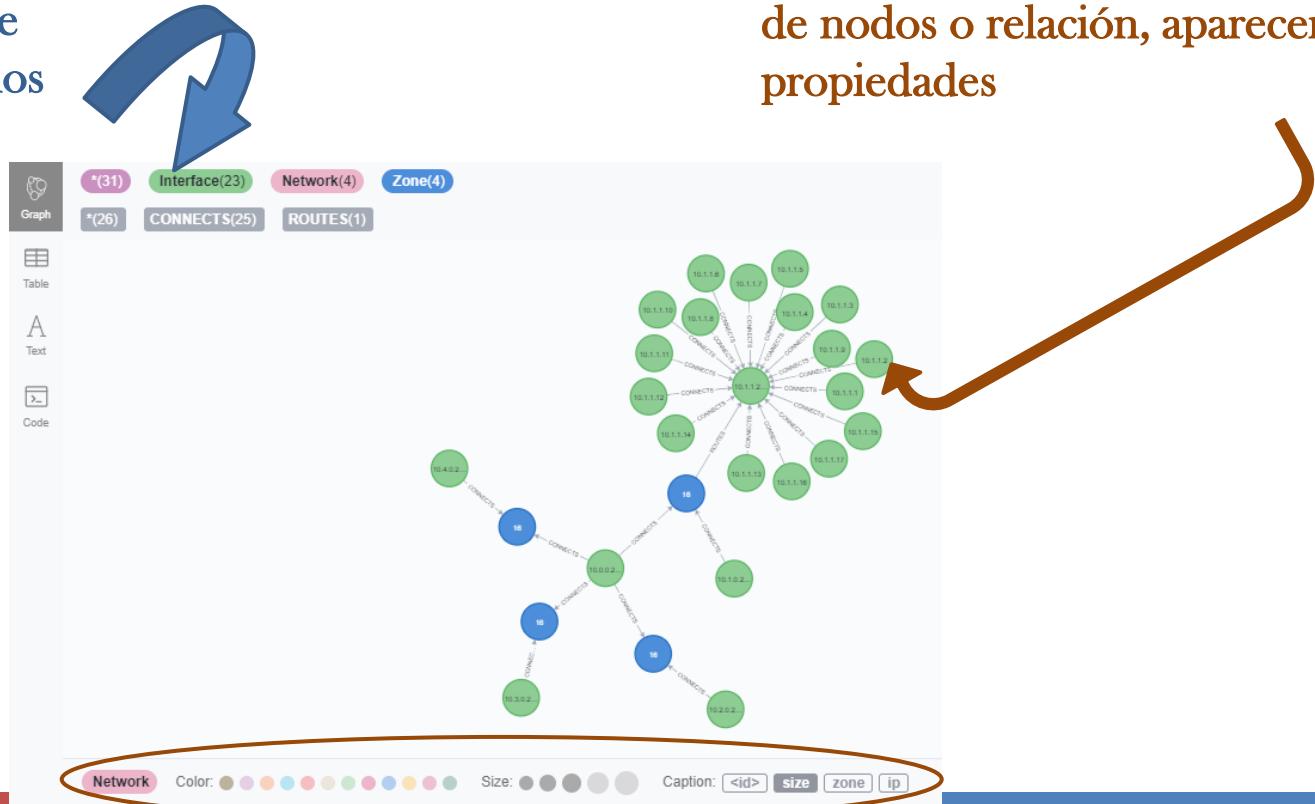
El Browser de Neo4j

Para ejecutar



Vista como grafo

Si pinchamos en etiqueta de nodo o de relación, podemos cambiar color, tamaño y leyenda



Si pasamos cursor por encima de nodos o relación, aparecen propiedades

Lenguaje de consulta: Cypher (CQL)

Usa patrones para describir datos

Lenguaje de pattern-matching

Sintaxis sencilla y legible

Cláusulas parecidas a SQL

Comandos para insertar, borrar, ...

Cláusulas como WHERE, ORDER BY, ...

Funciones para cadenas, funciones agregadas, ...

Comandos y cláusulas

Comando CQL	Uso
CREATE	Crear nodos, propiedades y relaciones
MATCH	Recuperar datos sobre nodos, propiedades y relaciones
RETURN	Devolver resultados de consultas
WHERE	Establecer condiciones para filtrar datos
DELETE	Borrar nodos y relaciones
REMOVE	Borrar propiedades de nodos y relaciones
ORDER BY	Ordenar resultados
SET	Añadir o actualizar etiquetas

Funciones

Función CQL	Uso
Cadenas	Trabajar con cadenas de caracteres
Agregación	Aplicar operaciones de agregación a resultados de consulta CWL
Relaciones	Obtener detalles de las relaciones como el nodo origen y destino

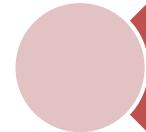
Tipos de datos

Tipo de datos CQL	Uso
boolean	Literales booleanos: true o false
byte	Enteros de 8 bits
short	Enteros de 16 bits
Int	Enteros de 32 bits
long	Enteros de 64 bits
float	Punto flotante de 32 bits
Double	Punto flotante de 64 bits
Char	Caracteres de 16 bits
String	Cadenas de caracteres

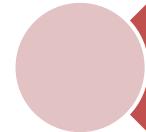
El comando CREATE



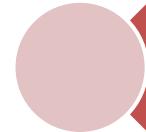
Nodos con propiedades



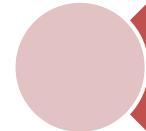
Nodos sin propiedades



Relaciones con propiedades



Relaciones sin propiedades



Etiquetas para nodos y relaciones

CREATE (<nombre-nodo>:<nombre-etiqueta>)

The screenshot shows the Neo4j Browser interface. In the top-left query editor, the command `$ CREATE (emp: EMPLEADO)` is entered. To the right of the editor are three small icons: a star, a refresh, and a play button. The play button is circled in red. On the far left, there is a vertical sidebar labeled "Code" with a code editor icon. The main workspace below the editor is currently empty. At the bottom of the workspace, a message reads: "Added 1 label, created 1 node, completed after 5 ms.".

\$ CREATE (emp: EMPLEADO)

Boton de ejecución

Code

Added 1 label, created 1 node, completed after 5 ms.

```
CREATE (
    <nombre-nodo>:<nombre-etiqueta>
{
    <nombre-propiedad1>:<valor-propiedad1>,
    .....
    <nombre-propiedadn>:<valor-propiedadn>
}
)
```

```
$ CREATE (e1: Empleado {id:23, nombre:'Juan González', salario: 35000, deptno: 10})
```



Added 1 label, created 1 node, set 4 properties, completed after 9 ms.



Table



Code

Equivalencia en SQL

```
CREATE TABLE EMPLEADO (
    id int,
    nombre varchar(20),
    salario int,
    deptno int
);
```

```
INSERT INTO PERSON VALUES(23, 'Juan Cruz', 35000, 10);
```

Otro ejemplo

```
$ CREATE (d1: Departamento {num:7, nombre:'Contabilidad', localización:  
'Madrid'})
```



Added 1 label, created 1 node, set 3 properties, completed after 10 ms.

Table

Code

Un nodo puede tener varias etiquetas



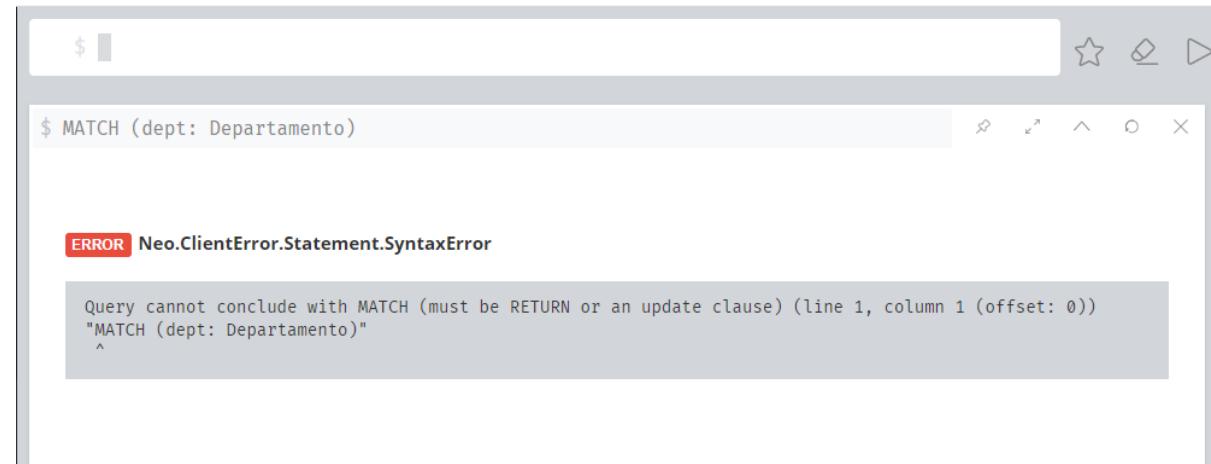
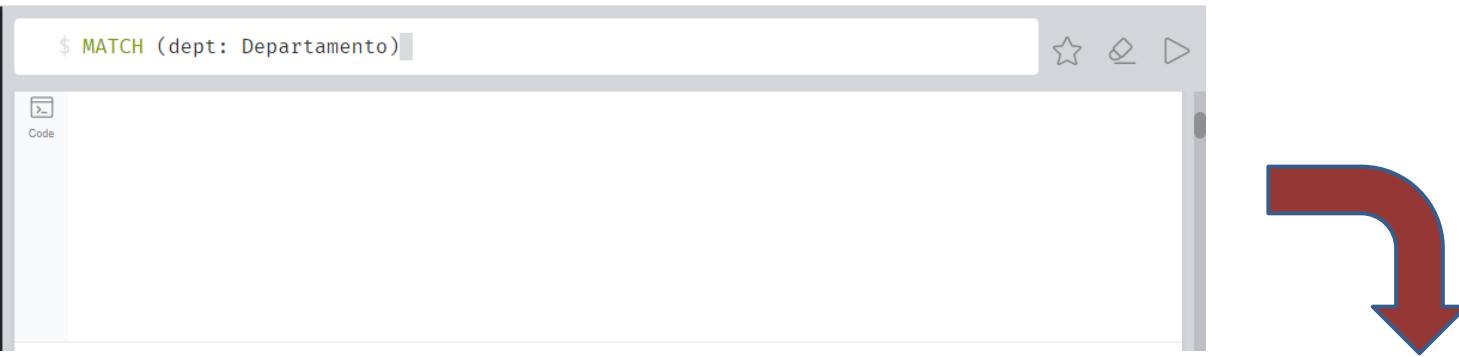
The screenshot shows the Neo4j Browser interface. In the top-left query editor, the command `$ CREATE (m: Movie: Cinema: Film: Picture)` is entered. To the right of the query are three icons: a star, a pencil, and a triangle. On the left side of the screen is a sidebar with two tabs: "Table" (which is selected) and "Code". The main content area displays the message: "Added 1 label, created 1 node, set 3 properties, completed after 10 ms."

El comando MATCH

- Para localizar datos sobre nodos, relaciones y propiedades

```
MATCH
(
    <nombre-nodo>:<nombre-etiqueta>
)
```

- Debe ir siempre con RETURN o comando de actualización



El comando RETURN

- Para devolver propiedades sobre nodos, relaciones y propiedades

```
RETURN  
(  
    <nombre-nodo>.<nombre-propiedad1>,  
    .....  
    <nombre-nodo>.<nombre-propiedadn>  
)
```

- Debe ir siempre con MATCH o CREATE

```
$ RETURN dept.num
```



ERROR Neo.ClientError.Statement.SyntaxError

```
Variable `dept` not defined (line 1, column 8 (offset: 7))
"RETURN dept.num"
^
```

El comando MATCH-RETURN

```
$ MATCH (dep:Departamento) RETURN dep.num, dep.nombre
```

	dep.num	dep.nombre
1	7	"Contabilidad"

MATCH-RETURN sin especificar propiedades

```
$ MATCH (dep:Departamento) RETURN dep
```

The screenshot shows the Neo4j browser interface. On the left, there is a sidebar with four tabs: 'Graph' (selected), 'Table', 'Text', and 'Code'. The main area displays a single node labeled 'Madrid' in a red circle. Above the node, the query '\$ MATCH (dep:Departamento) RETURN dep' is shown, along with the result '(1)' and 'Departamento(1)'. The browser has standard window controls at the top right.

```
SELECT * FROM Departamento
```

Más sobre RETURN

```
MATCH p=(a { nombre: "A" })-[r]->(b)  
RETURN *
```

```
MATCH p=(a { nombre: "A" })  
RETURN a.edad AS AlgoTotalmenteDiferente
```

```
MATCH (a { nombre: "A" })-->(b)  
RETURN DISTINCT b
```

Otros dos nodos

CLIENTE

```
$ CREATE (e:Cliente{id:"1001", nombre:"Abc", fdn:"01/10/1982"})
```



TARJETA DE CRÉDITO

```
$ CREATE (tc:TarjetaCredito{id:"5001",  
numero:"1234567890",cvv:"888",fechacaducidad:"03/23"})
```



Relaciones

```
$ MATCH (c:Cliente),(tc:TarjetaCredito) CREATE (c)-  
[r:DO_SHOPPING_WITH{fecha_compra:"07/12/2016", importe:350}]→(tc)
```



Set 2 properties, created 1 relationship, completed after 8 ms.

Crear una nueva relación entre dos nodos

```
$ CREATE p = (Nadal {nombre:"Rafa Nadal"})-[:TOPSCORER_OF]→  
  (Spa{nombre:"Spain"})-[:WINNER_OF]→(CD2019{nombre: "Copa Davis 2019"})  
RETURN p
```



La cláusula WHERE

WHERE (<condición><operador-booleano><condición>)

Operadores booleanos

AND

OR

NOT

XOR

Operadores de comparación

=

<>

<

>

<=

>=

Ejemplos

```
$ MATCH (e:Empleado)  
WHERE e.nombre = 'Abc'  
RETURN e
```



```
1 MATCH (e:Empleado)  
2 WHERE e.nombre = 'Abc' OR e.nombre = 'Xyz'  
3 RETURN e
```



Crear una nueva relación con la cláusula WHERE

```
1 MATCH (c:Cliente),(tc:TarjetaCredito)
2 WHERE c.id = "1001" AND tc.id="5001"
3 CREATE (c)-[r:DO_SHOPPING_WITH{fecha_compra:"07/12/2016", importe:350}]->(tc)
RETURN r
```



DELETE

```
$ MATCH (e:Empleado) DELETE e
```



```
$ MATCH (e:Empleado) DELETE e
```



Deleted 1 node, completed after 1 ms.

Table

Code

```
$ MATCH (tc: TarjetaCredito)-[r]-(c:Cliente)  
DELETE tc,r,c
```



```
$ MATCH (tc: TarjetaCredito)-[r]-(c:Cliente) D...
```



Deleted 2 nodes, deleted 3 relationships, completed after 3 ms.

Table

Code

REMOVE para eliminar una propiedad de todos los nodos

```
1 MATCH (tc: TarjetaCredito)
2 REMOVE tc.cvv
3 RETURN tc
```



REMOVE para eliminar una etiqueta de un nodo o relación

Graph

*(4) Cinema(1) Film(1) Movie(1) Picture(1)



Table

A Text

Code

```
1 MATCH (m:Movie)  
2 REMOVE m:Picture
```

★ ⌂ ▶

Graph

*(3) Cinema(1) Film(1) Movie(1)



Table

A Text

Code

SET para asignar valores a propiedades

```
1 MATCH (tc:TarjetaCredito)  
2 SET tc.atm_pin = 3456  
3 RETURN tc
```



ORDER BY

```
1 MATCH (e:Empleado)
2 RETURN e.id, e.nombre, e.salario, e.deptno
3 ORDER BY e.nombre
```

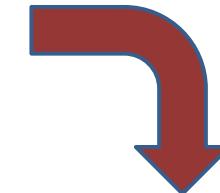


```
1 MATCH (e:Empleado)
2 RETURN e.id, e.nombre, e.salario, e.deptno
3 ORDER BY e.nombre DESC
```

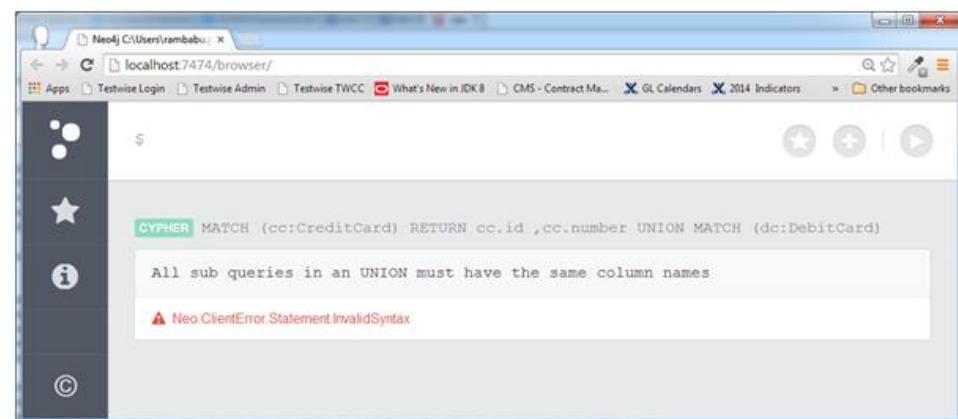


La cláusula UNION

```
1 MATCH (cc:TarjetaCredito) RETURN cc.id, cc.numero  
2 UNION  
3 MATCH (dc:TarjetaCredito) RETURN dc.id, dc.numero
```



```
MATCH (cc:TarjetaCredito)  
RETURN cc.id as id, cc.numero as numero  
UNION  
MATCH (dc:TarjetaCredito)  
RETURN dc.id as id, dc.numero as numero
```



La cláusula UNION ALL

- Mantiene filas duplicadas

```
MATCH (cc:TarjetaCredito)
RETURN cc.id as id, cc.numero as numero
UNION ALL
MATCH (dc:TarjetaCredito)
RETURN dc.id as id, dc.numero as numero
```

Las cláusulas *LIMIT* y *SKIP*

```
$ MATCH (n:Persona) RETURN n LIMIT 25
```

Graph

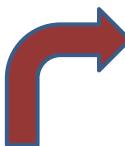
Table

A
Text

Code

"n"
{"nombre": "Mariap", "id": 1}
{"nombre": "Juan González", "id": 6}
{"nombre": "Juap", "id": 2}
{"nombre": "Pedrop", "id": 3}
{"nombre": "Elisap", "id": 4}
{"nombre": "Anap", "id": 5}

Muestra las 2 primeras filas



```
$ MATCH (n:Persona) RETURN n LIMIT 2
```

Graph

Table

"n"

{"nombre": "Mariap", "id": 1}

{"nombre": "Juan González", "id": 6}

```
$ MATCH (n:Persona) RETURN n SKIP 2
```

Graph

Table

"n"

{"nombre": "Juap", "id": 2}

{"nombre": "Pedrop", "id": 3}

{"nombre": "Elisap", "id": 4}

{"nombre": "Anap", "id": 5}

Salta las 2 primeras filas



MERGE = CREATE + MATCH

1

```
$ CREATE (e1:Estudiante {id:3,  
nombre: "Juan Perez"})
```

2

```
$ CREATE (e1:Estudiante {id:3,  
nombre: "Juan Perez"})
```

3

```
1 MATCH (e1:Estudiante)  
2 RETURN e1.id, e1.nombre
```

The screenshot shows a Neo4j browser window. At the top, there is a code editor with the following query:

```
$ MATCH (e1:Estudiante) RE...
```

Below the code editor is a results table. The table has two columns: "e1.id" and "e1.nombre". There are two rows in the table, both containing the value "3" in the "e1.id" column and the string "'Juan Perez'" in the "e1.nombre" column.

"e1.id"	"e1.nombre"
3	"Juan Perez"
3	"Juan Perez"

On the left side of the results table, there are three buttons: "Table", "Text" (which is currently selected), and "Code".

CREATE no comprueba
si el nodo ya existe

con MERGE

1

```
$ MERGE (e2:Estudiante {id:4,  
nombre: "Pilar Ruiz"})
```

2

```
$ MERGE (e2:Estudiante {id:4,  
nombre: "Pilar Ruiz"})
```

3

```
$ MATCH (n:Estudiante) RETURN n
```

```
$ MATCH (n:Estudiante) RET...
```



Graph



Table



Text

```
"n"
```

```
{"nombre":"Pilar Ruiz","id":4}
```

```
{"nombre":"Juan Perez","id":3}
```

```
{"nombre":"Juan Perez","id":3}
```

MERGE añade el nodo
solo si NO existe

JS (NOT) NULL

1

```
$ MATCH (n:Empleado) RETURN n
```

1

Graph

6

A

2

Code

2

CREATE (e:Empleado)

Asigna NULL a las propiedades no especificadas

```
$ MATCH (n:Empleado) RETURN n
```

1

Graph

1

Table

-
Tex

8

Cod

JS (NOT) NULL

```
MATCH (n:Empleado)  
WHERE n.id IS NOT NULL  
RETURN n
```



\$ MATCH (n:Empleado) WHERE n.id IS NOT NULL RETURN n

Graph

Table

A Text

Code

"n"
{"nombre": "Mariap", "id": 1}
{"nombre": "Juan González", "id": 6}
{"nombre": "Juanc", "id": 2}
{"nombre": "Pedrop", "id": 3}
{"nombre": "Elisap", "id": 4}
{"nombre": "Anap", "id": 5}

El operador IN

```
MATCH (e:Empleado)  
WHERE e.id IN [3,4]  
RETURN e
```

```
$ MATCH (e:Empleado) WHERE e.id IN [3,4] RETURN e
```



Graph



Table



"e"
{"nombre":"Pedrop","id":3}
{"nombre":"Elisap","id":4}



Funciones para Strings

Función	Descripción
UPPER	Cambiar letras a mayúsculas
LOWER	Cambiar letras a minúsculas
SUBSTRING	Extraer substring de un string
REPLACE	Sustituir un substring por otro substring

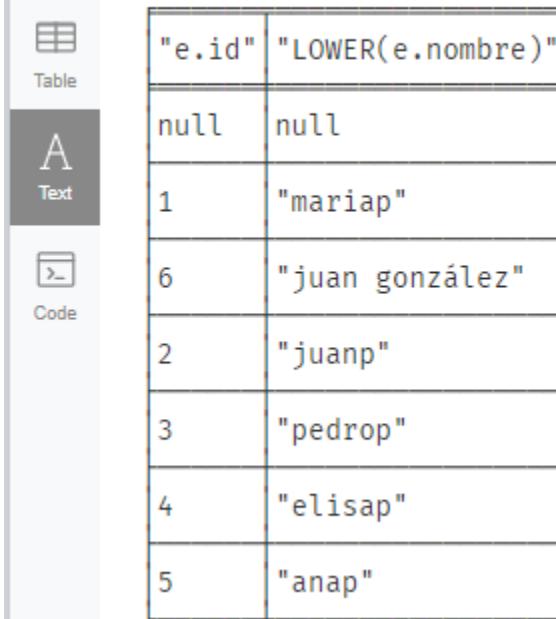
\$ MATCH (e:Empleado) RETURN e.id, UPPER(e.nombre)

The screenshot shows a Neo4j browser window. On the left, there's a sidebar with three tabs: 'Table' (selected), 'Text' (disabled), and 'Code' (disabled). The main area displays a table with two columns: 'e.id' and 'UPPER(e.nombre)'. The data rows are:

"e.id"	"UPPER(e.nombre)"
null	null
1	"MARIAP"
6	"JUAN GONZÁLEZ"
2	"JUANP"
3	"PEDROP"
4	"ELISAP"
5	"ANAP"

LOWER

```
$ MATCH (e:Empleado) RETURN e.id, LOWER(e.nombre)
```



The screenshot shows the Neo4j Browser interface with a query results table. The table has two columns: "e.id" and "LOWER(e.nombre)". The data rows are:

e.id	LOWER(e.nombre)
null	null
1	"mariap"
6	"juan gonzález"
2	"juanp"
3	"pedrop"
4	"elisap"
5	"anap"

SUBSTRNG

```
MATCH (e:Empleado)
RETURN e.id, SUBSTRING(e.nombre,0,2)
```

"e.id"	"SUBSTRING(e.nombre,0,2)"
null	null
1	"Ma"
6	"Ju"
2	"Ju"
3	"Pe"
4	"El"
5	"An"

comienzo longitud

REPLACE

```
$ MATCH (e:Estudiante) RETURN e.id,REPLACE(e.nombre,"a","e")
```



Rows	e.id	REPLACE(e.nombre,"a","e")
3		Merie

Returned 1 row in 543 ms.

Funciones agregadas

Función agregada	Descripción
COUNT	Devuelve el número de filas devueltas por el comando MATCH
MAX	Devuelve el valor máximo del conjunto de filas devueltas por el comando MATCH
MIN	Devuelve el valor mínimo del conjunto de filas devueltas por el comando MATCH
SUM	Devuelve la suma de las filas devueltas por el comando MATCH
AVG	Devuelve la media de las filas devueltas por el comando MATCH

count

```
$ MATCH (e:Empleado) RETURN COUNT(*)
```

The screenshot shows a Neo4j browser interface. On the left, there is a sidebar with three items: 'Table' (represented by a grid icon), 'Text' (represented by a text 'A' icon), and 'Code' (represented by a code editor icon). The 'Text' item is currently selected. In the main area, a query is displayed in a light gray box: '\$ MATCH (e:Empleado) RETURN COUNT(*)'. Below the query, the results are shown in a table with two rows. The first row contains the text '"COUNT(*)"' in blue, and the second row contains the number '7' in orange. The entire interface has a clean, modern design with a white background.

"COUNT(*)"
7

MAX y MIN

```
1 MATCH (n:Empleado)
2 RETURN MAX(n.salario), MIN(n.salario)
```

```
$ MATCH (n:Empleado) RETURN MAX(n.salario), MIN(...
```

Table

A

"MAX(n.salario)"	"MIN(n.salario)"
30000	25000

SUM y AVG

```
1 MATCH (n:Empleado)
2 RETURN SUM(n.salario), AVG(n.salario)
```

```
$ MATCH (n:Empleado) RETURN SUM(n.salario), AVG(
```



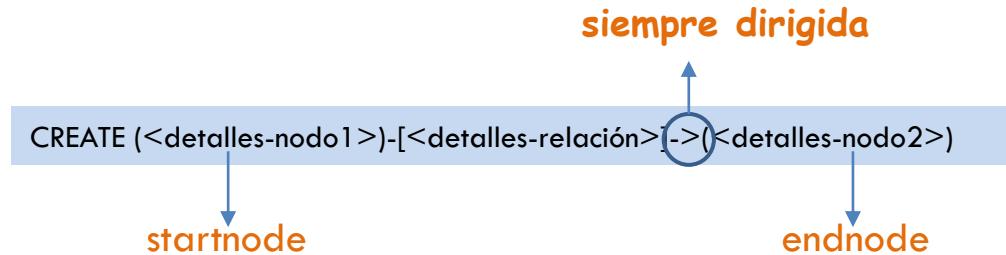
Table



Text

"SUM(n.salario)"	"AVG(n.salario)"
55000	27500.0

Funciones para relaciones

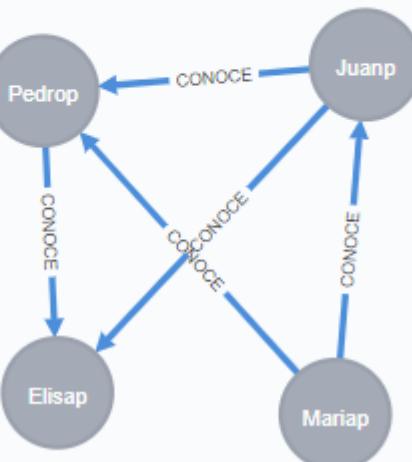


Función	Descripción
STARTNODE	Devuelve el nodo de inicio de una relación
ENDNODE	Devuelve el nodo de destino de una relación
ID	Devuelve el ID de una relación
TYPE	Devuelve el tipo de una relación como string

STARTNODE y *ENDNODE*

```
$ MATCH p=( )-[r:CONOCE]→() RETURN p
```

Graph *(4) Empleado(4)
Table *(5) CONOCE(5)



Table

Text

Code

```
$ MATCH (a{id:1})-[r:CONOCE]→(b{id:2}) RETURN STARTNODE(r)
```

*(1) Empleado(1)



Mariap

```
$ MATCH (a{id:1})-[r:CONOCE]→(b{id:2}) RETURN ENDNODE(r)
```

*(1) Empleado(1)



Juanp

ID y TYPE

```
$ MATCH (a{id:1})-[r:CONOCE]→(b{id:2}) RETURN ID(r), TYPE(r)
```

	ID(r)	TYPE(r)
1	181023	"CONOCE"

Patrones y reconocimiento de patrones

```
(:Persona) -[:LIVES_IN]-> (:Ciudad) -[:PART_OF]-> (:País)
```

- Se usan para
 - describir la forma de lo que buscamos
 - proporcionar el camino desde donde empezar a buscar las ocurrencias del patrón que buscamos
- En CREATE, MATCH, MERGE, WHERE, ...

Patrones y reconocimiento de patrones

- Patrones para nodos
 - (a)
- Podemos añadir propiedades
 - (a { name: "Andres", sport: "Brazilian Ju-Jitsu" })
- Patrones para caminos
 - (a) --> (b)
 - (a) --> (b) <-- (c)
 - (a) --> () <-- (c) *(se puede omitir nombre del nodo)*
 - (a) --> (b) *(nos vale cualquier dirección)*
 - (a:User) --> (b) *(podemos añadir etiquetas)*

Patrones y reconocimiento de patrones

- Patrones para relaciones

(n)-[r]->(n1)

(n)-[r]-(n1)

- Podemos añadir propiedades

[r:FRIEND {Since: 2004, LastVisited:"Jan-2015"}]

(a)-[{"blocked": false}]->(b)

Patrones y reconocimiento de patrones

- Opciones
 - (a)-[r:TYPE1|TYPE2]->(b)
- Se puede omitir el nombre de la relación
 - (a)-[:REL_TYPE]->(b)
- Podemos usar variables
 - Var = (n)-[:REL_TYPE]->(n1)

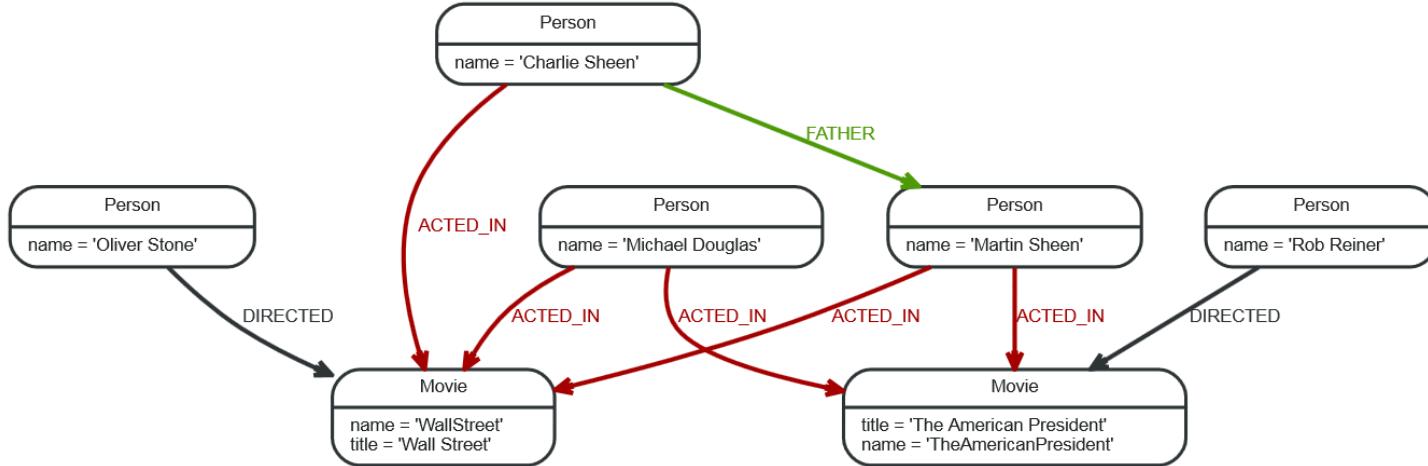
Patrones y reconocimiento de patrones

➤ Caminos de longitud variable

(a)-[*2]->(b)	<i>equivalente a (a)-->()-->(b)</i>
(a)-[*3..5]->(b)	<i>camino de longitud entre 3 y 5</i>
(a)-[*3..]->(b)	<i>camino de longitud 3 o mas</i>
(a)-[*..5]->(b)	<i>camino de longitud 5 o menos</i>
(a)-[*]->(b)	<i>camino de cualquier longitud</i>

```
MATCH (me)-[:KNOWS*1..2]-(remote_friend)  
WHERE me.name = "Filipa"  
RETURN remote_friend.name
```

shortestPath

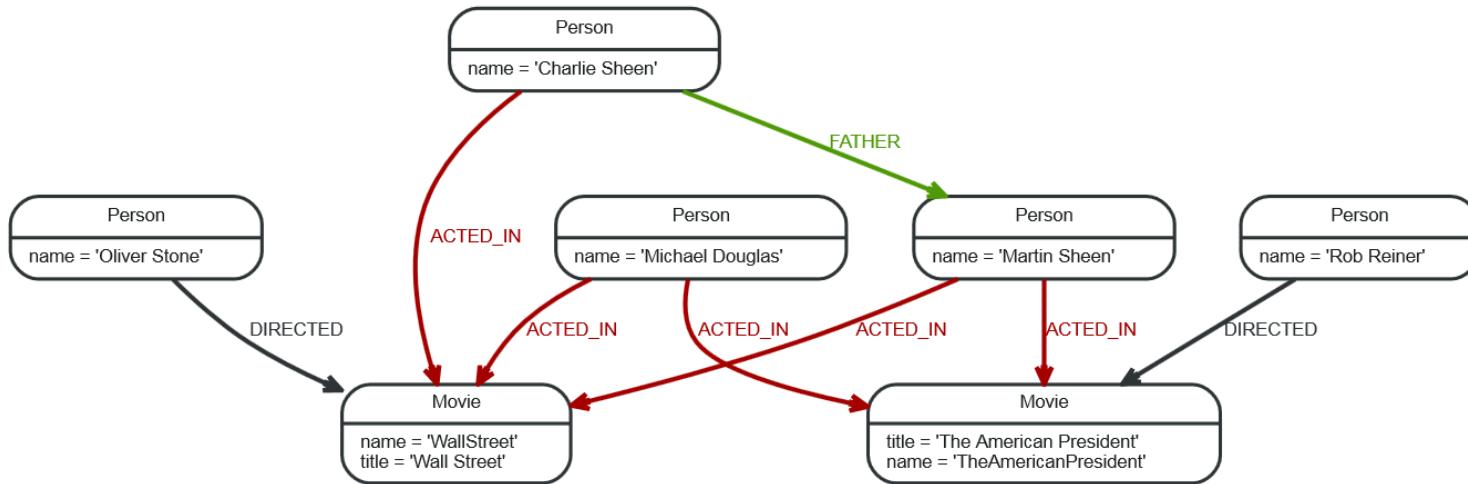


```
MATCH (martin:Person { name:"Martin Sheen" }), (oliver:Person { name:"Oliver Stone" }), p =  
shortestPath((martin)-[*..15]-(oliver))  
RETURN p
```

p
[Node[1]\{name:"Martin Sheen"\},:ACTED_IN[1]\{role:"Carl Fox"\},Node[5]\{title:"Wall Street"\},:DIRECTED[3]\{\},Node[3]\{name:"Oliver Stone"\}]
1 row



allShortestPaths



```
MATCH (martin:Person { name:"Martin Sheen" }),(michael:Person { name:"Michael Douglas" }), p =  
allShortestPaths((martin)-[*]-(michael))  
RETURN p
```

p
[Node[1]\{name:"Martin Sheen"\},:ACTED_IN[1]\{role:"Carl Fox"\},Node[5]\{title:"Wall Street"\},:ACTED_IN[2]\{role:"Gordon Gekko"\},Node[2]\{name:"Michael Douglas"\}]
[Node[1]\{name:"Martin Sheen"\},:ACTED_IN[4]\{role:"A.J. MacInerney"\},Node[6]\{title:"The American President"\},:ACTED_IN[5]\{role:"President Andrew Shepherd"\},Node[2]\{name:"Michael Douglas"\}]

Importar datos

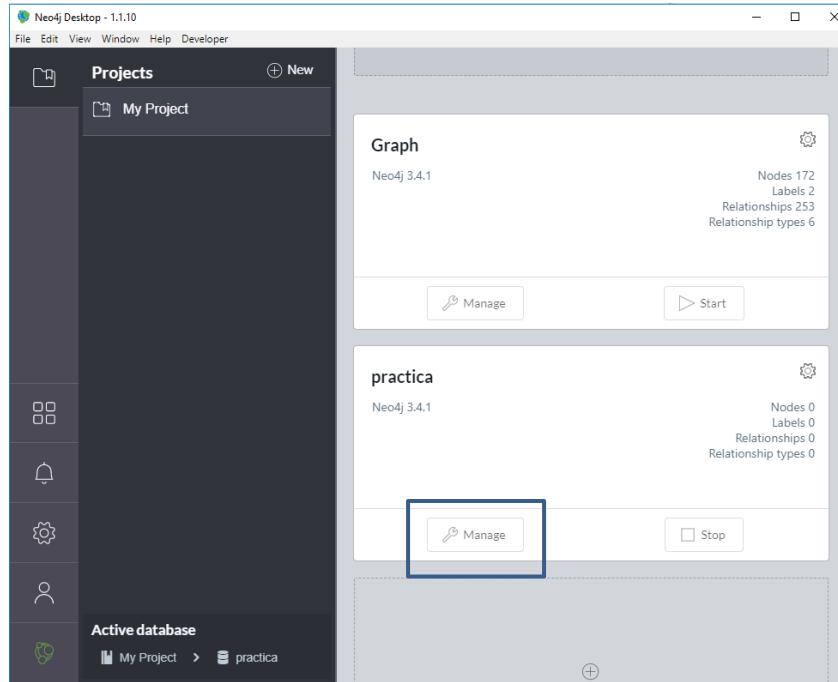
LOAD CSV WITH HEADERS FROM
"file:/est.csv" AS csvLine

CREATE (e:Estudiante{id: tolnt(csvLine.id),
nombre: csvLine.nombre})

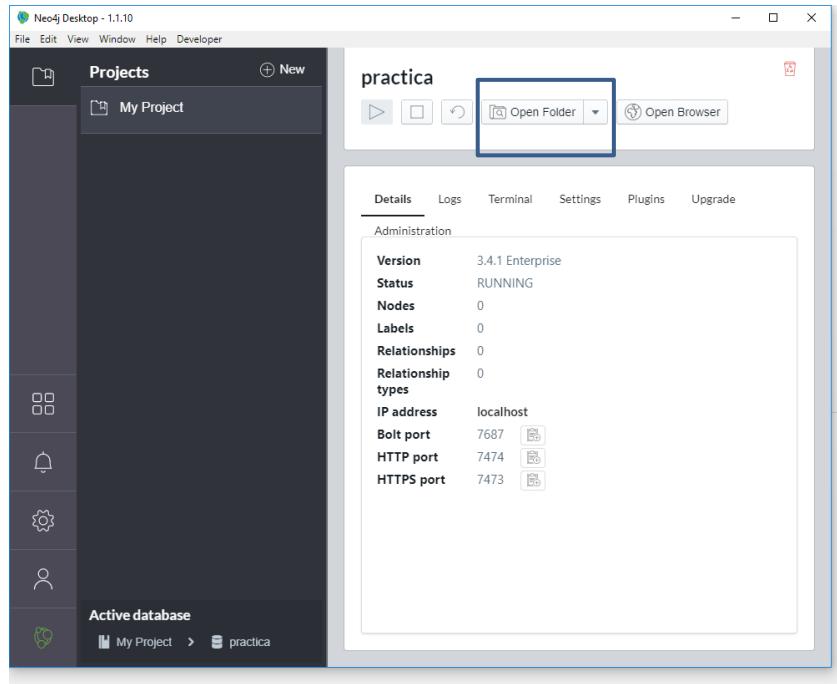
Northwind graph
The Movie Graph

estudiantes: Bloc de notas	
Archivo	Edición
Formato	Ver
Ayuda	X
id,nombre	
1,María	
2,Juan	
3,Pedro	
4,Elisa	
5,Ana	
6,Prado	
7,Rosa	
8,Clara	
9,Daniel	
10,Andrés	
11,Paco	
12,Begoña	
13,Macarena	
14,Teresa	
15,Fernando	
16,Luis	
17,Mar	
18,Carlos	
19,Marta	
20,Tirsa	

Colocar el fichero en la carpeta adecuada



Colocar el fichero en la carpeta adecuada



Borrar el contenido de la BD

MATCH (n)

OPTIONAL MATCH (n)-[r]-0

DELETE n,r

Mostrar el contenido de la BD

```
MATCH (n)  
RETURN n
```

Bibliografía

- NoSQL for mere mortals. Dan Sullivan. Pearson Education. 2015.
- Graph databases. Ian Robinson, Jim Webber & Emil Eifrem. O'Reilly. 2015.
- Neo4j: qué es y para qué sirve una base de datos orientada a grafos. BBVAOpen4U.
<https://bbvaopen4u.com/es/actualidad/neo4j-que-es-y-para-que-sirve-una-base-de-datos-orientada-grafos>
- Neo4j Tutorial. <https://www.tutorialspoint.com/neo4j/index.htm>



redis

índice

1. Introducción
2. Estructuras de datos
3. Programación nativa: LUA
4. Aplicaciones reales
5. Python y Redis

Introducción

- Redis (**RE**mote **D**ictionary **S**erver) es un almacén en memoria de estructuras de datos open source (licencia BSD) utilizada habitualmente como base de datos, caché o message broker ('traductor' intermedio entre lenguajes).
- Soporta estructuras de datos tales como strings, hashes, listas, sets (colecciones), sorted sets ...
- Diseñada para desplegarse con alta disponibilidad y escalabilidad (replicación y particiones)
- Hace uso de scripting de LUA nativo

Introducción

- Fácil de usar
- Ligera, de pequeño tamaño y muy rápida
- Buen rendimiento en lectura y escritura
- Soportado por multitud de lenguajes:
 - Python: `python-py`
 - Java: `jedis`
 - C: `hiredis`
 - Node.js: `node_redis`
 - Y muchos más: <https://redis.io/clients>

Introducción

- Cada elemento se compone de 3 componentes:
 - **Clave (key)**: identificador único del elemento
 - **Valor (value)**: estructura de datos almacenada
 - **Tiempo de vida (expires)**: indica la fecha de caducidad del elemento



Estructuras de datos

- Cada elemento posee una clave (key)
 - Debe ser única
 - Es binary safe: se trata como bytes y se ignora todo aspecto del texto
 - Puede ser un String, un Hash, el contenido de un JPEG...
 - Reglas a seguir:
 - Una clave muy larga no es buena idea (comparaciones caras en tiempo, alto gasto en memoria)
 - Una clave muy corta no es buena idea: balance entre clave corta y clave larga
 - El tamaño máximo para una clave es de 512 MB
 - Se recomienda seguir siempre el mismo esquema de claves dentro de una base de datos

Estructuras de datos

- Redis Expires:
 - De uso opcional
 - Permite asignar un timeout a una key
 - Pasado el tiempo indicado se ejecuta un 'DEL' internamente
 - Se puede asignar en segundos o milisegundos

Estructuras de datos

- Redis Strings:
 - Estructura de datos más sencilla
 - Puede representar un texto, un número o un valor binario (JPEG...)
 - Las keys son siempre Strings
 - No pueden ser mayores de 512 MB
 - El tipo de dato determina las operaciones que podemos utilizar

Estructuras de datos

- Redis Lists:
 - Lista enlazada, no Array
 - PRO:
 - Tiempo para añadir 1 elemento en una lista de 10 elementos
= Tiempo para añadir 1 elemento en una lista de 1.000.000 elementos
 - CONTRA:
 - Acceder por índice no es tan rápido
 - Prima velocidad de adición frente a velocidad de consulta
 - Si interesa la velocidad de consulta mejor utilizar un Sorted Set

Estructuras de datos

- Casos de uso:
 - Recordar las últimas actualizaciones publicadas por un usuario en una red social
 - Comunicación entre procesos
 - Patrón consumer-producer
 - Listas limitadas:
 - Van 'olvidando' los primeros elementos con el tiempo
 - Colas de procesamiento:
 - Permite bloquear operaciones y evitar hacer polling (BRPOP y BLPOP)

Estructuras de datos

- Redis Hashes:
 - Colección de pares clave–valor (diccionario)
 - Se puede acceder a los pares de forma independiente
 - Tamaño del hash limitado por memoria (virtualmente ilimitado)
 - Hashes de pequeño tamaño se guardan de forma más eficiente
 - Casos de uso:
 - Perfectos para representar objetos
 - Caché

Estructuras de datos

- Redis Sets:
 - Colecciones desordenadas de Strings (únicos)
 - Permite comprobar si un elemento ya se ha añadido
 - Permite realizar intersecciones y uniones
 - Casos de uso:
 - Expresar relaciones entre objetos (asignar tags)

Estructuras de datos

- Redis Sorted Sets:
 - Mezcla entre Sets y Hashses
 - Colección de Strings (únicos)
 - Cada elemento tiene asignado un 'score'
 - Clave (string) – Valor (score)
 - Se almacenan ordenados siguiendo estas reglas:
 - Si A y B son elementos de nuestro Sorted Set
 - A > B si A.score > B.score
 - A > B si A.score = B.score & lexicographical(A.key) > lexicographical(B.key)
 - Implementación eficiente

Estructuras de datos

- Redis Sorted Sets:
 - Permite operaciones sobre rangos de scores
 - El score se puede actualizar
 - El coste de una actualización no es elevado
 - Casos de uso:
 - Listado de “más vistos/populares/etc”

Estructuras de datos

- Redis Bitmaps:
 - No es una estructura de datos
 - Conjunto de operaciones a nivel de bit para Strings
 - 512MB → 2^{32} bits
 - Tipos de operaciones:
 - Operaciones a nivel de bit (cambiar valor, obtener valor)
 - Operaciones sobre grupo de bits (contar bits asignados, etc)
 - Compresión de espacio a la hora de almacenar información

Estructuras de datos

- Redis Bitmaps:
 - Casos de uso:
 - Análisis en tiempo real
 - Ejemplo: Usuarios únicos que se conectaron a mi web hoy
 - Almacenar información booleana asociada a IDs de objetos
 - Ejemplo: Listado de ID usuario y boolean de si quieren o no recibir mi newsletter

Estructuras de datos

- Redis HyperLogLogs:
 - Estructura de datos probabilística utilizada para contar cosas únicas
 - Para contar elementos únicos se necesita gastar una cantidad de memoria proporcional a la cantidad de elementos a contar
 - Hay un conjunto de algoritmos que intercambian memoria por precisión
 - Redis tiene un porcentaje de error menor al 1%
 - Consume un máximo de 12k bytes de memoria
 - Internamente utiliza Redis Strings

Programación nativa en Lua

- Sitio oficial: www.lua.org
- Creado en 1993
- Lenguaje compacto
- Fácilmente embebible en otras aplicaciones
- Utilizado principalmente para el desarrollo de juegos y como lenguaje de scripting nativo en: Adobe Photoshop Lightroom, Angry Birds, Apache HTTP Server, VLC, World of Warcraft, ...
- Y Redis



Programación nativa en Lua

- Características
 - Lenguaje de propósito general
 - Lenguaje tipado dinámicamente: al declarar una variable no es necesario indicar el tipo de valores que va a guardar (como Python)
 - Lenguaje interpretado: analizado y compilado en tiempo de ejecución
 - Gestión automática de memoria
 - Lenguaje extensible: podemos desarrollar módulos que pueden ser utilizados en otros programas, etc
 - Buen rendimiento
 - Es simple

Programación nativa en Lua

- Permite crear extensiones propias para tu BD Redis
- Podemos ejecutar scripts Lua desde Redis con el siguiente comando:
 > *EVAL 'local val="Hello World" return val' 0*
- Los scripts de LUA actúan como transacciones inteligentes (permiten implementar control de errores, condiciones, etc)

Programación nativa en Lua

- Ejecución atómica
 - Un único script en ejecución al mismo tiempo
- Se define un tiempo de ejecución máximo para cada script (parámetro lua-time-limit, por defecto 5 segundos)
- Alcanzado el tiempo de ejecución máximo, el script se detendrá
- Un script a medias puede causar inconsistencias en la BD

Programación nativa en Lua

- Claves y argumentos
 - ***EVAL 'local val="Hello World" return val' 0***
 - El 0 indica que se pasan un total de 0 claves al script
 - ***EVAL "return ARGV[1]..''..KEYS[1]" 1 keyName "Hello"***
 - El 1 indica que se pasa una clave al script (name:first)
 - El resto de parámetros son argumentos
 - Se accede a una clave utilizando KEYS[i]
 - Se accede a un argumento utilizando ARGV[i]
 - ¿Cómo accedemos al valor de una clave?

Programación nativa en Lua

- Accediendo a los valores de Redis
 - Desde un script de LUA podemos llamar a una función de Redis con el comando redis.call()
 - *EVAL 'return ARGV[1].. " ..redis.call("get",KEYS[1])' 1
keyName "Hello"*
 - Si hemos asignado previamente un valor a la clave name:first obtendremos algo como esto:
"Hello Carlos"

Programación nativa en Lua

- Bibliotecas disponibles en Lua de Redis:
 - base, bitop, cmsgpack, math, redis, string, struct, table
- Conversión de tipos Redis-Lua:

REDIS	LUA
integer	number
string	string
array	table
status	table con 1 campo 'ok'
error	table con 1 campo 'err'
Nil	false

LUA	REDIS
number	integer
string	string
table	array (corta al llegar a un nil)
table con 1 campo 'ok'	status
table con 1 campo 'err'	error
false	Nil

Programación nativa en Lua

- Creando scripts más complejos
 - Un script complejo se volvería muy “sucio” al lanzar el comando EVAL
 - Almacenamos nuestro script en un fichero y ejecutamos el mismo

\$> redis-cli --eval hello.lua name:first , Hello

hello.lua:

```
local name=redis.call("get", KEYS[1])
local greet=ARGV[1]
local result=greet.." "..name
return result
```

- redis-cli cuenta las claves por nosotros
 - La ',' indica el fin de las claves y el inicio de los argumentos

Programación nativa en Lua

- Gestión de scripts
 - El comando **EVAL** envía el cuerpo del script cada vez que se ejecuta
 - Cada ejecución requiere volver a compilar el script
 - Redis posee una caché de scripts
 - Cada script posee una huella única determinada por su **SHA1**
 - El comando **SCRIPT LOAD** carga un script en la caché y devuelve su **SHA1**
 - El comando **EVALSHA** junto con el identificador ejecuta dicho script desde la caché
 - **SCRIPT LOAD "return redis.call('INFO', 'Server")**
 - **EVALSHA identificador 0**
 - El comando **SCRIPT EXIST** comprueba si un script se encuentra en la caché
 - El comando **SCRIPT FLUSH** borra TODOS los scripts de la caché
 - El comando **SCRIPT KILL** termina la ejecución del script en ejecución

Programación nativa en Lua

- Gestión de errores
 - Las llamadas a Redis pueden devolver error
 - Dos formas de tratarlos desde Lua:
 - **redis.call()** → Detiene la ejecución .
 - Devuelve el error como resultado del script
 - **redis.pcall()** → No detiene la ejecución.
 - Devuelve un table con el status error y el script sigue ejecutándose
 - Se pueden dejar trazas del script con **redis.log()**

Programación nativa en Lua

- Debugging con Lua o LDB
 - Depurador de línea de comandos
 - Breakpoints: `redis.breakpoint()`
 - Ejecución paso a paso
 - **continue/resume**: continua la ejecución hasta otro breakpoint
 - **step over**: ejecuta la línea actual
 - **step in**: ejecuta la primera línea de la función en la que se encuentra
 - **step out**: sale de la función a la que llamó y se detiene
 - **Consola de consulta**: Permite ejecutar comandos
 - **Pila de llamadas**: Contiene las llamadas activas
- Ejecución:
\$> redis-cli --ldb --eval script.lua clave1 ... claveN, arg1 ... argN

Python y Redis

- Python
 - Librería **redis-py**

```
import redis

r = redis.Redis(host='hostname', port=port, password='password')

r.set("key", "Hello World")
print(r.get("key"))
```

- <https://redislabs.com/lp/python-redis/>

Python y Redis

- Cómo cargar un script LUA

```
import redis
```

```
r = redis.Redis(host='hostname', port=port,  
                 password='password')
```

```
luafile = open("<complete_path_to_the_file>", "r").read()  
myluascript = r.register_script(luafile)
```

```
result = myluascript(keys=[<my_keys>], args=[<my_args>])
```

Ventajas vs Desventajas



- **Velocidad, ya que guarda los datos en RAM.**
- **Alto rendimiento.**
- **Fácil de usar e instalar.**



- No puede guardar tablas enteras.
- Consumo RAM elevado.
- Baja relationalidad de datos.

Ejemplos reales

- Caché de páginas web
 - Mayor velocidad
 - Evita la carga del servidor
- Sesiones de usuario
 - El identificador de sesión tiene toda la información asociada
 - Las sesiones finalizan automáticamente al terminar el TTL de la clave
- Carrito de la compra
 - Guardar los artículos seleccionados por el usuario

Ejemplos reales

- **Caché de bases de datos**
 - Almacenar el resultado de consultas de BBDD operacionales que se ejecuten frecuentemente
- **Contadores**
 - Para manejar contadores y estadísticas en tiempo real
 - Ejemplos: votos de usuarios, contadores de acceso a un recurso, etc.
- **Listas de elementos recientes**
 - Mostrar las listas de las últimas actualizaciones de algún componente
 - Ejemplos: últimos comentarios sobre un post, los últimos artículos vistos de un catálogo, etc