El futuro digital es de todos

MinTIC

UNIVERSIDAD EL BOSQUE

MisiónTIC2022

Base de Datos NoSQL

Mag. Carlos Adolfo Beltrán Castro

Contexto de surgimiento

Mag. Carlos Adolfo Beltrán Castro

Año 2015

Por cada minuto del día

YouTube 300 hs. de Video
Facebook 4,166.667 User share
Twitter 347,222 tweets
Apple 51,000 Apps Download
Whatsapp 347,222 Photos
Uber 694 pasajeros
Tinder 590,278 Users Swipe
SnapChat 284,722 Snaps

Población Total de Internet

3.200.000.000 de personas

PRINCIPALES CAMBIOS QUE SE PRODUJERON EN LA TECNOLOGÍA Y EN LOS ÚLTIMOS 15 AÑOS

- MASIFICACIÓN USO DE INTERNET
- SURGIMIENTO DE LAS REDES SOCIALES
- CRECIMIENTO EXPONENCIAL DE DISPOSITIVOS MÓVILES
- INTERFACES DE USUARIO MAS SIMPLES E INTUITIVAS
- CAMBIOS EN LAS FORMAS DE PROCESAMIENTO
- FUERTE BAJA EN LOS COSTOS DE ALMACENAMIENTO

CADA DÍA CREAMOS 2,5 QUINTILLONES DE BYTES DE DATOS. (2,5 Exabytes)

EL 90% DE LOS DATOS DEL MUNDO DE HOY SE GENERARON EN LOS ÚLTIMOS 2 AÑOS

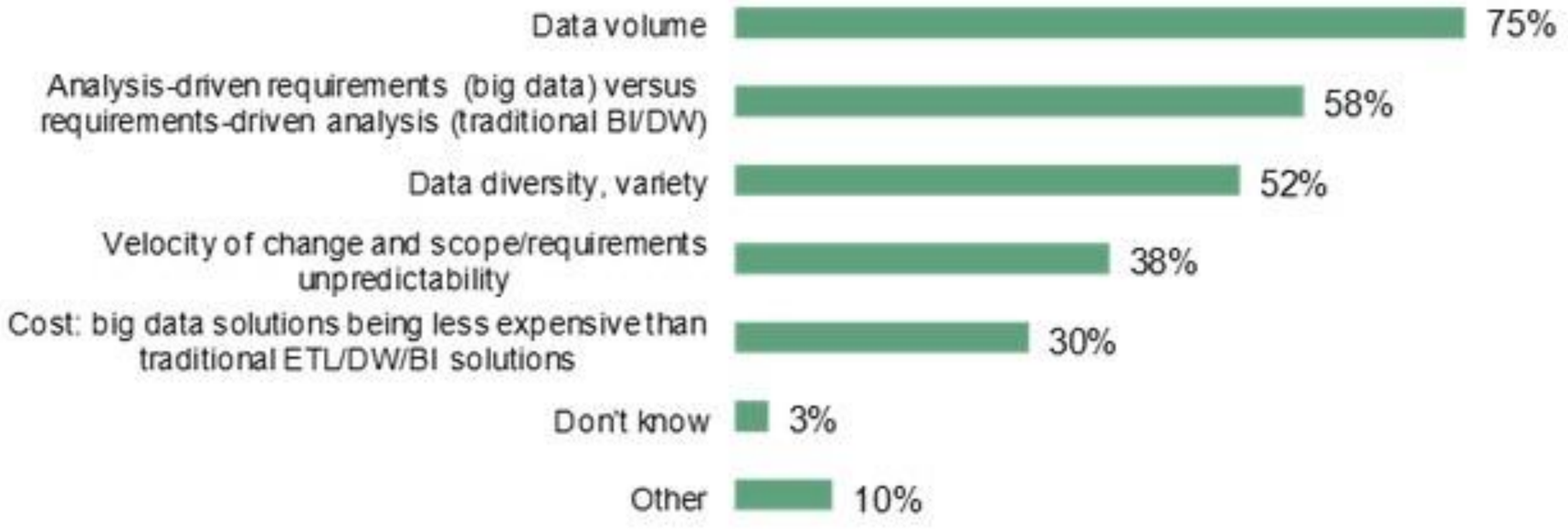
Big Data

Mag. Carlos Adolfo Beltrán Castro

Big Data es el sector de IT que hace referencia a *grandes conjuntos de datos* que por la *velocidad* a la que se generan, la capacidad para tratarlos y los *múltiples formatos y fuentes*, es necesario procesarlos con mecanismos distintos a los tradicionales.

"*Volumen masivo de datos*, tanto *estructurados como no-estructurados*, los cuales son *demasiado grandes y difíciles de procesar* con las bases de datos y el software *tradicionales*." (ONU, 2012)

En ambientes tradicionales de BI y DW primero se generan los requerimientos y luego las aplicaciones. Dicho de otra forma, los requerimientos direccionan las aplicaciones. En Big Data es al revés, ya que se utiliza la exploración de datos libre para generar hipótesis para encontrar un patrón



fuente: forrester research. global big data survey

- Volumen
- Velocidad
- Variedad
- "Veracidad"

El costo es un factor en muchos casos. Las tecnologías utilizadas en Big Data son más económicas que las tradicionales.

El futuro digital es de todos | MinTIC

UNIVERSIDAD EL BOSQUE

SQL Mision TIC 2022

Introducción a

Bases de Datos NoSQL

Mag. Carlos Adolfo Beltrán Castro

¿ Qué es NoSQL ?

Sistemas de gestión de bases de datos que difieren del modelo clásico de bases de datos relacionales: no usan SQL como lenguaje de consulta, los datos almacenados no requieren estructuras fijas como tablas, no garantizan consistencia plena y escalan horizontalmente.

Orientadas a columnas

Este tipo de bases de datos están pensadas para realizar consultas y agregaciones sobre grandes cantidades de datos. Funcionan de forma parecida a las bases de datos relacionales, pero almacenando columnas de datos en lugar de registros.

De clave-valor

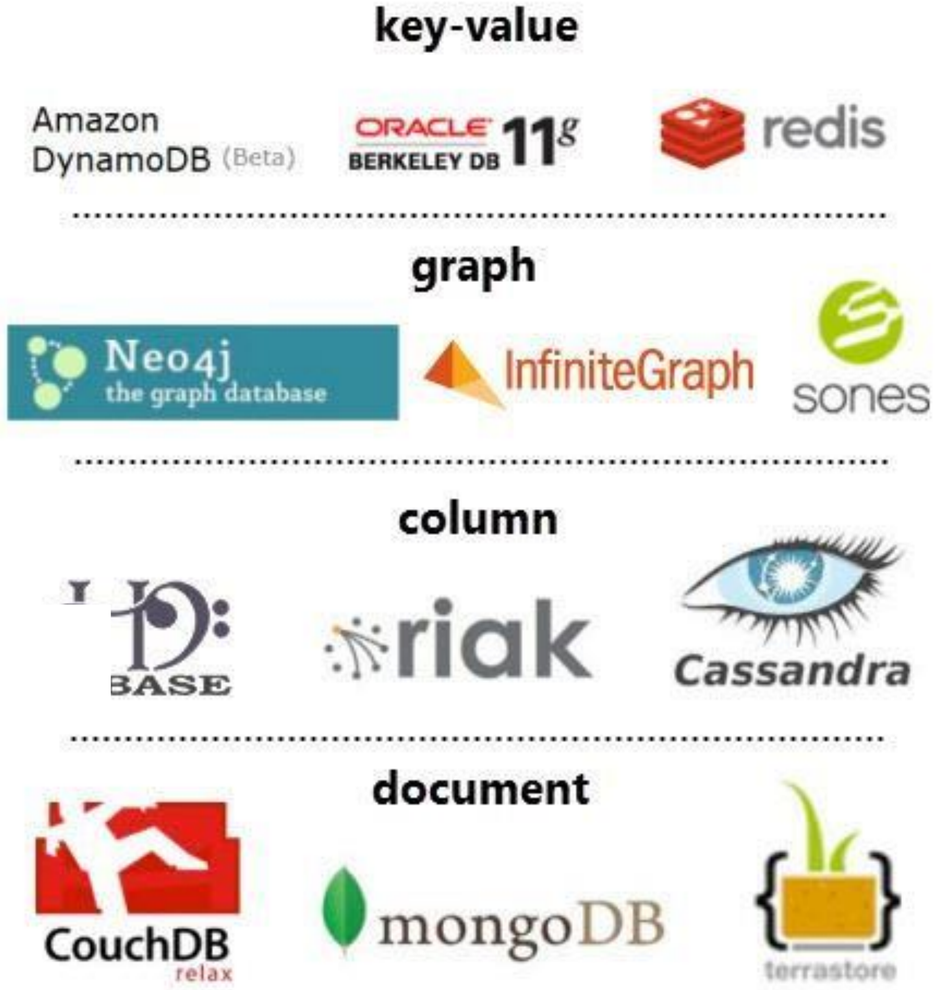
Estas son las más sencillas de entender. Simplemente guardan tuplas que contienen una clave y su valor. Cuándo se quiere recuperar un dato, simplemente se busca por su clave y se recupera el valor.

En grafo

Basadas en la teoría de grafos utilizan nodos y aristas para representar los datos almacenados. Son muy útiles para guardar información en modelos con muchas relaciones, como redes y conexiones sociales

Orientadas a documentos

Son aquellas que gestionan datos semi estructurados. Es decir documentos. Estos datos son almacenados en algún formato estándar como puede ser XML, JSON o BSON.


key-value
Amazon DynamoDB (Beta)
ORACLE BERKELEY DB 11g
redis
graph
Neo4j the graph database
InfiniteGraph
sones
column
BASE
riak
Cassandra
document
CouchDB relax
mongoDB
terrastore

309 systems in ranking, August 2016

Rank Aug 2016	Rank Jul 2016	Rank Aug 2015	DBMS	Database Model	Score Aug 2016	Score Jul 2016	Score Aug 2015
1.	1.	1.	Oracle	Relational DBMS	1427.72	-13.81	-25.30
2.	2.	2.	MySQL	Relational DBMS	1357.03	-6.25	+65.00
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1205.04	+12.16	+96.39
4.	4.	4.	MongoDB	Document store	318.49	+3.49	+23.84
5.	5.	5.	PostgreSQL	Relational DBMS	315.25	+4.10	+33.39
6.	6.	6.	DB2	Relational DBMS	185.89	+0.81	-15.35
7.	7.	↑ 8.	Cassandra	Wide column store	130.24	-0.47	+16.24
8.	8.	↓ 7.	Microsoft Access	Relational DBMS	124.05	-0.85	-20.15
9.	9.	9.	SQLite	Relational DBMS	109.86	+1.32	+4.04
10.	10.	10.	Redis	Key-value store	107.32	-0.71	+8.51
11.	11.	↑ 14.	Elasticsearch	Search engine	92.49	+3.87	+22.85
12.	12.	↑ 13.	Teradata	Relational DBMS	73.64	-0.29	+0.05
13.	13.	↓ 11.	SAP Adaptive Server	Relational DBMS	71.04	+0.31	-14.07
14.	14.	↓ 12.	Solr	Search engine	65.77	+1.08	-16.13
15.	15.	15.	HBase	Wide column store	55.51	+2.37	-4.43
16.	16.	↑ 17.	FileMaker	Relational DBMS	55.01	+3.45	+3.14
17.	↑ 18.	↑ 18.	Splunk	Search engine	48.90	+2.26	+6.71
18.	↓ 17.	↓ 16.	Hive	Relational DBMS	47.82	+0.27	-6.06
19.	19.	19.	SAP HANA	Relational DBMS	42.73	+0.93	+4.48
20.	20.	↑ 25.	MariaDB	Relational DBMS	36.88	+1.08	+12.76
21.	21.	↑ 22.	Neo4j	Graph DBMS	35.57	+1.88	+2.41
22.	22.	↓ 20.	Informix	Relational DBMS	29.05	+0.49	-7.75
23.	23.	↓ 21.	Memcached	Key-value store	27.69	+0.50	-5.69
24.	24.	24.	Couchbase	Document store	27.40	+1.42	+1.24
25.	25.	↑ 28.	Amazon DynamoDB	Document store	26.60	+1.67	+8.15

http://db-engines.com/en/ranking

Base de datos NoSQL

NoSQL se refiere a una gran variedad de tecnologías de bases de datos que se han desarrollado en respuesta a las necesidades de desarrollo de las aplicaciones modernas:

- Los desarrolladores trabajamos con aplicaciones que generan enormes volúmenes de datos nuevos y en constante evolución (estructurados, semiestructurados, no estructurados y polimórficos).
- Aquellos ciclos de desarrollo en cascada que duraban 12 a 18 meses, hace tiempo que pasaron a la historia. Ahora se trabaja en equipos pequeños, que realizan sprints de desarrollo ágiles con iteraciones rápidas y que generan código cada semana o cada quince días, algunos incluso varias veces al día.
- Las aplicaciones que antes servían a un conjunto finito de usuarios ahora se proporcionan como servicios, que no solo deben funcionar sin interrupción, sino que además tienen que ser accesibles desde muchos dispositivos distintos y deben poder escalarse a millones de usuarios de todo el mundo.
- Las empresas ahora recurren a arquitecturas de escalado horizontal, con software de código abierto, servidores convencionales e informática en la nube, en lugar de utilizar grandes estructuras monolíticas de servidores y almacenamiento.
- Las bases de datos relacionales no se diseñaron para poder hacer frente a la escalabilidad y agilidad que necesitan las aplicaciones modernas, ni para beneficiarse de los sistemas de almacenamiento básicos y de la potencia de proceso que existen hoy en día.

Tipos de base de datos NoSQL

- **Bases de datos de documentos**: en estas bases de datos se asocia cada clave con una estructura de datos compleja que se denomina 'documento'. Los documentos pueden contener muchos pares de clave-valor distintos, o pares de clave-matriz o, incluso, documentos anidados.
- **Almacenes de grafos**: se utilizan para almacenar información sobre redes de datos, como las conexiones sociales. Ejemplos de almacenes de grafos son Neo4J y Giraph.
- **Almacenes de clave-valor**: son las bases de datos NoSQL más simples. Cada elemento de la base de datos se almacena como un nombre de atributo (o “clave”), junto con su valor. Ejemplos de almacenes de clave-valor son Riak y Berkeley DB. En algunos almacenes de clave-valor, como Redis, cada valor puede tener un tipo, como “entero”, lo que le añade funcionalidad.
- **Bases de datos orientadas a columnas**: estas bases de datos, como Cassandra o HBase, permiten realizar consultas en grandes conjuntos de datos y almacenan los datos en columnas, en lugar de filas.

Ventajas de NoSQL

Si se comparan con las bases de datos relacionales, las bases de datos NoSQL son más escalables y ofrecen un mayor rendimiento. Además, su modelo de datos aborda varias cuestiones que el modelo relacional pasa por alto:

- Grandes volúmenes de datos estructurados, semiestructurados y no estructurados en constante cambio.
- Sprints de desarrollo ágiles, iteración rápida de los esquemas y generación frecuente de código.
- Programación orientada a objetos flexible y fácil de usar.
- Arquitectura de escalado horizontal, distribuida geográficamente, en lugar de una arquitectura monolítica.

INTRODUCCIÓN A MONGODB

Mag. Carlos Adolfo Beltrán Castro

- Las bases de datos almacenan y recuperan documentos que pueden ser XML,

  JSON, BSON, etc.

- Los documentos almacenados son similares unos con otros

  pero no necesariamente con la misma estructura.

- Su nombre surge de la palabra en inglés “hu**mongo**us” (que significa enorme).
- MongoDB guarda estructuras de datos en documentos tipo JSON (JavaScript Object Notation) con un esquema dinámico.
- Internamente MongoDB almacena los datos en formato BSON (Binary JavaScript Object Notation).
- BSON está diseñado para tener un almacenamiento y velocidad más eficiente.

mongoDB

2007

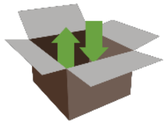
La empresa 10gen lo desarrolla cuando estaba desarrollando una Plataforma cómo servicio (PaaS - Platform as a Service). Similar a Google App Engine.

Bases de Datos Documentales

2009

En este año MongoDB es lanzado como Producto. Es publicado bajo licencia de código abierto AGPL.

Bases de Datos de Propósitos Generales



Bases de Datos De Código Abierto

2011

Se lanza la versión 1.4 considerada como una Base de Datos lista para producción.

2016

Actualmente MongoDB está por la versión 3.2.8 y es la Base de Datos NoSQL con mayor popularidad.

RDBMS	MongoDB
Database instance	MongoDB instance
Database / Schema	Database
Table	Collection
Row	Document
Rowid	_id
Join	Dbref

MongoDB

- Orientada a documentos, dentro de la familia de las bases de datos NoSQL, y desarrollada con código abierto.
- En lugar de tablas, MongoDB guarda estructuras de datos en documentos similares a JSON, con un esquema dinámico (MongoDB utiliza una especificación llamada BSON), haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.
- El esquema dinámico significa, que puede cambiar la estructura o definición del documento, en cualquier momento, sin hacer nada sobre la base de datos.
- En un esquema tradicional de base de datos SQL, el esquema, es decir la definición de la tabla con sus campos y tipos de datos, debe preservarse siempre. En caso de querer cambiarla, hay que ejecutar los comandos correspondientes para realizar su actualización.

Principales Características

- **Consultas Ad hoc:** Búsqueda por campos, consultas de rangos y expresiones regulares. Las consultas pueden devolver un campo específico del documento, pero también pueden ser una función Javascript definida por el usuario.
- **Indexación:** Cualquier campo puede ser indexado, también es posible hacer índices secundarios. Los índices en MongoDB son iguales a los usados en BDs relacionales: lograr “ordenar” la estructura de la base para mejorar su rendimiento sobre los campos indexados.
- **Replicación**: MongoDB soporta el tipo de replicación primario-secundario. Cada grupo de primario y sus secundarios se denomina replica set. El primario puede ejecutar comandos de lectura y escritura. Los secundarios replican los datos del primario y sólo se pueden usar para lectura o para copia de seguridad, pero no se pueden realizar escrituras. Los secundarios tienen la habilidad de poder elegir un nuevo primario en caso de que el primario actual deje de responder.

Principales Características

- **Balanceo de carga**: Se puede escalar de forma horizontal usando el concepto de “shard”. El desarrollador elige una clave de sharding, la cual determina cómo serán distribuidos los datos de una colección. Los datos son divididos en rangos (basado en la clave de sharding) y distribuidos a través de múltiples shard. Cada shard puede ser una réplica set. MongoDB tiene la capacidad de ejecutarse en múltiples servidores, balanceando la carga y/o replicando los datos para poder mantener el sistema funcionando en caso que exista un fallo de hardware. La configuración automática es fácil de implementar bajo MongoDB y pueden agregarse nuevos servidores a MongoDB con el sistema de base de datos funcionando.
- **Agregación**: El framework de agregación está construido como un pipeline (cadena de etapas), en el que los datos van pasando a través de diferentes etapas en las cuales estos datos son modificados, agregados, filtrados y formateados, hasta obtener el resultado deseado. Todo este proceso es capaz de utilizar índices si existieran y se produce en memoria. Asimismo, MongoDB proporciona una función MapReduce que puede ser utilizada para el procesamiento por lotes de datos y operaciones de agregación.

Relaciones Uno a Uno con documentos embebidos

Modelo Normalizado

```
Colección Personas
  { _id: "u0001",
nombre: "Juan Martín Hernandez" }
```

```
Colección Direcciones
{ persona_id: "u0001",
calle: "Malabia 2277",
   ciudad: "CABA",
 provincia: "CABA",
codPostal: "1425" }
```

Si la dirección es un dato frecuentemente consultado junto con el Nombre de la persona, la mejor opción será embeber la dirección en los datos de la persona.

```
Colección Personas
  { _id: "u0001",
nombre: "Juan Martín Hernandez",

direccion:{calle: "Malabia 2277",
           ciudad: "CABA",
         provincia: "CABA",
         codPostal: "1425" }
}
```

Con una sola consulta podríamos recuperar toda la información de una persona.

Relaciones Uno a Muchos
Con Documentos Embebidos

Modelo Normalizado

Colección Personas

```
{ _id: "u0001",
nombre: "Juan Martín Hernandez" }
```

Colección Direcciones

```
{ persona_id: "u0001",
calle: "Malabia 2277",
ciudad: "CABA",
provincia: "CABA",
codPostal: "1425" }

{persona_id: "u0001",
calle: "Av. Santa Fe 3455",
ciudad: "Mar del Plata",
provincia: "Buenos Aires",
codPostal: "7600" }
```

Si las direcciones son un dato frecuentemente consultado junto con el Nombre de la persona, la mejor opción será embeber las direcciones en los datos de la persona.

Colección Personas

```
{ _id: "u0001",
nombre: "Juan Martín Hernandez",

direcciones:[{calle: "Malabia 2277",
ciudad: "CABA",
provincia: "CABA",
codPostal: "1425" },

{calle: "Av. Santa Fe 3455",
ciudad: "Mar del Plata",
provincia: "Buenos Aires",
codPostal: "7600" }
]
}
```

Con una sola consulta podríamos recuperar toda la información de una persona.

Mag. Carlos Adolfo Beltrán Castro

Relaciones Uno a Muchos Con Documentos Referenciados

Colección libros

```
{titulo: "MongoDB: The Definitive Guide",
  autor:[ "K. Chodorow", "M. Dirolf" ],
 fechaPublicacion: ISODate("2010-09-24"),
              paginas: 216,
            lenguaje: "Ingles",
 editor: { nombre: "O'Reilly Media",
            anioFundacion: 1980,
             USAState: "CA" } }

{titulo: "50 Tips and Tricks for MongoDB…",
            autor: "K. Chodorow",
 fechaPublicacion: ISODate("2011-05-06"),
              paginas: 68,
            lenguaje: "Ingles",
 editor: { nombre: "O'Reilly Media",
            anioFundacion:  1980,
            USAState: "CA"    } }
```

Colección Editores

```
{ nombre: "O'Reilly Media",
     anioFundacion: 1980,
       USAState: "CA",
  libros: [987654321,1234567890] }
```

Colección Libros

```
{_id: 987654321
 titulo: "MongoDB: The Definitive Guide",
  autor:[ "K. Chodorow", "M. Dirolf" ],
 fechaPublicacion: ISODate("2010-09-24"),
              paginas: 216,
            lenguaje: "Ingles"}
{_id: 1234567890
titulo: "50 Tips and Tricks for MongoDB…",
            autor: "K. Chodorow",
 fechaPublicacion: ISODate("2011-05-06"),
              paginas: 68,
            lenguaje: "Ingles"}
```

Cuando usamos referencias, el crecimiento de las relaciones determinan donde conviene almacenar la referencia. Por ej. Si el nro. de libros por editor es chico y no crecerá mucho, este modelo podría ser conveniente.

Relaciones Uno a Muchos Con Documentos Referenciados

Colección libros

```
{titulo: "MongoDB: The Definitive Guide",
  autor:[ "K. Chodorow", "M. Dirolf" ],
 fechaPublicacion: ISODate("2010-09-24"),
              paginas: 216,
            lenguaje: "Ingles",
 editor: { nombre: "O'Reilly Media",
             anioFundacion: 1980,
              USAState: "CA" } }

{titulo: "50 Tips and Tricks for MongoDB…",
            autor: "K. Chodorow",
 fechaPublicacion: ISODate("2011-05-06"),
              paginas: 68,
            lenguaje: "Ingles",
 editor: { nombre: "O'Reilly Media",
             anioFundacion:  1980,
             USAState: "CA"    } }
```

Colección Editores

```
{ _id: "oreilly"
  nombre: "O'Reilly Media",
    anioFundacion: 1980,
      USAState: "CA",
}
```

Colección Libros

```
{_id: 987654321
 titulo: "MongoDB: The Definitive Guide",
  autor:[ "K. Chodorow", "M. Dirolf" ],
 fechaPublicacion: ISODate("2010-09-24"),
              paginas: 216,
            lenguaje: "Ingles",
            idEditor: "oreilly"}

{_id: 1234567890
titulo: "50 Tips and Tricks for MongoDB…",
            autor: "K. Chodorow",
 fechaPublicacion: ISODate("2011-05-06"),
              paginas: 68,
            lenguaje: "Ingles",
            idEditor: "oreilly"}
```

En cambio si queremos evitar Arreglos mutables y crecientes podemos implementar una referencia al editor dentro de cada libro.

Logging de Eventos

•las bases de datos basadas en documentos puede loguear cualquier clase de eventos y almacenarlos con sus diferentes estructuras.

• Pueden funcionar como un repositorio central de logueo de eventos.

CMS, blogging

• su falta de estructura predefinida hace que funcionen bien para este tipo de aplicaciones.

Web-analytics / Real-Time analytics

• Almacenar cantidad de vistas a una página o visitantes únicos.

Commerce

• A menudo requieren tener esquemas flexibles para los productos y órdenes

INSTALACIÓN
Y
CONFIGURACIÓN

Mag. Carlos Adolfo Beltrán Castro

1. Instalar MongoDB

2. Actualizar variable Path

    - Buscar el directorio donde se instaló MongoDb (por ejemplo C:\Program Files\MongoDB\Server\5.0\bin)
    - Añadir MongoDb a la variable Path (Inicio > Equipo > Propiedades del Sistema > Opciones avanzadas)

3. Crear una carpeta (con los permisos adecuados) para guardar la base de datos

    - C:/Data/Db es la carpeta default para MongoDb
    - Si quisiéramos tener otro path debemos ejecutar desde consola: mongod --dbpath ruta/nueva-a/la-carpeta-db

4. Abrir conexión desde Consola de Windows (Símbolo del Sistema) ejecutando el comando mongod

5. Mantener la consola de conexión abierta y abrir una nueva consola para operar sobre la base

UTILIZANDO MONGODB

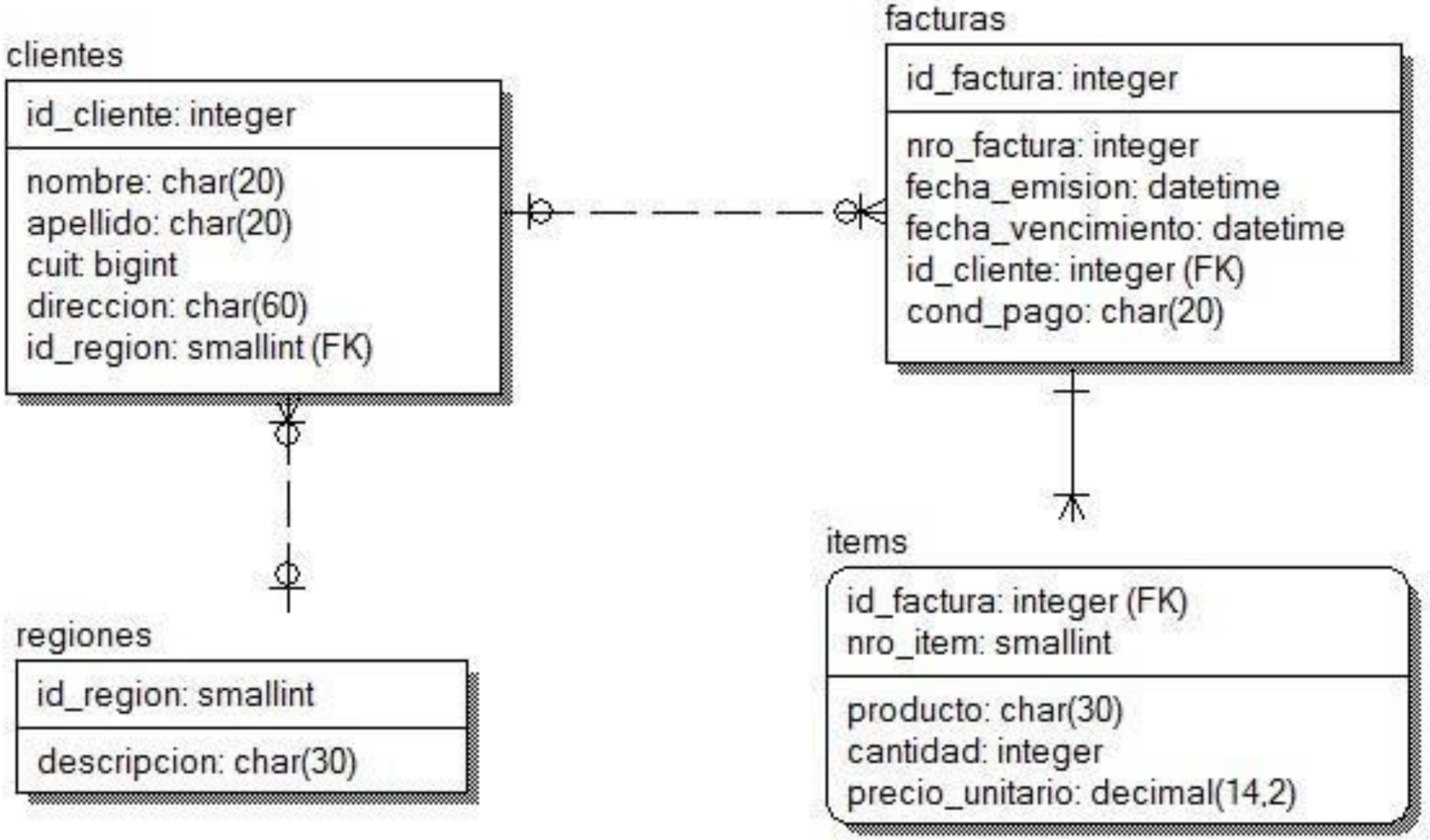
Mag. Carlos Adolfo Beltrán Castro

Instalación de MongoDB

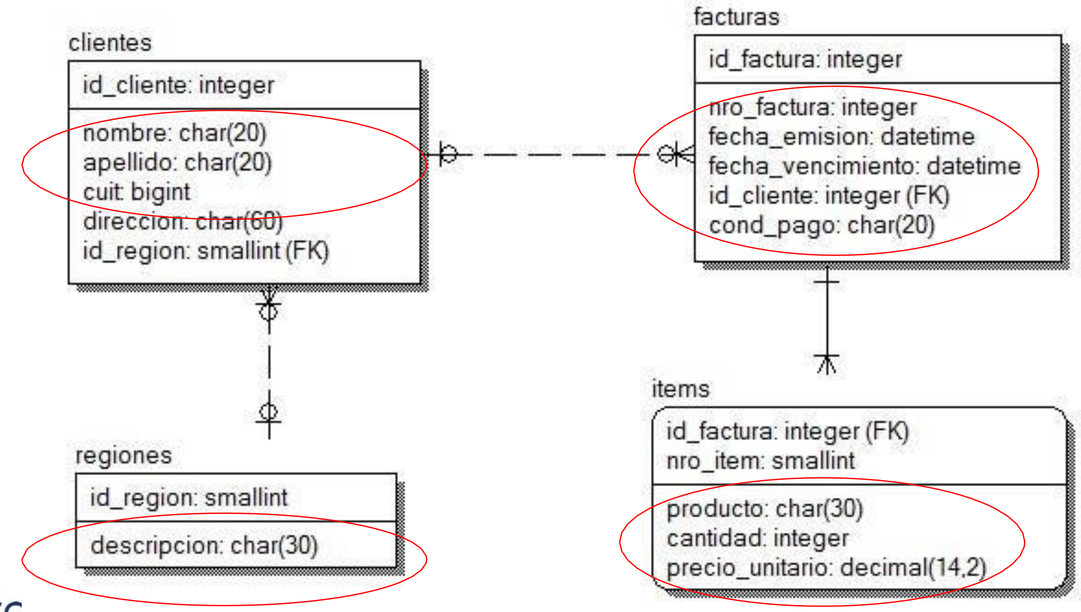
- En este link **https://docs.mongodb.com/manual/installation/** encontrarás los pasos para configurar MongoDB Community Edition, la versión gratuita, en tu computadora, de acuerdo a tu propio sistema operativo, y ejecutarlo localmente.
- Otra opción es crear una cuenta de Mongo Atlas aquí **https://www.mongodb.com/cloud/atlas?lang=es-es**. Hay una versión gratuita que es más que suficiente para comenzar a trabajar.
- Otra versión es una interfaz gráfica llamada **Mongo Compass Community**, que es de uso gratuito y se encuentra en constante desarrollo y actualización. Puedes descargarlo desde aquí **https://www.mongodb.com/products/compass**.

Instalación de MongoDB en Windows

- En este link **https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/** encontrarás los pasos para instalar MongoDB Community Edition en windows.


clientes
id_cliente: integer
nombre: char(20)
apellido: char(20)
cuit: bigint
direccion: char(60)
id_region: smallint (FK)
facturas
id_factura: integer
nro_factura: integer
fecha_emision: datetime
fecha_vencimiento: datetime
id_cliente: integer (FK)
cond_pago: char(20)
regiones
id_region: smallint
descripcion: char(30)
items
id_factura: integer (FK)
nro_item: smallint
producto: char(30)
cantidad: integer
precio_unitario: decimal(14,2)

Armaremos un modelo que contenga la información de las facturas y todos sus ítems, detallando el nombre, apellido, cuit y región del cliente al que se le emitió la factura, para poder realizar consultas desde un portal de facturas de la forma más performante posible.



UTILIZACIÓN DE MONGODB DESDE LA LÍNEA DE COMANDOS

Mag. Carlos Adolfo Beltrán Castro

Inicio de servidor mongo por consola:

C:>mongod

Inicio de cliente mongo por consola:

C:>mongo

Inicio Básico con MongoDb

https://docs.mongodb.com/manual/tutorial/getting-started/

Ver bases de datos

show dbs

Ver colecciones

show collections

Operaciones Básicas en Shell

- Iniciar mongodb con mongod como Administrador
- Iniciar el Shell de mongo en otro Shell como Admiinistrador.
- Operaciones básicas

```
> x = 200
200
> x
200
> x / 5
40
> x * 2
400
> Math.sin(Math.PI / 2)
1
> new Date()
ISODate("2021-11-16T02:28:58.644Z")
> new Date("2021/08/11")
ISODate("2021-08-11T05:00:00Z")
> function factorial(n) { if (n<=1) return n; return n * factorial(n-1)}
> factorial
function factorial(n) { if (n<=1) return n; return n * factorial(n-1)}
> factorial(5)
120
```

Base de datos en uso: db
Mostrar las BDs: show dbs
Ayuda: help

```
> db
examples
> show dbs
admin     0.000GB
config    0.000GB
examples  0.000GB
local     0.000GB
tienda    0.000GB
> help
        db.help()                    help on db methods
        db.mycoll.help()             help on collection methods
        sh.help()                    sharding helpers
        rs.help()                    replica set helpers
        help admin                   administrative help
        help connect                 connecting to a db help
        help keys                    key shortcuts
        help misc                    misc things to know
        help mr                      mapreduce

        show dbs                     show database names
        show collections             show collections in current database
        show users                   show users in current database
        show profile                 show most recent system.profile entries with time >= 1ms
        show logs                    show the accessible logger names
        show log [name]              prints out the last segment of log in memory, 'global' is default
        use <db_name>                set current database
        db.mycoll.find()             list objects in collection mycoll
        db.mycoll.find( { a : 1 } )  list objects in mycoll where a == 1
        it                           result of the last line evaluated; use to further iterate
        DBQuery.shellBatchSize = x   set default number of items to display on shell
        exit                         quit the mongo shell
> _
```

Operaciones Básicas en Shell

Ayuda sobre BDs:
db.help()

```
> 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.commandHelp(name) returns the help for the command
        db.createUser(userDocument)
        db.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase(writeConcern)
        db.dropUser(username)
        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.getLastError() - just returns the err msg string
        db.getLastErrorObj() - return full status object
        db.getLogComponents()
        db.getMongo() get the server connection object
        db.getMongo().setSecondaryOk() allow queries on a replication secondary server
        db.getName()
        db.getProfilingLevel() - deprecated
        db.getProfilingStatus() - returns if profiling is on and slow threshold
        db.getReplicationInfo()
        db.getSiblingDB(name) get the db at the same server as this one
        db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
        db.hostInfo() get details about the server's host
        db.isMaster() check replica primary status
        db.hello() check replica primary status
        db.killOp(opid) kills the current operation in the db
        db.listCommands() lists all the db commands
        db.loadServerScripts() loads all the scripts in db.system.js
        db.logout()
```

Operaciones Básicas en Shell

```
> show dbs
admin     0.000GB
config    0.000GB
examples  0.000GB
local     0.000GB
tienda    0.000GB
> db
examples
> use tiendaweb
switched to db tiendaweb
> db
tiendaweb
> dbs
uncaught exception: ReferenceError: dbs is not defined :
@(shell):1:1
> show dbs
admin     0.000GB
config    0.000GB
examples  0.000GB
local     0.000GB
tienda    0.000GB
> db.productos.insert({"name":"laptop"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin      0.000GB
config     0.000GB
examples   0.000GB
local      0.000GB
tienda     0.000GB
tiendaweb  0.000GB
> show collections
productos
```

Mag. Carlos Adolfo Beltrán Castro

Operaciones Básicas en Shell

```
> db
tiendaweb
> db.dropDatabase()
{ "ok" : 1 }
> use tiendaweb
switched to db tiendaweb
> show collections
> db.createCollection("users")
{ "ok" : 1 }
> db.createCollection("products")
{ "ok" : 1 }
> db.createCollection("customers")
{ "ok" : 1 }
> show collections
customers
products
users
> db.products.drop()
true
> show collections
customers
users
>
```

Operaciones Básicas en Shell

```
> db.users.drop()
true
> db.customers.drop()
true
```

```
> db.products.find()
{ "_id" : ObjectId("61931fc5f0ec07f73d1c4251"), "nombre" : "laptop", "precio" : 40.5, "activo" : false, "fecha_cr
eacion" : ISODate("1999-12-12T05:00:00Z"), "somedata" : [ 1, "a", [ ] ], "proveedor" : { "name" : "Dell", "versio
n" : "xps", "location" : { "ciudad" : "China", "address" : "pppllsak" } } }
> db.products.find().pretty()
{
        "_id" : ObjectId("61931fc5f0ec07f73d1c4251"),
        "nombre" : "laptop",
        "precio" : 40.5,
        "activo" : false,
        "fecha_creacion" : ISODate("1999-12-12T05:00:00Z"),
        "somedata" : [
                1,
                "a",
                [ ]
        ],
        "proveedor" : {
                "name" : "Dell",
                "version" : "xps",
                "location" : {
                        "ciudad" : "China",
                        "address" : "pppllsak"
                }
        }
}
```

Mag. Carlos Adolfo Beltrán Castro

Operaciones Básicas en Shell

```
> use mystore
switched to db mystore
> db.createCollection("products")
{ "ok" : 1 }
> show collections
products
> db.products.insert({"name": "keyboard"})
WriteResult({ "nInserted" : 1 })
> db.products.find()
{ "_id" : ObjectId("5d5c5c0af358bf39a9e50c9d"), "name" : "keyboard" }
> db.products.find().pretty()
{ "_id" : ObjectId("5d5c5c0af358bf39a9e50c9d"), "name" : "keyboard" }
> db.products.insert({"name": "laptop", "price": 999.99})
WriteResult({ "nInserted" : 1 })
> db.products.find()
{ "_id" : ObjectId("5d5c5c0af358bf39a9e50c9d"), "name" : "keyboard" }
{ "_id" : ObjectId("5d5c5c59f358bf39a9e50c9e"), "name" : "laptop", "price" : 999.99 }
```

Operaciones Básicas en Shell

```
db.products.insert([
    {
        "name": "mouse",
        "description": "razer mouse",
        "tags": ["computers", "gaming"],
        "quantity": 14,
        "created_at": new Date()
    },
    {
        "name": "monitor",
        "description": "lg monitor",
        "tags": ["computers", "gaming"],
        "quantity": 14,
        "created_at": new Date()
    }
])
```

Mag. Carlos Adolfo Beltrán Castro

Operaciones Básicas en Shell

```
> db.products.insert([
...     {
...         "name": "mouse",
...         "description": "razer mouse",
...         "tags": ["computers", "gaming"],
...         "quantity": 14,
...         "created_at": new Date()
...     },
...     {
...         "name": "monitor",
...         "description": "lg monitor",
...         "tags": ["computers", "gaming"],
...         "quantity": 3,
...         "created_at": new Date()
...     }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Mag. Carlos Adolfo Beltrán Castro

Operaciones Básicas en Shell

```
> db.products.find().pretty()
{ "_id" : ObjectId("5d5c5c0af358bf39a9e50c9d"), "name" : "keyboard" }
{
        "_id" : ObjectId("5d5c5c59f358bf39a9e50c9e"),
        "name" : "laptop",
        "price" : 999.99
}
{
        "_id" : ObjectId("5d5c5dc7f358bf39a9e50c9f"),
        "name" : "mouse",
        "description" : "razer mouse",
        "tags" : [
                "computers",
                "gaming"
        ],
        "quantity" : 14,
        "created_at" : ISODate("2019-08-20T20:53:27.903Z")
}
{
        "_id" : ObjectId("5d5c5dc7f358bf39a9e50ca0"),
        "name" : "monitor",
        "description" : "lg monitor",
        "tags" : [
                "computers",
                "gaming"
        ],
        "quantity" : 3,
        "created_at" : ISODate("2019-08-20T20:53:27.903Z")
}
```

Mag. Carlos Adolfo Beltrán Castro

Operaciones Básicas en Shell
Búsqueda de un documento

```
> db.products.find({"price": 999.99})
{ "_id" : ObjectId("5d5c5c59f358bf39a9e50c9e"), "name" : "laptop", "price" : 999.99 }
> db.products.find({"price": 999.99}).pretty()
{
        "_id" : ObjectId("5d5c5c59f358bf39a9e50c9e"),
        "name" : "laptop",
        "price" : 999.99
}
>
```

Ver todos los comandos:

db.facturas.help()

Ejemplos:

Ver todos los documentos:

db.facturas.find()

Cantidad de documentos en la colección:

db.facturas.count()

Espacio ocupado por los documentos de la colección:

db.facturas.dataSize()

Buscar documento para cliente con determinado apellido

db.facturas.find({"cliente.apellido":"Malinez"})

Consultar los primeros dos documentos

db.facturas.find().limit(2)

Consultar documentos salteando los primeros dos documentos

db.facturas.find().skip(2)

Visualización mejorada

db.facturas.find().pretty()

Consultar dos documentos, salteando los dos primeros documentos de una colección, mostrándolos en un modo mejorado.

db.facturas.find().limit(2).skip(2).pretty()

Consultar documentos sólo mostrando algunos datos

db.facturas.find({"cliente.apellido":"Malinez"}, {"cliente.cuit":1, "cliente.region":1})

Buscar documento para cliente con dos criterios

Para Zavasi teníamos dos documentos: db.facturas.find({"cliente.apellido":"Zavasi"}).pretty()

Agregamos un criterio: db.facturas.find({"cliente.apellido":"Zavasi", "nroFactura":1001.0}).pretty()

Ordenamiento en forma ascendente

db.facturas.find().sort({nroFactura:1})

Ordenamiento en forma descendente

db.facturas.find().sort({nroFactura:-1})

Busca la cantidad de facturas cuyo Nro. de Factura sea mayor que 1465

Operadores $

```
db.facturas.find( { nroFactura : {$gt:1465} } ).count()
```

$gt
$gte
$lt
$lte
$not
$or
$in
$nin
$exist
$regex

Busca las facturas cuya fecha de emisión sea mayor o igual al 24/02/2014.

```
db.facturas.find({fechaEmision:{$gte: ISODate("2014-02-24T00:00:00Z")} } )
```

El método insert tiene la siguiente sintaxis:

Evalúa si existe un próximo documento. Devuelve True o False.

```
        db.collection.insert
( <document or array of documents>,
 { writeConcern: <document>,
   ordered: <boolean> } )
```

writeconcern	Es opcional, lo veremos en la parte de consistencia.
Ordered	lo vemos en un par de slides

Ejemplo, inserción de un documento sin _id:

```
db.facturas.insert({nroFactura:30003,codPago:"CONTADO"})
```

_id: Document Id único autogenerado

```
> db.facturas.insert({nroFactura:30003,codPago:"CONTADO"})
WriteResult({ "nInserted" : 1 })
>
>
> db.facturas.find({nroFactura:30003})
{ "_id" : ObjectId("5459a129cc19250561ad5f82"), "nroFactura" : 30003, "codPago"
: "CONTADO" }
```

Ejemplo, inserción de un documento con _id:

```
db.facturas.insert({_id:23094776, nroFactura:30004,codPago:”CONTADO”})
```

```
> db.facturas.insert({_id:23094776,nroFactura:30004,codPago:"CONTADO"})
WriteResult({ "nInserted" : 1 })
>
>
> db.facturas.find({nroFactura:30004})
{ "_id" : 23094776, "nroFactura" : 30004, "codPago" : "CONTADO" }
```

Al crear una colección, el motor de BD crea un índice único sobre el atributo _id.

```
> db.facturas.insert({_id:23094776,nroFactura:30004,codPago:"30dsFF"})
WriteResult({
        "nInserted" : 0,
        "writeError" : {
                "code" : 11000,
                "errmsg" : "insertDocument :: caused by :: 11000 E11000 duplicat
e key error index: finanzas.facturas.$_id_  dup key: { : 23094776.0 }"
        }
})
```

Operación Remove

Sintaxis

```
db.<collection_name>.remove({criterio_de_eliminación})
```

Esta operación eliminará los documentos que cumplan con el criterio definido.

Warning: Remove es una operación de tipo multi-documento!!

Recomendación: Es conveniente antes de borrar hacer un find o un count para asegurarse lo que quiero borrar.

Ejemplo 1 – Borrado de TODOS LOS DOCUMENTOS de una colección

`db.accesos.remove({})`

Elimina **TODOS LOS ELEMENTOS** de una colección.

```
> db.accesos.remove({})
WriteResult({ "nRemoved" : 3 })
>
> db.accesos.find()
```

Ejemplo 2 – Remove por clave primaria

```
db.updtst.remove({_id:100})
```

Elimina el documento cuyo _id sea 100 de la colección **updtst**.

```
> db.updtst.remove({_id:100})
WriteResult({ "nRemoved" : 1 })
>
> db.updtst.find()
{ "_id" : 300, "items" : [ 88, 99, 97 ] }
{ "_id" : 200 }
```

Ejemplo 3 – Remove por un criterio con múltiples documentos que aplican

```
db.updtst.remove({items:88})
```

```
> db.updtst.find()
{ "_id" : 300, "items" : [ 88, 99, 97 ] }
{ "_id" : 500, "items" : 88 }
{ "_id" : 200 }
{ "_id" : 400, "items" : [ 77, 88 ] }
> db.updtst.remove({items:88})
WriteResult({ "nRemoved" : 3 })
> db.updtst.find()
{ "_id" : 200 }
>
```

Permite modificar uno o más documentos de una colección. Por default modifica sólo un documento.

```
db.coleccion.update ( {clausula_where},
                      {documento_o_expresión_a_modificar},
                      { upsert, multi, writeconcern}
                    )
```

upsert (true o false)Si está configurado en “True” significa que realizará un update si existe un documento que concuerda con el criterio, o un insert si no existe algún documento que concuerde con el criterio. El valor default es “false”, en este caso no realiza un insert cuando no existe documento que concuerde con el criterio.

multi (true o false) Es opcional. Si es configurado en true, el update realiza la actualización de multiples documentos que concuerdan con el criterio cláusula_where. Si es configurado en false, modifica solo un documento. El valor default es false. Sólo actúa en updates parciales con operadores $.

writeconcern Es opcional, lo veremos en la parte de consistencia.

Update Totales/Completos

Se realiza el update del documento completo, reemplazando el mismo.

Update Parciales

Operadores

Operadores sobre cualquier atributo

$set	Permite modificar el valor de un atributo, o agregar un nuevo atributo al documento.
$unset	Permite eliminar un atributo de un documento.
$inc	Incrementa o decrementa el valor de un atributo (n ó –n)

Operadores sobre Arrays

$push	Agrega un elemento a un Array o crea un Array con un elemento.
$addtoSet	Agrega un elemento al Array solo si no existe en el Array.
$pushAll	Agrega varios elementos a un Array con los valores indicados o crea un Array con esos elementos. *(Operación Múltiple)*
$pop	Elimina un elemento de un Array por sus extremos, permitiendo eliminar el primer elemento (-1)o el último (1).
$pull	Elimina todos los elementos de un Array que contengan el valor indicado.
$pullAll	Elimina todos los elementos de un Array que contengan alguno de los valores indicados. *(Operación Múltiple)*

Update Totales/Completos

```
db.updtst.update({x:2},{"x" : 2, "y" : 999 })
```

Este comando reemplaza el primer documento encontrado por con valor x:2 por este otro en donde el elemento y:999, no tengo el control de cuál estoy modificando, lo correcto era modificar poniendo en el criterio el _id.

```
> db.updtst.update({x:2},{"x" : 2, "y" : 999 })
> db.updtst.find()
{ "_id" : ObjectId("536a8240793253ebed598065"), "x" : 1, "y" : 999 }
{ "_id" : ObjectId("536a8245793253ebed598066"), "x" : 2, "y" : 999 }
{ "_id" : ObjectId("536a8248793253ebed598067"), "x" : 2, "y" : 100 }
{ "_id" : ObjectId("536a824b793253ebed598068"), "x" : 2, "y" : 300 }
{ "_id" : ObjectId("536a8250793253ebed598069"), "x" : 3, "y" : 100 }
{ "_id" : ObjectId("536a8254793253ebed59806a"), "x" : 3, "y" : 200 }
{ "_id" : ObjectId("536a8257793253ebed59806b"), "x" : 3, "y" : 300 }
```

Update Parciales

Ejemplo 1 – Operador $set – Modificación de un valor de un atributo existente

Dado el siguiente documento:

```
> db.updtst.insert({_id:100,x:10,y:100})
```

```
db.updtst.update({_id:100},{$set : {x:100}})
```

Realizará una modificación del valor de atributo x a 100

```
> db.updtst.find({_id:100})
{ "_id" : 100, "x" : 100, "y" : 100 }
```

Update Parciales

Otro Ejemplo – Operador $set – Opción multi – Agregar un atributo en todos los documentos

```
db.updtst.update({x:2},{$set : {z:"NUEVO"}},{multi:true})
```

Este reemplaza en TODOS los documentos encontrados con valor x:2 agregando el atributo z:"NUEVO"

```
> db.updtst.update({x:2},{$set : {z:"NUEVO"}},{multi:true})
> db.updtst.find({x:2})
{ "_id" : ObjectId("536a8245793253ebed598066"), "x" : 2, "y" : 999, "z" : "NUEVO
" }
{ "_id" : ObjectId("536a8248793253ebed598067"), "x" : 2, "y" : 100, "z" : "NUEVO
" }
{ "_id" : ObjectId("536a824b793253ebed598068"), "x" : 2, "y" : 300, "z" : "NUEVO
" }
```