WorkShop MongoDB Introductorio

Intro a BigData - MongoDB

DEFINIENDO BIG DATA

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

DBlandIT®
databases + science +IT

DEFINIENDO BIG DATA

DBlandIT®
databases + science +IT

DEFINIENDO BIG DATA

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)

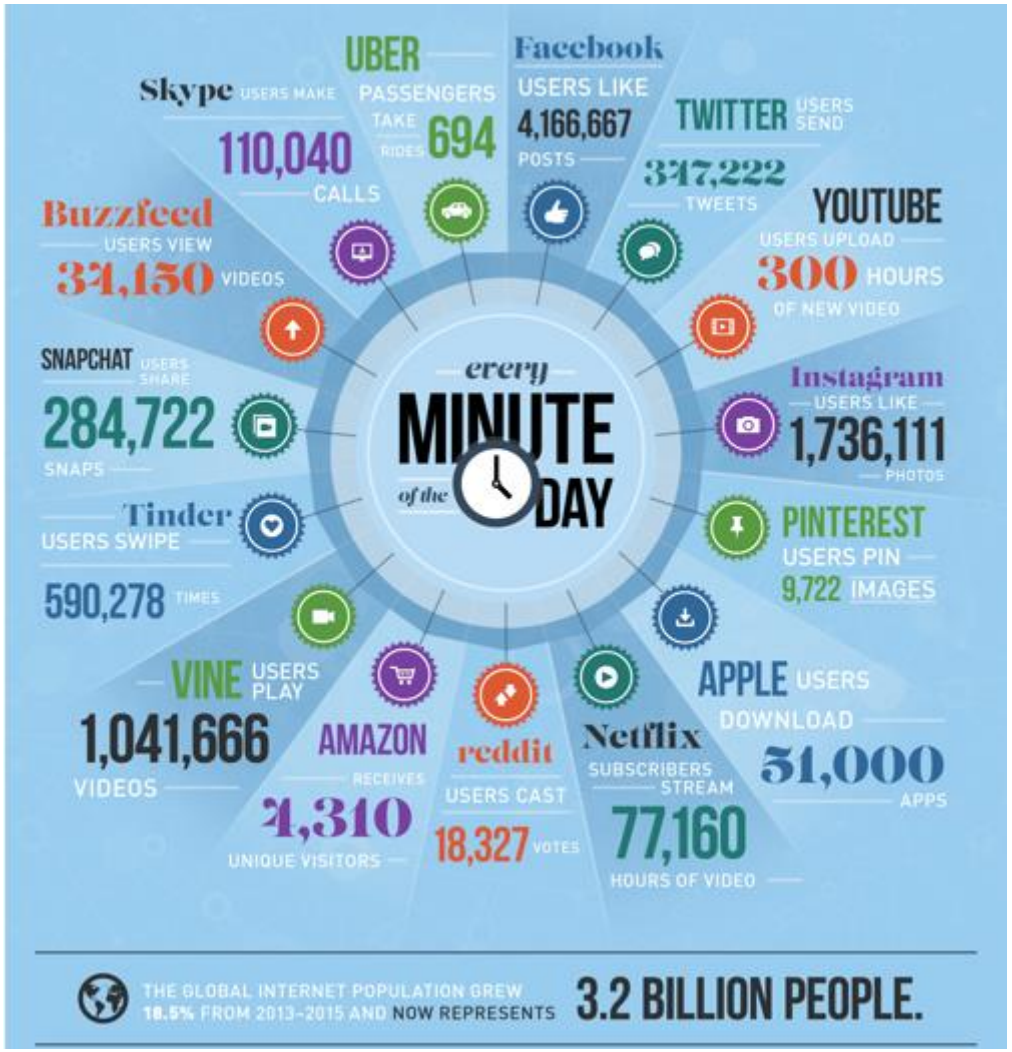
BIG DATA

- Volumen
- Velocidad
- Variedad
- "Veracidad"

DBlandIT®
databases + science + IT

DEFINIENDO BIG DATA

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

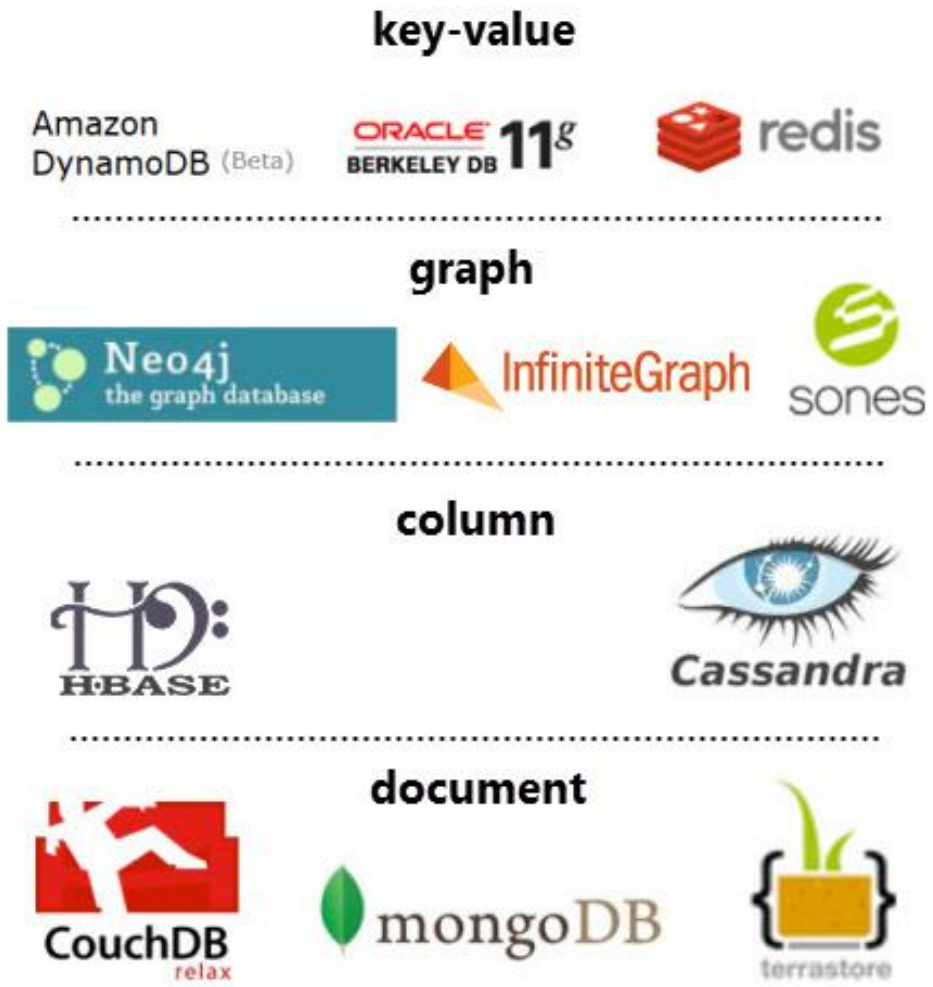
DBlandIT®
databases + science +IT

Introducción a

Bases de Datos NoSQL

DBlandIT®
databases + science +IT

NOSQL DATABASE TYPES



¿ 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.

¿ Qué es la Pesistencia Políglota ?

Utilizar dentro de un mismo ambiente o aplicación un conjunto de bases de datos, que colabora, cada una en lo que es más importante.

DBlandIT®
databases + science +IT

NOSQL – DB-ENGINES.COM

309 systems in ranking, August 2016

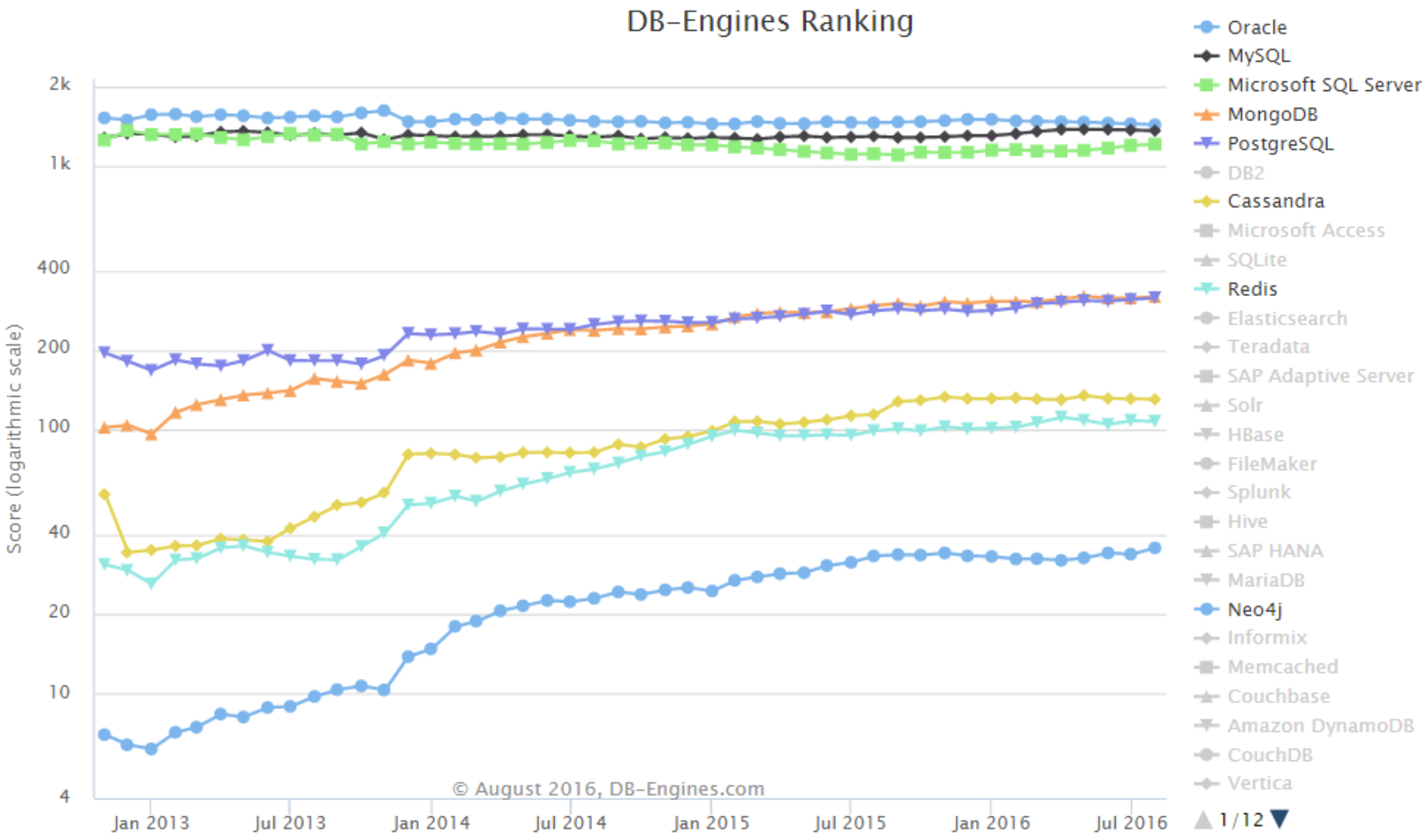
Rank Aug 2016	Rank Jul 2016	Rank Aug 2015	DBMS	Database Model	Score Aug 2016	Score Jul 2016	Score Aug 2015
1.	1.	1.	Oracle	Relational DBMS	1427.72	-13.81	-25.30
2.	2.	2.	MySQL	Relational DBMS	1357.03	-6.25	+65.00
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1205.04	+12.16	+96.39
4.	4.	4.	MongoDB	Document 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
HOY

DBlandIT®
databases + science + IT

NOSQL – DB-ENGINES.COM

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



DBlandIT®
databases + science +IT

Document Based

INTRODUCCIÓN A MONGODB

Document Based

- Las bases de datos almacenan y recuperan documentos que pueden ser XML, JSON, BSON, etc.
- Estos documentos son estructuras de datos en forma de árbol jerárquico que consisten de mapas, colecciones, y valores escalares.
- Los documentos almacenados son similares unos con otros pero no necesariamente con la misma estructura.
- MongoDB, CouchBase, CouchDB, Rethink DB, RavenDB,…

MongoDB

- 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.

DBlandIT®
databases + science +IT

El Origen

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.

DBlandIT®
databases + science +IT

MongoDB Características

- JSON Document Model con Esquema Dinámico
- Particionamiento automático (Auto-Sharding) para Escalamiento Horizontal
- Búsquedas de texto (Full Text Search)
- Aggregation Framework y MapReduce Nativo o con Hadooop.
- Soporte de Indices Completo y flexible
- Consultas Complejas.
- Soporta Replicación para Alta Disponibilidad.
- Manejo de Seguridad Avanzada
- Almacenamiento de archivos de gran tamaño en su file system interno GridFS.

DBlandIT®
databases + science +IT

Terminología RDBMS vs. Document Based (MongoDB)

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

Modelado de Relaciones entre Documentos

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.

DBlandIT®
databases + science +IT

Modelado de Relaciones entre Documentos

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.

DBlandIT® databases + science + IT

Modelado de Relaciones entre Documentos

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.

DBlandIT®
databases + science +IT

Modelado de Relaciones entre Documentos

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.

DBlandIT® databases + science + IT

En qué casos usarlas ?

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.

E-Commerce:

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

¿ En qué casos NO usarlas ?

Transacciones Complejas con diferentes operaciones

- no están soportadas, salvo en RavenDB.

Consultas contra estructuras de agregados variables.

- que los datos se almacenen con cualquier estructura no implica que sea óptimo consultar por cualquier clave. Si los agregados varían entre sí, las consultas debieran variar también. Puede llevar a normalizar los datos, que no es lo que queremos.

MongoDB – CRUD y Caso Práctico

Comenzamos con MongoDB.

Caso Práctico

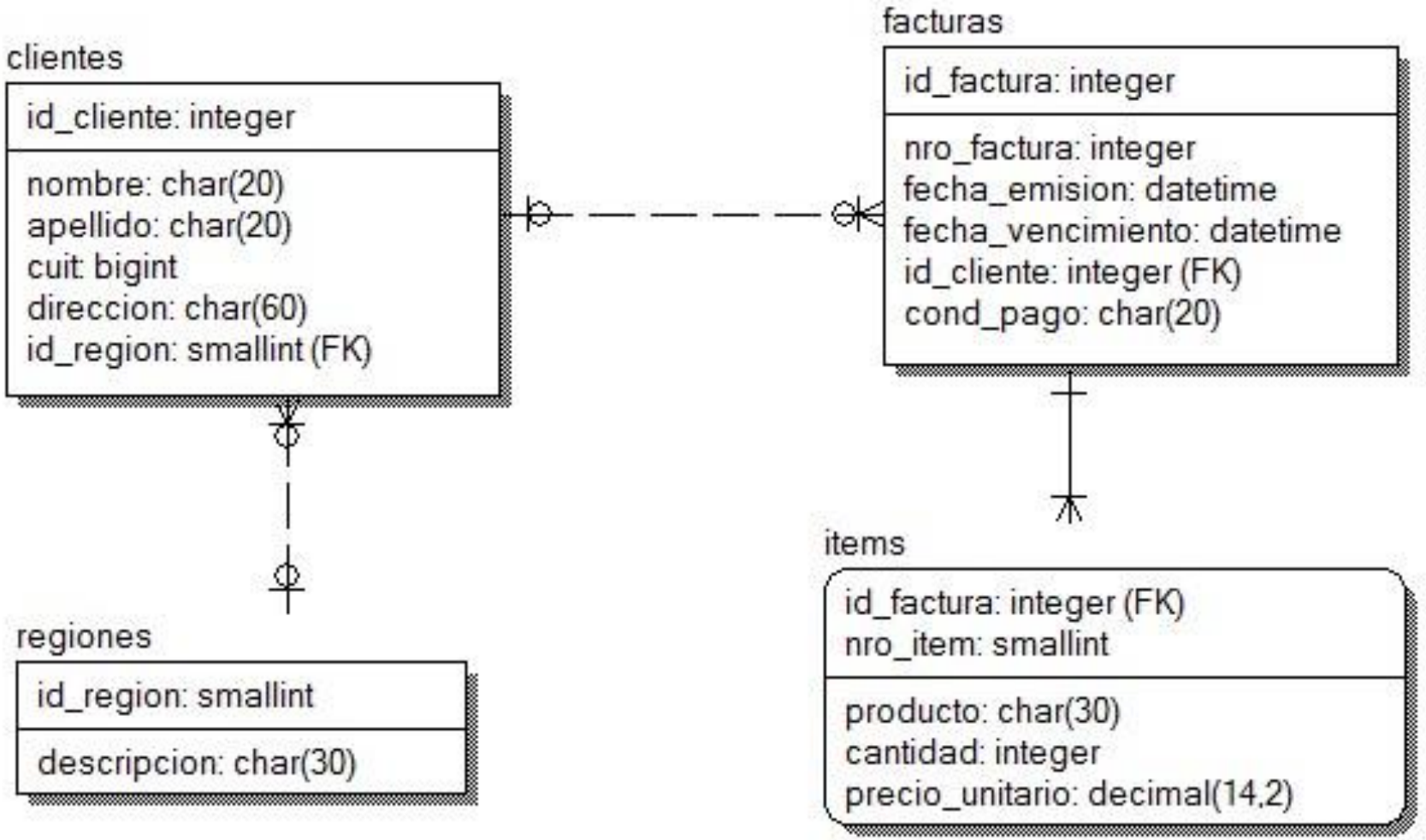
Modelado

CRUD Básico

Ejercicios

DBlandIT®
databases + science +IT

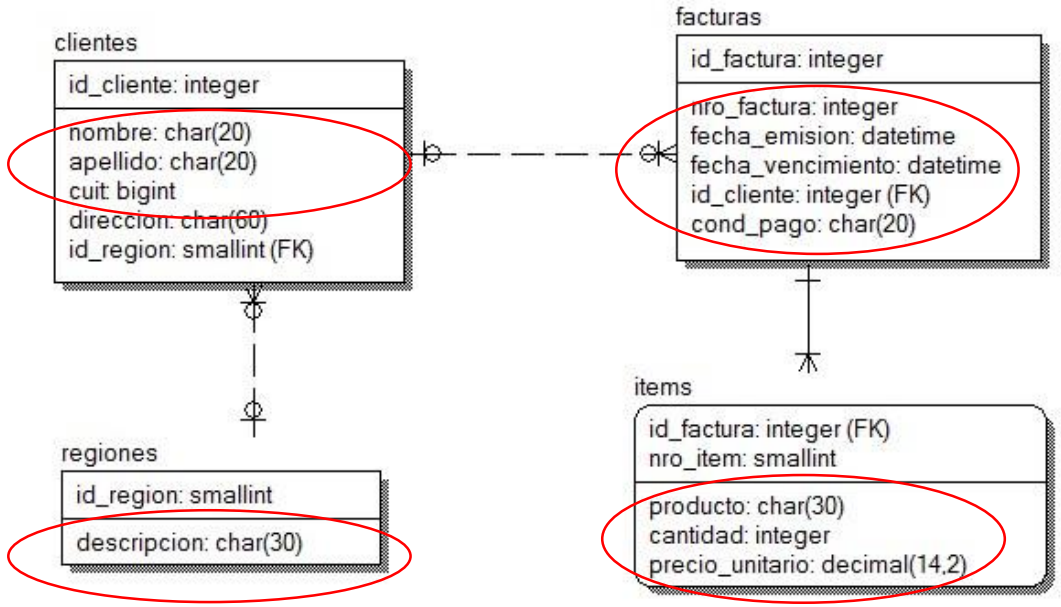
Caso Práctico



DBlandIT®
databases + science + IT

Caso Práctico

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.



DBlandIT®
databases + science +IT

Caso Práctico

```
{ “_id”: ObjectID(“d9d9d9d9d999999999”),
   nroFactura: 9999999,
   fechaEmision: ISODate(“yyyy-mm-ddThh:mm:ssZ”),
   fechaVencimiento: ISODate(“yyyy-mm-ddThh:mm:ssZ”),
   condPago: “XXXXXXXX”,
  “cliente”:{ nombre: “XXXXXX”,
                apellido: “XXXXXXXX”,
                cuit:9999999999999,
                region: “CABA”
              },
  “items”:[{producto:”XXXXXXXX”, cantidad: 999, precio:99.99} ,
            {producto:”XXXXXXXX”, cantidad: 999, precio:99.99}
           ]
}
```

DBlandIT®
databases + science +IT

Operaciones sobre una Colección

Instrucción es sobre una Colección

db.coleccion.help()

```
> db.ordenes.help()
DBCollection help
        db.ordenes.find().help() - show DBCursor help
        db.ordenes.count()
        db.ordenes.copyTo(newColl) - duplicates collection by copying all docume
nts to newColl; no indexes are copied.
        db.ordenes.convertToCapped(maxBytes) - calls {convertToCapped:'ordenes',
 size:maxBytes}} command
        db.ordenes.dataSize()
        db.ordenes.distinct( key ) - e.g. db.ordenes.distinct( 'x' )
        db.ordenes.drop() drop the collection
        db.ordenes.dropIndex(index) - e.g. db.ordenes.dropIndex( "indexName" ) o
r db.ordenes.dropIndex( { "indexKey" : 1 } )
        db.ordenes.dropIndexes()
        db.ordenes.ensureIndex(keypattern[,options]) - options is an object with
 these possible fields: name, unique, dropDups
        db.ordenes.reIndex()
        db.ordenes.find([query],[fields]) - query is an optional query filter. f
ields is optional set of fields to return.
                                                        e.g. db.ordenes.find( {x:7
```

DBlandIT®
databases + science +IT

Consultando una Colección – Criterios de Selección

Buscar documentos que contengan precios de items sean iguales a 490, mostrando sólo los atributos _id, nro. de factura y array de item. El atributo _id lo muestra por default.

```
db.facturas.find( { "item.precio":490 } ,{nroFactura:1,item:1})
```

```
>
> db.productos.find( { "item.precio":490 } ,{nroFactura:1,item:1})
> db.facturas.find( { "item.precio":490 } ,{nroFactura:1,item:1})
{ "_id" : ObjectId("53685dbb2baf7b93f61df564"), "nroFactura" : 1447, "item" : [
        {       "producto" : " CORREA 12mm",    "cantidad" : 11,        "precio"
 : 18 },        {       "producto" : "TALADRO 12mm",    "cantidad" : 1,
"precio" : 490 } ] }
{ "_id" : ObjectId("53685dbb2baf7b93f61df567"), "nroFactura" : 1450, "item" : [
        {       "producto" : "TUERCA 2mm",      "cantidad" : 2,         "precio"
 : 60 },        {       "producto" : "TALADRO 12mm",    "cantidad" : 1,
"precio" : 490 },       {       "producto" : "TUERCA 5mm",      "cantidad" : 15,
        "precio" : 90 } ] }
{ "_id" : ObjectId("53685dbb2baf7b93f61df569"), "nroFactura" : 1452, "item" : [
        {       "producto" : "SET HERRAMIENTAS",        "cantidad" : 1,
"precio" : 700 },       {       "producto" : "TALADRO 12mm",    "cantidad" : 1,
        "precio" : 490 } ] }
```

DBlandIT®
databases + science +IT

Consultando una Colección

```
> db.facturas.find().limit(2).skip(2).pretty()
{
        "_id" : ObjectId("53685dbb2baf7b93f61df564"),
        "nroFactura" : 1447,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-02-20T00:00:00Z"),
        "condPago" : "CONTADO",
        "cliente" : {
                "nombre" : "Marina",
                "apellido" : "Malinez",
                "cuit" : 2740488484,
                "region" : "CENTRO"
        },
        "item" : [
                {
                        "producto" : " CORREA 12mm",
                        "cantidad" : 11,
                        "precio" : 18
                },
                {
                        "producto" : "TALADRO 12mm",
                        "cantidad" : 1,
                        "precio" : 490
                }
        ]
}
{
        "_id" : ObjectId("53685dbb2baf7b93f61df565"),
        "nroFactura" : 1448,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-03-22T00:00:00Z"),
        "condPago" : "30 Ds FF",
        "cliente" : {
                "nombre" : "Martin",
                "apellido" : "Zavasi",
                "cuit" : 2038373771,
                "region" : "CABA"
        },
        "item" : [
                {
                        "producto" : "CORREA 10mm",
                        "cantidad" : 2,
                        "precio" : 134
                }
        ]
}
> _
```

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

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

DBlandIT®
databases + science + IT

Consultando una Colección – Criterios de Selección

Buscar documentos cuya número de factura sea igual a 1450 y la condición de pago sea igual a "CONTADO".
Mostrando todos sus atributos.

```
db.facturas.find( { nroFactura : 1450, condPago:"CONTADO" } ,{})
```

```
>
> db.facturas.find( { nroFactura : 1450, condPago:"CONTADO" } ,{})
{ "_id" : ObjectId("53685dbb2baf7b93f61df567"), "nroFactura" : 1450, "fechaEmisi
on" : ISODate("2014-02-24T00:00:00Z"), "fechaVencimiento" : ISODate("2014-02-24T
00:00:00Z"), "condPago" : "CONTADO", "cliente" : { "nombre" : "Juan Manuel", "ap
ellido" : "Manoni", "cuit" : 2029889382, "region" : "NEA" }, "item" : [
{       "producto" : "TUERCA 2mm",      "cantidad" : 2,          "precio" : 60 },
        {       "producto" : "TALADRO 12mm",    "cantidad" : 1,          "precio"
 : 490 },       {       "producto" : "TUERCA 5mm",      "cantidad" : 15,
"precio" : 90 } ] }
>
```

DBlandIT®
databases + science +IT

Consultando una Colección – Criterios de Selección

Operadores $

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

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

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

```
>
> db.facturas.find( { nroFactura : {$gt:1465} } ).count()
30512
>
```

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")} } )
```

```
> db.facturas.find({fechaEmision:{$gte: ISODate("2014-02-24T00:00:00Z")} } )
{ "_id" : ObjectId("53685dbb2baf7b93f61df567"), "nroFactura" : 1450, "fechaEmisi
on" : ISODate("2014-02-24T00:00:00Z"), "fechaVencimiento" : ISODate("2014-02-24T
00:00:00Z"), "condPago" : "CONTADO", "cliente" : { "nombre" : "Juan Manuel", "ap
ellido" : "Manoni", "cuit" : 2029889382, "region" : "NEA" }, "item" : [
{       "producto" : "TUERCA 2mm",      "cantidad" : 2,          "precio" : 60 },
        {       "producto" : "TALADRO 12mm",    "cantidad" : 1,          "precio"
 : 490 },       {       "producto" : "TUERCA 5mm",      "cantidad" : 15,
"precio" : 90 } ] }
{ "_id" : ObjectId("53685dbb2baf7b93f61df568"), "nroFactura" : 1451, "fechaEmisi
on" : ISODate("2014-02-24T00:00:00Z"), "fechaVencimiento" : ISODate("2014-03-26T
00:00:00Z"), "condPago" : "30 Ds FF", "cliente" : { "nombre" : "Soledad", "apell
ido" : "Lavagno", "cuit" : 2729887543, "region" : "NOA" }, "item" : [  {  "produ
```

DBlandIT®
databases + science +IT

Consultando una Colección – Ordenamiento

Ordenamiento de documentos.

```
db.facturas.find({},{nroFactura:1,fechaEmision:1}).sort({fechaEmision:1})     -- Orden ascescendente
db.facturas.find({},{nroFactura:1,fechaEmision:1}).sort({fechaEmision:-1})    -- Orden descendente
```

```
> db.facturas.find({},{nroFactura:1,fechaEmision:1,_id:0}).sort({fechaEmision:-1
})
{ "nroFactura" : 1459, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1460, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1466, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1467, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1473, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1474, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1480, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1481, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1487, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1488, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1494, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
{ "nroFactura" : 1495, "fechaEmision" : ISODate("2014-02-25T00:00:00Z") }
```

DBlandIT®
databases + science + IT

Insertando un Documento

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" }
```

DBlandIT®
databases + science + IT

Insertando un Documento

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 }"
        }
})
```

DBlandIT®
databases + science +IT

Borrando Documentos

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()
```

DBlandIT®
databases + science +IT

Borrando Documentos

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 }
>
```

DBlandIT®
databases + science +IT

Modificando Documentos

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.

Modificando Documentos

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)*

Modificando Documentos Completos

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 }
```

DBlandIT®
databases + science +IT

Modificando Documentos Parciales

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 }
```

DBlandIT®
databases + science +IT

Modificando Documentos Parciales

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
" }
```

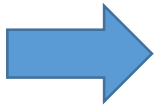
DBlandIT®
databases + science + IT

Modificando Documentos Completos

Update Totales/Completos

```
mydoc=db.facturas.findOne({nroFactura:1449})
```

```
> mydoc=db.facturas.findOne({nroFactura:1449})
{
        "_id" : ObjectId("53685dbb2baf7b93f61df566"),
        "nroFactura" : 1449,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-02-20T00:00:00Z"),
        "condPago" : "CONTADO",
        "cliente" : {
                "nombre" : "Martin",
                "apellido" : "Zavasi",
                "cuit" : 2038373771,
                "region" : "CABA"
        },
        "item" : [
                {
                        "producto" : "TUERCA 2mm",
                        "cantidad" : 6,
                        "precio" : 60
                },
                {
                        "producto" : "CORREA 10mm",
                        "cantidad" : 12,
                        "precio" : 134
                }
        ]
}
```



```
> db.facturas.findOne({nroFactura:1449})
{
        "_id" : ObjectId("53685dbb2baf7b93f61df566"),
        "nroFactura" : 1449,
        "fechaEmision" : ISODate("2014-02-20T00:00:00Z"),
        "fechaVencimiento" : ISODate("2014-02-20T00:00:00Z"),
        "condPago" : "XXX",
        "cliente" : {
                "nombre" : "Martin",
                "apellido" : "Zavasi",
                "cuit" : 2038373771,
                "region" : "CABA"
        },
        "item" : [
                {
                        "producto" : "TUERCA 2mm",
                        "cantidad" : 6,
                        "precio" : 60
                },
                {
                        "producto" : "CORREA 10mm",
                        "cantidad" : 12,
                        "precio" : 134
                }
        ]
}
```

```
mydoc.condPago="XXX"
db.facturas.update(    { _id: mydoc._id },   mydoc )
```

Sólo actualizamos el documento cuyo _id fue el recuperado con el findOne()

DBlandIT®
databases + science + IT

Un primer acercamiento

Caso Práctico en MongoDB

Levanto una instancia mongo

mongod --dbpath c:\data\db - por default se ejecuta en port 27017

Creo la BD Finanzas y cargo datos los datos de archive

mongoimport -d finanzas -c facturas facturas.json

Levanto el shell de mongo

mongo -- Por default levanta el motor una BD **test**

Consulto las BD existentes

>show dbs

Accedo a BD finanzas

>cd finanzas

Consulto las colecciones existentes

>show collections

DBlandIT®
databases + science +IT

Un primer acercamiento

Insertar registros en una nueva base

```
use finanzas2                          --si no existe la BD finanzas2, la crea en el primer insert.

db.facturas2.insert                    --si no existe la colección facturas2, la crea.
({nroFactura:1448, fechaEmision:ISODate('2014-02-20 00:00:00Z' ),
fechaVencimiento:ISODate('2014-03-22 00:00:00Z' ),
condPago:'30 Ds FF',
cliente:{nombre:'Martín',apellido:'Zavasi',cuit:2038373771,region:'CABA'},
item:[{producto:'CORREA 10mm', cantidad:2, precio:134}  ]  }  )
```

Ejercicios

Consultas

1. Consultar la cantidad de documentos insertados.
2. Obtener 1 sólo documento para ver el esquema y los nombres de los campos. Sin mostrar el _id.
3. Listar todos los datos de la factura 1149.
4. Obtener sólo los datos de cliente de las facturas donde se haya comprado “CORREA 10mm”. Ordenar por apellido del cliente.
5. Obtener sólo el nombre del producto de las facturas donde se haya comprado 15 unidades.

Altas, bajas y modificaciones

6. Insertar una factura número 999 con usted como cliente, habiendo comprado un Destornillador.
7. Eliminar todas las facturas de los clientes de la región CENTRO..
8. A la factura número 1500 cambiarle la condición de pago a “30 Ds FF”
9. A cada factura del cliente Lavagno agregarle el campo “tipo” con el valor “VIP”. Este deberá estar dentro del campo cliente. (cliente:{nombre:…, apellido:…, tipo:…, …}).

Respuestas

Consultas

1. Consultar la cantidad de documentos insertados.
   ```
   db.facturas.count()
   ```

2. Obtener 1 sólo documento para ver el esquema y los nombres de los campos. Sin mostrar el _id.
   ```
   db.facturas.findOne({},{_id:0})
   db.facturas.find ({},{_id:0}).limit(1)
   ```

3. Listar todos los datos de la factura 1149.
   ```
   db.facturas.find({nroFactura:1149})
   ```

4. Obtener sólo los datos de cliente de las facturas donde se haya comprado “CORREA 10mm”. Ordenar por apellido del cliente.
   ```
   db.facturas.find({"item.producto":"CORREA 10mm"}).sort({"cliente.apellido":1})
   ```

5. Obtener sólo el nombre del producto de las facturas donde se haya comprado 15 unidades.
   ```
   db.facturas.find({"item.cantidad":15},{"item.producto":1,_id:0})
   ```

Respuestas

Altas, bajas y modificaciones

1. Insertar una factura número 999 con usted como cliente, región CENTRO, habiendo comprado un Destornillador.

```
db.facturas.insert({nroFactura:999,cliente:{apellido:"Mi apellido",nombre:"Mi Nombre"},item:[{producto:"Destornillador"}]})
```

2. Eliminar todas las facturas de los clientes de la región CENTRO.

```
db.facturas.remove({"cliente.region":"CENTRO"})
```

3. A la factura número 999 cambiarle la condición de pago a "30 Ds FF"

```
db.facturas.update({nroFactura:999},{$set:{condPago:"30 Ds FF"}} ,{multi:true})
```

4. A cada factura del cliente Lavagno agregarle el campo "tipo" con el valor "VIP". Este deberá estar dentro del campo cliente. (cliente:{nombre:…, apellido:…, tipo:…, …}).

```
db.facturas.update({"cliente.apellido":"Lavagno"},{$set:{"cliente.tipo":"VIP"}},{multi:true})
```

DBlandIT®
databases + science + IT

Preguntas ???

DBlandIT – BIG DATA for YOUR BUSINESS

SOBRE DBlandIT

¿Quiénes Somos?

- Profesionales con amplia experiencia en el campo de la captura, almacenamiento, procesamiento, explotación y análisis de datos.
- DBlandIT es una empresa argentina con foco exclusivo en tecnologías sobre BIG DATA.

¿Qué hacemos?

- Proporcionamos servicios on y off-shore referidos a BIG DATA abarcando desde el mejor uso de las herramientas tecnológicas que lo soportan - arquitectura, implementación, soporte post implementación, mejores prácticas, educación - hasta la consultoría de negocio - evaluación de proyectos, factibilidad, planes estratégicos de implementación -.

¿Por qué lo hacemos?

- Queremos ser una empresa líder en la región en lo relacionado a las tecnologías emergentes de captura, almacenamiento, procesamiento, explotación y análisis de datos asociadas al concepto de BIG DATA.
- Para esto, queremos proveer servicios de valor agregado en las tecnologías emergentes aplicadas al Negocio, procesando, analizando y explotando a grandes volúmenes de datos.

Partnership

- DBlandIT es Ready Service Partner de mongoDB para Argentina y la Región. *(https://www.mongodb.com/partners)*

DBlandIT®
databases + science +IT

DBlandIT – BIG DATA for YOUR BUSINESS

Muchas Gracias !!!

Ing. Juan Antonio Zaffaroni
Founder & CEO
jzaffaroni@dblandit.com
www.dblandit.com

DBlandIT®
databases + science +IT