

# Curso Intro BigData - MongoDB

Bases de Datos Document Based - 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.

# Document Based

- MongoDB
- CouchBase
- CouchDB
- RethinkDB
- Terrastore
- OrientDB
- RavenDB
- Lotus Notes

# MongoDB



- Su nombre surge de la palabra en inglés "hu**mongous**" (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.

# El Origen



2007

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

2009

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

2011

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

# El Origen



2015

**En Marzo se lanza la versión 3.0 que da la posibilidad de cambiar el motor de almacenamiento por el nuevo WiredTiger.**

2018

**En junio se lanzó la versión la versión 4.0 que tiene Transacciones en un Cluster con Replica Set.**

2019

**En agosto la versión 4.2 con transacciones en un Cluster con la Base de Datos distribuida.**

2021

**MongoDB Last Version 5.0**

# MongoDB – NoSQL Líder en el Mercado

381 systems in ranking, November 2021

Rank			DBMS	Database Model	Score		
Nov 2021	Oct 2021	Nov 2020			Nov 2021	Oct 2021	Nov 2020
1.	1.	1.	Oracle +	Relational, Multi-model ⓘ	1272.73	+2.38	-72.27
2.	2.	2.	MySQL +	Relational, Multi-model ⓘ	1211.52	-8.25	-30.12
3.	3.	3.	Microsoft SQL Server +	Relational, Multi-model ⓘ	954.29	-16.32	-83.35
4.	4.	4.	PostgreSQL + ⓘ	Relational, Multi-model ⓘ	597.27	+10.30	+42.22
5.	5.	5.	MongoDB +	Document, Multi-model ⓘ	487.35	-6.21	+33.52
6.	6.	↑ 7.	Redis +	Key-value, Multi-model ⓘ	171.50	+0.15	+16.08
7.	7.	↓ 6.	IBM Db2	Relational, Multi-model ⓘ	167.52	+1.56	+5.90
8.	8.	8.	Elasticsearch	Search engine, Multi-model ⓘ	159.09	+0.84	+7.54
9.	9.	9.	SQLite +	Relational	129.80	+0.43	+6.48
10.	10.	10.	Cassandra +	Wide column	120.88	+1.61	+2.13
11.	11.	11.	Microsoft Access	Relational	119.24	+2.86	+2.01
12.	12.	12.	MariaDB +	Relational, Multi-model ⓘ	102.19	-0.41	+9.90
13.	13.	13.	Splunk	Search engine	92.31	+1.69	+2.60
14.	14.	↑ 15.	Hive +	Relational	83.31	-1.43	+13.05
15.	15.	↑ 17.	Microsoft Azure SQL Database	Relational, Multi-model ⓘ	81.32	+1.60	+14.33
16.	16.	16.	Amazon DynamoDB +	Multi-model ⓘ	76.99	+0.43	+8.09
17.	17.	↓ 14.	Teradata +	Relational, Multi-model ⓘ	69.59	-0.24	-6.01
18.	18.	↑ 42.	Snowflake +	Relational	64.19	+5.93	+54.09
19.	19.	↑ 20.	Neo4j +	Graph	57.98	+0.11	+4.45
20.	20.	↓ 19.	SAP HANA +	Relational, Multi-model ⓘ	55.53	+0.26	+1.95
21.	21.	↑ 23.	FileMaker	Relational	54.22	+1.38	+7.56
22.	22.	↓ 21.	Solr	Search engine, Multi-model ⓘ	53.85	+2.69	+2.04

# MongoDB – NoSQL Líder en el Mercado

THE FORRESTER WAVE™  
Big Data NoSQL  
Q1 2019





# MongoDB – NoSQL Líder en el Mercado



aws

Productos Soluciones Precios Documentación Aprender Red de socios AWS Marketplace Habilitación para clientes Eventos Explorar más

Amazon DocumentDB Información general Características Precios Introducción Recursos Preguntas frecuentes Clientes Migraciones

« Base de datos

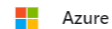
## Amazon DocumentDB

(compatible con MongoDB)

Escale las cargas de trabajo JSON con facilidad mediante un servicio de base de datos de documentos completamente administrado

Introducción a Amazon DocumentDB Contacto de ventas

- Admita millones de solicitudes de lectura de documentos por segundo al escalar los recursos informáticos y de almacenamiento de forma independiente.
- Automatice el aprovisionamiento de hardware, la aplicación de parches, la configuración y otras tareas de administración de bases de datos.
- Logre una durabilidad del 99,999999999 % con la replicación automática, la copia de seguridad continua y el aislamiento estricto de la red.
- Utilice los controladores y las herramientas existentes de MongoDB con las API de código abierto MongoDB 3.6 y 4.0 de Apache 2.0.



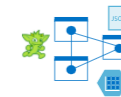
Empiece gratis

## ¿Qué se puede hacer con la cuenta gratuita?

Aquí hay algunas ideas de todo lo que puede hacer con Azure



Conéctese rápidamente a miles de millones de puntos de datos de todo el mundo.



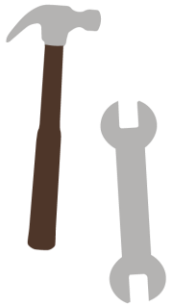
Desarrolle aplicaciones rápidamente con las API que prefiera para MongoDB, Cassandra, Gremlin, etc.



Cree aplicaciones sin servidor con Azure Functions y Azure Cosmos DB.

# MongoDB

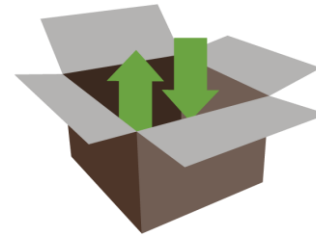
Es la Base de Datos NoSQL líder en:



Bases de Datos de  
Propósitos Generales



Bases de Datos  
Documentales

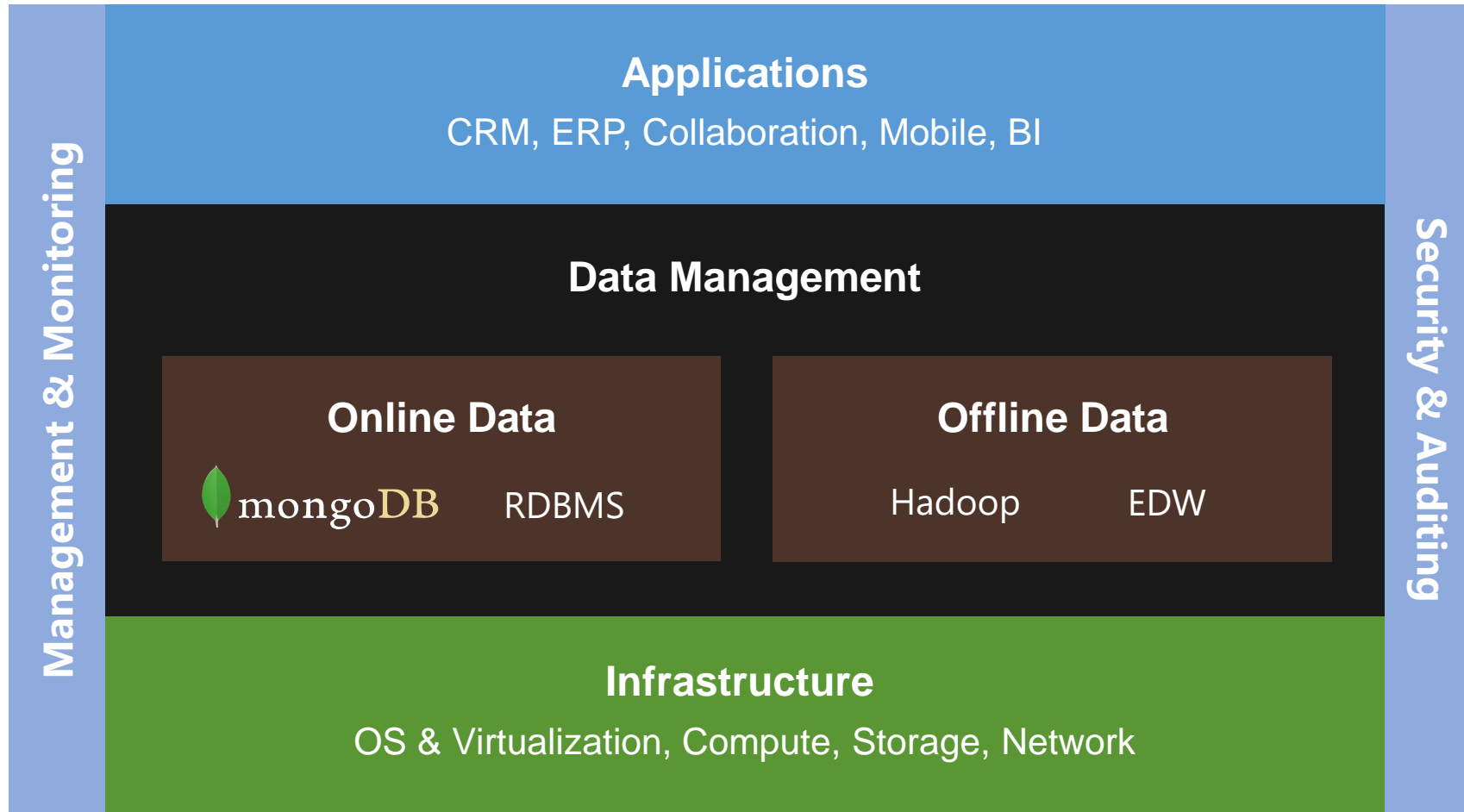


Bases de Datos  
De Código Abierto

# 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 Hadoop.
- 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.

# MongoDB & Enterprise IT Stack



# Terminología RDBMS vs. Document Based (MongoDB)

RDBMS	MongoDB
Database instance	MongoDB instance
Database / Schema	Database
Table	Collection
Row	Document
Rowid	_id
Join	Dbref \$lookup, \$graphlookup <i>Denormalización (\$unwind)</i>

# Estructura lógica

**JSON**    {"nro":123,  
              "descr":"prueba"}

## BSON

4 bytes	1 byte	3 bytes	1 byte	4 bytes	1 byte	5 bytes	1 byte	4 bytes	6 bytes	1 byte	1 byte
Length	DType	nro	\0	123	Dtype	descr	\0	LenTxt	prueba	\0	0
Tamaño	Atributo nro					Atributo descr					Fin doc

Total 32 bytes

### Tamaño Máximo

Los documentos no pueden ser mayores a **16Mb**, pero existen objetos denominados **GridFS** que permiten almacenar info de gran tamaño (Gb/Tb).

# 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.

# 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.



# 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,
          USASState: "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,
          USASState: "CA" } }
```



### Colección Editores

```
{ nombre: "O'Reilly Media",
  anioFundacion: 1980,
  USASState: "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.

Copyright (C) DBlandIT SRL. Todos los derechos reservados.

# 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,
          USASState: "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,
          USASState: "CA" } }
```



### Colección Editores

```
{ _id: "oreilly"
  nombre: "O'Reilly Media",
  anioFundacion: 1980,
  USASState: "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.**

# Tipos de datos

Data Type	Description
String	Todos los strings tienen que ser UTF-8. Si el string no es UTF-8 MongoDB lo tomará como Dato Binario.
Arrays	[valor, valor, valor]
Documentos embebidos	{key:value, key2:value2}
Dates	Se puede utilizar el constructor ISODATE para asignar valores de la forma ISODate("2012-12-19T06:01:17.171Z")
ObjectId	ObjectId es un tipo de datos BSON de 12 bytes. Para crear un Nuevo objectId x = ObjectId()
NumberLong	Maneja enteros de hasta 64 bits NumberLong("2090845886852")
NumberInt	Maneja enteros de hasta 32 bits NumberInt("2090845882")
Boolean	true or false
Numbers	MongoDB trata a los valores numéricos como valores Float

# ¿En qué casos usarlas?

MongoDB tiene un lenguaje de consulta de datos basado en JSON.

- Contiene construcciones como \$match para la cláusula "where", \$sort para ordenar los datos, o \$explain para mostrar el plan de ejecución.
- Contiene también la posibilidad de expresar consultas mediante expresiones regulares:

```
db.orders.find({"items.product.name": /Refactoring/})
```

# ¿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.

# ¿En qué casos usarlas?

## **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

- MongoDB soporta transacciones a partir de la 4.0 y transacciones distribuidas a partir de la 4.2

## Consultas contra estructuras de agregados variables.

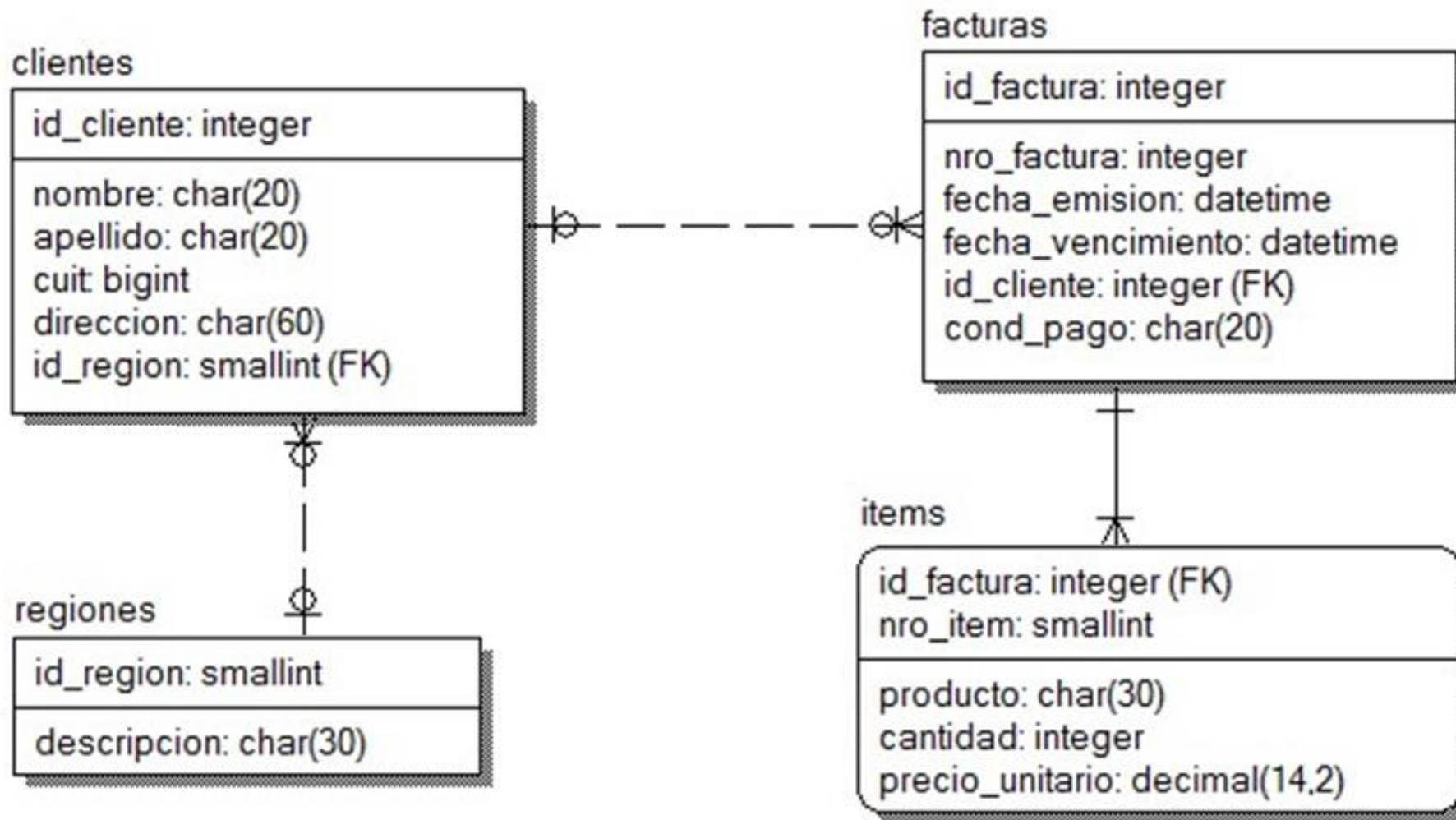
- que los datos se almacenen con cualquier estructura no implica que sea óptimo consultar por cualquier clave. Si los agregados varían entre sí, las consultas debieran variar también. Puede llevar a normalizar los datos, que no es lo que queremos.

Fin

**Pasemos a la parte práctica,  
Comenzamos con MongoDB.**



# Caso Práctico



# Caso Práctico

## Cientes

<u>id_cliente</u>	nombre	apellido	cuit	direccion	<u>id_region</u>
1	Juan Manuel	Manoni	2029889382	B. de Irigoyen 384	2
2	Martín	Zavasi	2038373771	Belgrano 4522	3
3	Marina	Malinez	2740488484	Alberdi 898	4
4	Soledad	Lavagno	2729887543	R. Saenz Peña 583	1

## Regiones

<u>id_region</u>	descripcion
1	NOA
2	NEA
3	CABA
4	CENTRO

Primary Keys

Foreign Keys

## Facturas

<u>id_factura</u>	nro_factura	fecha_emision	fecha_vencimiento	<u>id_cliente</u>	cond_pago
1	1447	20/02/2014	20/02/2014	3	CONTADO
2	1448	20/02/2014	22/03/2014	2	30 Ds FF
3	1449	20/02/2014	20/02/2014	2	CONTADO
4	1450	24/02/2014	24/02/2014	1	CONTADO
5	1451	24/02/2014	26/03/2014	4	30 Ds FF
6	1452	25/02/2014	26/04/2014	1	60 Ds FF
7	1453	25/02/2014	25/02/2014	1	CONTADO

## Items

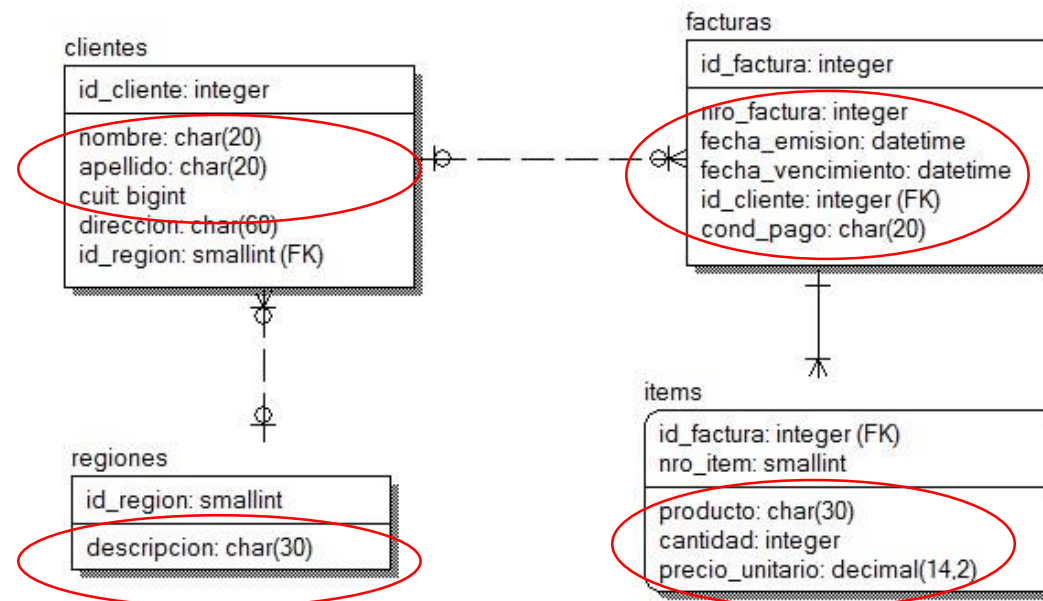
<u>id_factura</u>	<u>nro_item</u>	producto	cantidad	precio_unitario
1	1	CORREA 12mm	11	18,00
1	2	TALADRO 12mm	1	490,00
2	1	CORREA 10mm	2	134,00
3	1	TUERCA 2"	6	60,00
3	2	CORREA 10mm	12	134,00
4	1	TUERCA 2"	2	60,00
4	2	TALADRO 12mm	1	490,00
4	3	TUERCA 5"	15	90
5	1	SET HERRAMIENT	1	700,00
6	1	SET HERRAMIENT	1	700,00
6	2	TALADRO 12mm	1	490,00
7	1	TUERCA 5"	10	90

# Caso Práctico

**Se requiere armar 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.**

# Caso Práctico

**Se requiere armar 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.**



# Caso Práctico

```
{  
  nroFactura: 9999999,  
  fechaEmision: ISODate("yyyy-mm-ddThh:mm:ssZ"),  
  fechaVencimiento: ISODate("yyyy-mm-ddThh:mm:ssZ"),  
  condPago: "XXXXXXXXX"  
}
```

# Caso Práctico

```
{  
  nroFactura: 9999999,  
  fechaEmision: ISODate("yyyy-mm-ddThh:mm:ssZ"),  
  fechaVencimiento: ISODate("yyyy-mm-ddThh:mm:ssZ"),  
  condPago: "XXXXXXXXX",  
  "cliente":{ nombre: "XXXXXXX",  
               apellido: "XXXXXXXXX",  
               cuit:999999999999999,  
               region: "CABA"  
            }  
}
```

# Caso Práctico

```
{  
  nroFactura: 9999999,  
  fechaEmision: ISODate("yyyy-mm-ddThh:mm:ssZ"),  
  fechaVencimiento: ISODate("yyyy-mm-ddThh:mm:ssZ"),  
  condPago: "XXXXXXXX",  
  "cliente":{ nombre: "XXXXXXX",  
               apellido: "XXXXXXXXX",  
               cuit:99999999999999,  
               region: "CABA"  
            },  
  "items":[{producto:"XXXXXXXXX", cantidad: 999, precio:99.99} ,  
            {producto:"XXXXXXXXX", cantidad: 999, precio:99.99}  
          ]  
}
```

# Un primer acercamiento

## Caso Práctico en MongoDB

### Ejemplo de inserción y consulta de documentos

Levanto una instancia mongo

**mongod --dbpath c:\data\** --Por default se ejecuta en port 27017

Levanto el shell de mongo

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

Insertamos datos en la Base

**db.prueba.insert({x:1})** --Se crea la colección **prueba** si no existe.

**db.prueba.insert({x:1,y:2})**

**db.prueba.insert({x:2,y:3})**

**db.prueba.insert({x:1,y:4,z:33})**

Consultamos los datos

**db.prueba.find()**

**db.prueba.find().sort({x:1,y:1})** --Ordena el resultado por el atributo **x** y el atributo **y** ascendentes.



# Un primer acercamiento

## Insertar registros en una nueva base

**use finanzas**

*--si no existe la BD **finanzas**, la crea en el primer insert.*

**db.facturas.insert**

*--si no existe la colección **facturas**, 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} ] } )
```

**db.facturas.insert**

```
{nroFactura:1449, fechaEmision:ISODate('2014-02-20 00:00:00Z' ),  
fechaVencimiento:ISODate('2014-02-20 00:00:00Z' ), condPago:'CONTADO',  
cliente:{nombre:'Martín',apellido:'Zavasi',cuit:2038373771,region:'CABA'},  
item:[ {producto:'TUERCA 2mm', cantidad:6, precio:60},  
{producto:'CORREA 10mm', cantidad:12, precio:134} ] } )
```

# Un primer acercamiento

## Consultas de los documentos insertados

```
db.facturas.find()
```

```
db.facturas.find({condPago:'CONTADO'},{_id:0,"cliente.apellido":1}).pretty()
```

```
<
  "_id" : ObjectId<"534b52bc43ad02d44397b71c">,
  "nroFactura" : 1449,
  "fechaEmision" : ISODate<"2014-02-20T00:00:00Z">,
  "fechaVencimiento" : ISODate<"2014-02-20T00:00:00Z">,
  "condPago" : "CONTADO",
  "cliente" : {
    "nombre" : "Martín",
    "apellido" : "Zavasi",
    "cuit" : 2038373771,
    "region" : "CABA"
  },
  "item" : [
    {
      "producto" : "TUERCA 2mm",
      "cantidad" : 6,
      "precio" : 60
    },
    {
      "producto" : "CORREA 10mm",
      "cantidad" : 12,
      "precio" : 134
    }
  ]
}>
```