



Escuela Profesional de  
Ciencia de la Computación

BDI Fase 3

# Base de Datos I

Dr. Edward Hinojosa C.

Dr. Edgar Sarmiento C.

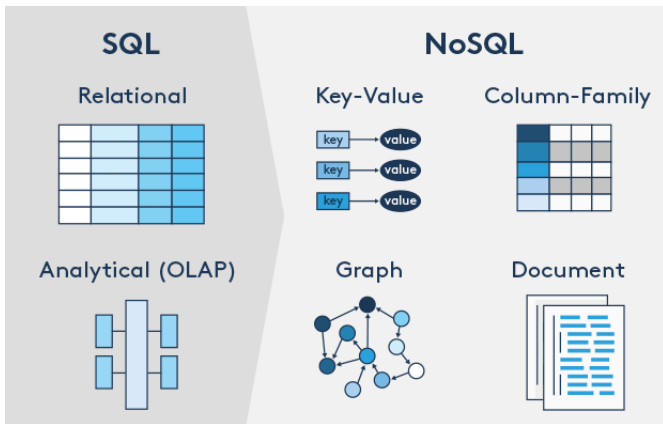
Universidad Nacional de San Agustín de Arequipa

2020/Semestre Par

# Índice

- 1 Bases de Datos NoSQL
- 2 BD NoSQL Orientados a Documentos
- 3 MongoDB
- 4 BDR vs BDOD

# Tipos de Bases de Datos NoSQL



# BD NoSQL Orientados a Documentos

- Las Bases de Datos Orientadas a Documentos (BDOD) utilizan el concepto de datos y documentos **autocontenidos y autodescriptivos**.
- Esto implica que el documento en sí define cómo debe presentarse y cuál es el significado de los datos almacenados en su estructura.

# BD NoSQL Orientados a Documentos

- Una BDOD tiene la característica de contener:
  - Toda la información importante en un solo documento.
  - Estar libre de esquemas.
  - Contar con identificadores universales únicos (UUID), posibilitando la consulta de documentos a través de métodos avanzados de agrupamiento y filtrado (MapReduce).
  - Permitir redundancia e inconsistencia.

# BD NoSQL Orientados a Documentos

## Document Database

---



# MongoDB

- MongoDB es una base de datos orientada a documentos que comenzó en 2007 pero se completó en el 2009.



# MongoDB

- MongoDB tiene la característica de ser open source licenciada por la GNU AGPL (Affero General Public License) hasta versión 3.0.
- Actualmente MongoDB continua siendo de código abierto, aunque, para las ediciones empresariales, debemos pagar la licencia.
- Se considera de alto rendimiento.
- No contiene esquemas.
- Escrita en C++.
- Es Multiplataforma. MongoDB tiene binarios para varias plataformas como Windows, Mac OS X, Linux y Solaris.
- Utiliza el lenguaje JSON.
- Más características en <http://mongodb.com/>



# MongoDB

- Entre las empresas que ya utilizan MongoDB se encuentran:
  - Disney
  - SAP
  - The New York Times
  - The Guardian
  - Viber
  - ...

# MongoDB

## Who uses MongoDB



# Instalando MongoDB

## Choose which type of deployment is best for you



### Cloud

The easiest way to run MongoDB



### On-Premises

Download on your own infrastructure



### Tools

Do more with your data(base)

## MongoDB Enterprise Server

## MongoDB Community Server

MongoDB offers both an Enterprise and Community version of its powerful distributed document database. The community version offers the flexible document model along with ad hoc queries, indexing, and real time aggregation to provide powerful ways to access and analyze your data. As a distributed system you get high availability through built-in replication and failover along with horizontal scalability with native sharding.

The MongoDB Enterprise Server gives you all of this and more. Review the Enterprise Server tab to learn what else is available.

## Available Downloads

Version

4.4.2 (current)

Platform

Windows

Package

msi



Download

Copy Link

## Current releases & packages

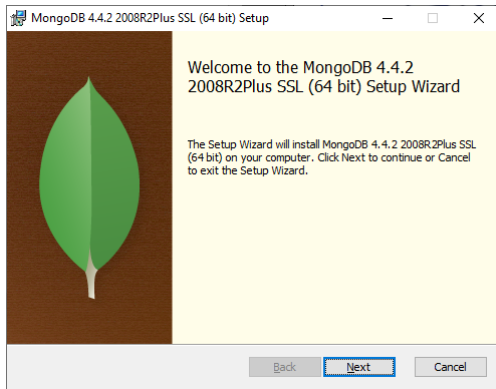
[Development releases](#)

[Archived releases](#)

[Changelog](#)

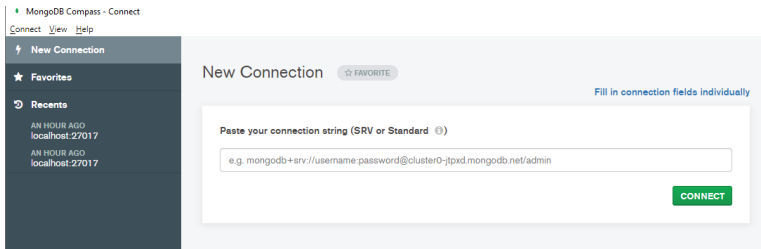
[Release Notes](#)

# Instalando MongoDB



# Instalando MongoDB

- Utilizaremos MongoDB Compass. La interfaz gráfica de usuario de MongoDB.



# Ejemplo - Blog

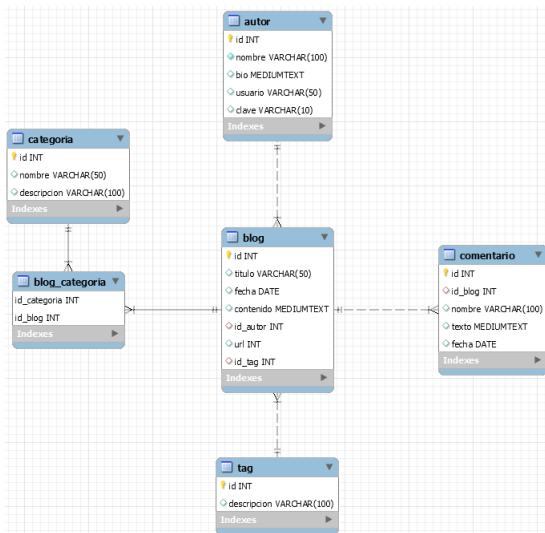
- Cada blog posee un enlace único que se utiliza para ser accedido.
- Cada blog posee un autor y fecha de creación.
- Cada blog posee un título y un contenido.
- Cada blog posee categorías y un tag.
- Un blog puede tener varios comentarios. De cada comentario se almacena el nombre de quien realizó el comentario, fecha y comentario.

`https://www.consumidor.ftc.gov/blog/2020/04/cheques-por-el-coronavirus-aplanemos-la-curva-de-las-esta`

# Blog - Modelo Relacional

- Blog con un FK con Autor.
- Autor
- Categoría
- BlogCategoría: Tabla Intermedia, FK con Categoría y Blog
- Tag: FK con Artículo
- Comentarios: FK con Blog

# Blog - Modelo Relacional





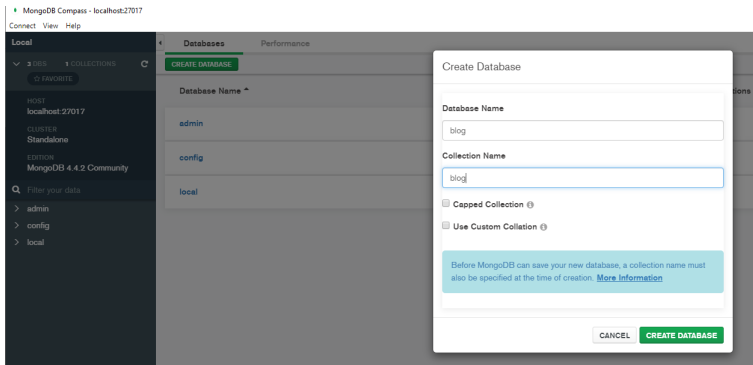
# Blog - Modelo Relacional

- Para poder mostrar el blog debo realizar la siguiente consulta (se debe utilizar varios JOINS):

```
SELECT blog.*, autor.*
....
INNER JOIN autor ....
WHERE URL='XX'
SELECT categoria.*
....
INNER JOIN blog_categoria ....
SELECT categoria.*
.....
SELECT comentario.*
.....
```

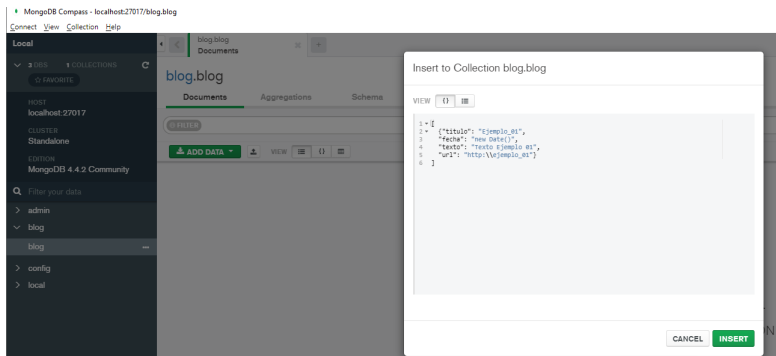
# Blog - Modelo no Relacional

- Creamos la base de datos y la primera colección de documentos:



# Blog - Modelo no Relacional

- Insertamos un nuevo documento (en este caso un blog):



# Blog - Modelo no Relacional

MongoDB Compass - localhost:27017/blog.blog

Connect View Collection Help

Local

3 DBS 1 COLLECTIONS

☆ FAVORITE

HOST  
localhost:27017

CLUSTER  
Standalone

EDITION  
MongoDB 4.4.2 Community

Filter your data

- admin
- blog
- blog
- config
- local

blog.blog Documents

blog.blog

Documents Aggregations Schema Expl

FILTER

ADD DATA

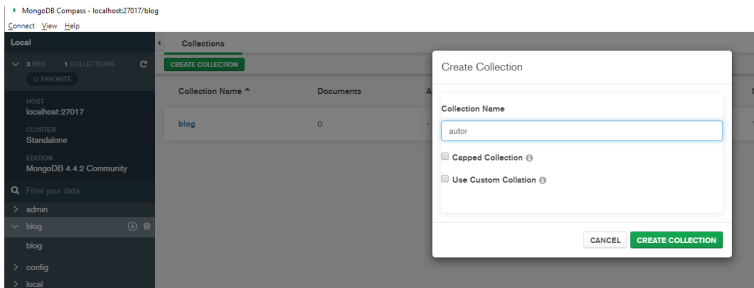
VIEW

```

_id: ObjectId("5fdbb86551a3d02c9cb975f7")
titulo: "Ejemplo_01"
fecha: "new Date()"
texto: "Texto Ejemplo 01"
url: "http:\\ejemplo_01"
  
```

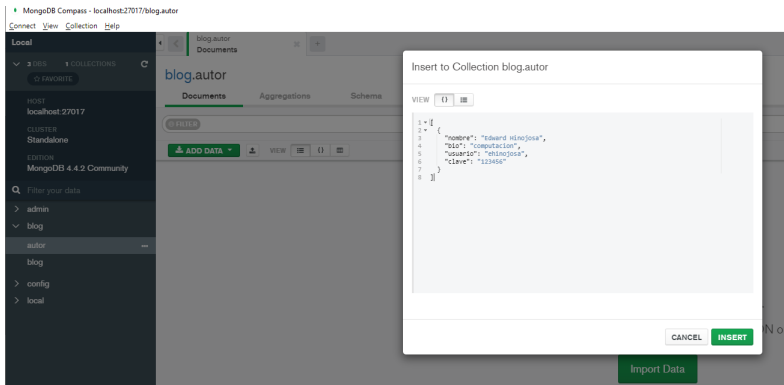
# Blog - Modelo no Relacional

- Creamos la colección autor:

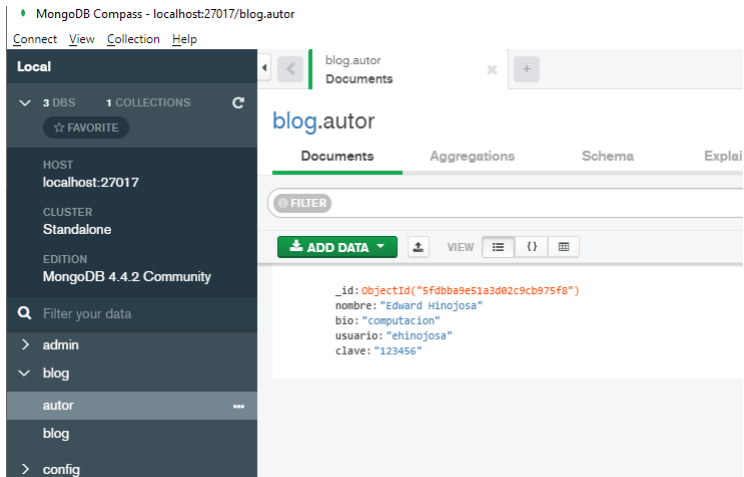


# Blog - Modelo no Relacional

- Insertamos un nuevo autor:



# Blog - Modelo no Relacional



# Blog - Modelo no Relacional

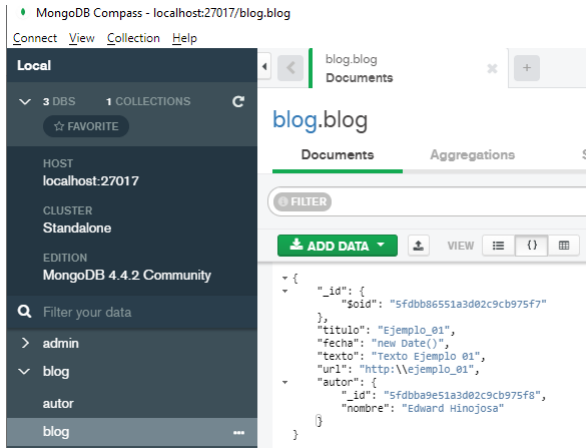
- Si quisieramos asignar el anterior autor al blog anterior pensaríamos en modificar el documento de blog adicionando solo el código generado anteriormente.
- En este caso no solo debemos adicionar le código, sino también lo que deseamos que aparezca en el blog (solo el nombre, no se necesita el usuario y contraseña). Recordemos, cada documento debe ser autocontenido y autodescriptivo.



# Blog - Modelo no Relacional

- Con ello en una consulta para mostrar el blog, no necesitamos realizar un join. Debemos pensar que la consulta sea lo más simple posible.
- Desventaja: En caso el nombre cambie, debemos cambiarlo en todos los documentos.

# Blog - Modelo no Relacional



# Blog - Modelo no Relacional

- En el caso del tag del blog lo podemos adicionar como un nuevo campo.

```
{
  "_id": {
    "$oid": "5fdbb86551a3d02c9cb975f7"
  },
  "titulo": "Ejemplo_01",
  "fecha": "new Date()",
  "texto": "Texto Ejemplo 01",
  "url": "http:\\ejemplo_01",
  "autor": {
    "_id": "5fdbba9e51a3d02c9cb975f8",
    "nombre": "Edward Hinojosa"
  },
  "tag": "tag 01"
}
```

# Blog - Modelo no Relacional

- En el caso de categorías podemos adicionar un array de campos.
- Con ello evitamos la tabla intermedia.

```
{
  "_id": {
    "$oid": "5fdbb86551a3d02c9cb975f7"
  },
  "titulo": "Ejemplo_01",
  "fecha": "new Date()",
  "texto": "Texto Ejemplo 01",
  "url": "http:\\ejemplo_01",
  "autor": {
    "_id": "5fdbba9e51a3d02c9cb975f8",
    "nombre": "Edward Hinojosa"
  },
  "tag": "tag 01",
  "categorias": [
    {"nombre": "categoria 01"},
    {"nombre": "categoria 02"},
    {"nombre": "categoria 03"}
  ]
}
```

# Blog - Modelo no Relacional

- Por último, los comentarios se pueden adicionar como las categorías, considerando más campos para cada array.

```
{
  "_id": {
    "$oid": "5fdbb86551a3d02c9cb975f7"
  },
  "titulo": "Ejemplo_01",
  "fecha": "new Date()",
  "texto": "Texto Ejemplo 01",
  "url": "http:\\ejemplo_01",
  "autor": {
    "_id": "5fdbba9e51a3d02c9cb975f8",
    "nombre": "Edward Hinojosa"
  },
  "tag": "tag 01",
  "categorias": [
    {"nombre": "categoria 01"},
    {"nombre": "categoria 02"},
    {"nombre": "categoria 03"}
  ],
  "comentarios": [
    {
      "nombre": "Edgar Sarmiento",
      "texto": "Buen blog",
      "fecha": "2018-11-06T:15:24:14.047Z"
    },
    {
      "nombre": "Juan Carlos Gutierrez",
      "texto": "Interesante blog",
      "fecha": "2018-11-10T:18:50:17.024Z"
    }
  ]
}
```

# Blog - Modelo no Relacional

- Con ello podemos mostrar todo el blog con una consulta simple. Ese es el objetivo principal.
- Se pueden realizar operaciones de inserción, actualización y eliminación (no las veremos) mediante comandos.
- Las operaciones mencionadas son abstraídas de los programadores.

# ¡GRACIAS!

