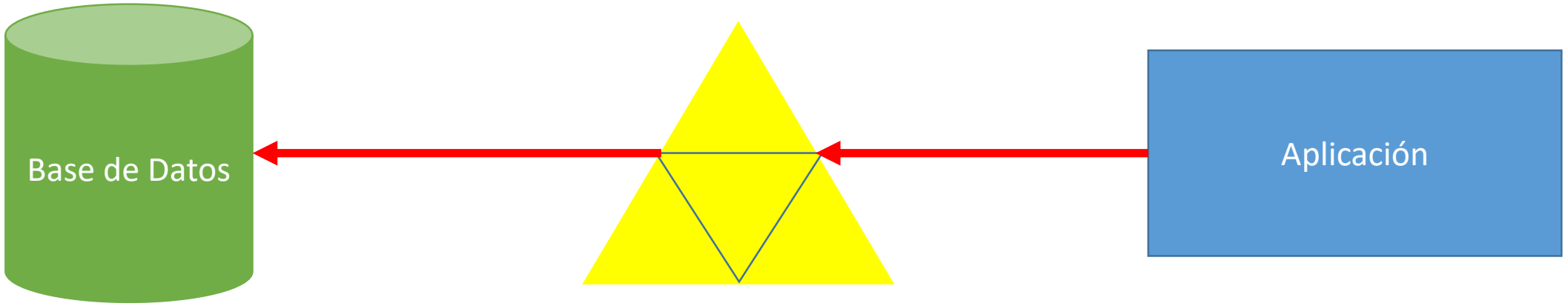
Mongo DB

Objetivo: Stack MEAN



Mongo DB

Servicios REST
NodeJS, Express

Clientes REST
Angular

Bases de Datos Relacionales Vs No Relacionales

CRITERIO

- Desde una perspectiva operativa, no comercial.
- Generalmente, la perspectiva comercial engrandece las ventajas y minimiza las desventajas de su producto
- Nos formamos un criterio cuando nos convertimos en usuarios de la tecnología, en proyectos reales.

VENTAS

Después de cerrar la venta de un proyecto… A cumplir todo lo que ventas prometió

OPERACIONES

Bases de Datos Relacionales Vs No Relacionales

CARACTERÍSTICAS

Bases de Datos Relacionales

- Uso de tablas, donde cada columna representa un campo y cada renglón representa un registro.
- Las tablas se relacionan entre sí por medio de llaves foráneas.
- Las tablas representan entidades, por ejemplo: Profesores, alumnos, edificios, salones de clases.
- Uso de normalización (3 formas normales).
- Propiedades ACID (Atomicity, Consistency, Isolation, Durability) a nivel de Transacciones.
- Consistencia de datos garantizada por medio de llaves foráneas y constraints. Si una llave foránea existe en una tabla, está garantizado que esa llave es primaria en otra tabla. No te permite la creación de un registro en una tabla “A” que tiene una llave foránea hacia la tabla “B” y que no existe en la tabla “B”.
- El diseño y estructura de las tablas es sencillo.

Bases de Datos Relacionales Vs No Relacionales

CARACTERÍSTICAS

Bases de Datos No Relacionales

- Existen 3 categorías principales dentro de las bases de datos no relacionales:
  - Documet Stores: Utilizan los conceptos de documentos y colecciones. Los documentos equivalen a los registros en una RDBMS, y las colecciones equivalen a las tablas. Es una extensión de la categoría Key-value store. Ejemplos: MongoDB, CouchDB.
  - Key-value Stores: Los datos son almacenados con una llave asociada a cada valor, similar a un Hash-table. Dependiendo del manejador de base de datos, una llave podría almacenar varios valores. Es el modelo de datos más simple que se puede tener. Una diferencia muy importante en comparación a las otras categorías es la funcionalidad que se provee para acceder y manipular los datos. Ejemplos: Memcached, Redis.
  - Graph databases: Utiliza los conceptos de nodos y aristas utilizados en un grafo. Los nodos corresponden a entidades y las aristas representan las relaciones que existen entre los nodos. Ejemplos: Neo4j, AllegroGraph.
- Otras Categorías: Wide Column Store, Multimodel Databases, XML Databases,etc. (http://nosql-database.org/)

Bases de Datos Relacionales Vs No Relacionales

CARACTERÍSTICAS

Bases de Datos No Relacionales – Document Stores

- Schemaless: No cuenta con un esquema predefinido, los documentos pueden tener diferentes campos según se requiera.
- Propiedades ACID (Atomicity, Consistency, Isolation, Durability) a nivel de Documentos.
- Uso de colecciones como tablas y documentos como registros.
- Pre Join / Embeded Data: Gran parte de la información relacionada con una entidad puede estar contendia dentro del mismo documento, mediante el uso de arreglos y/o más documentos.
- No existe el concepto de constrains: No garantiza la integridad de la información de forma nativa. La integridad se tiene que cuidar desde el código de la aplicación.
- Decisiones sobre el diseño y estructura de las colecciones son complejas.

Bases de Datos Relacionales Vs No Relacionales

¿Cuándo utilizar NoSQL o RDBMS?

- Las características no presenten un riesgo en el éxito del proyecto.
- Se pueden cubrir todos los requisitos de información definida en el sistema.
- Cubre los atributos de calidad establecidos por los stakeholders.
- Poseen el conocimiento y experiencia suficiente para no atrasar el proyecto con una curva de aprendizaje.
- Existe un requisito explicito por parte del cliente para utilizar uno de los dos enfoques.
- Complejidad de diseño de estructura de documentos con MongoDB en aplicaciones con muchas tablas.
- Usen lo mejor de ambos mundos. Pueden utilizar las dos bases de datos en una misma aplicación, tomando en cuenta funcionalidad que requiera el uso de transacciones y funcionalidad que aproveche las ventajas de de “Pre Join / Embeded Data”.

Diseño de Base de Datos Relacionales

3 Formas Normales

1NF (First Normal Form): Eliminar grupos de datos repetidos, creando una nueva tabla para cada grupo de datos relacionados, que serán identificados por una llave primaria.

ID_ALUMNO	NOMBRE	SALON
1	Juan Pérez	A4308, A5201
2	Guillermo Hernández	A4308
3	Pablo Ramírez	A5201, A6102, A4208

1NF

ID_ALUMNO	NOMBRE
1	Juan Pérez
2	Guillermo Hernández
3	Pablo Ramírez

ID_ALUMNO	SALON
1	A4308
1	A5201
2	A4308
3	A5201
3	A6102
3	A4208

Diseño de Base de Datos Relacionales

3 Formas Normales

2NF (Second Normal Form): Si un conjunto de valores es el mismo para varios registros, crear una nueva tabla con dicho conjunto de valores y ligar las dos tablas por medio de una llave foránea.

1NF

ID_ALUMNO	NOMBRE
1	Juan Pérez
2	Guillermo Hernández
3	Pablo Ramírez

ID_ALUMNO	SALON
1	A4308
1	A5201
2	A4308
3	A5201
3	A6102
3	A4208

2NF

ID_ALUMNO	NOMBRE
1	Juan Pérez
2	Guillermo Hernández
3	Pablo Ramírez

ID_ALUMNO	ID_SALON
1	1
1	2
2	1
3	2
3	3
3	4

ID_SALON	NOMBRE
1	A4308
2	A5201
3	A6102
4	A4208

Diseño de Base de Datos Relacionales

3 Formas Normales

3NF (Third Normal Form): Los campos que no dependan de la llave primaria de una tabla deben ser removidos, y de ser necesario, deben colocarse en otra tabla.

2NF

ID_ALUMNO	NOMBRE	CLAVE_CAMPUS	NOMBRE_CAMPUS
1	Juan Pérez	CEM	Campus Estado de México
2	Guillermo Hernández	CEM	Campus Estado de México
3	Pablo Ramírez	CCM	Campus Ciudad de México

3NF

ID_ALUMNO	NOMBRE	CLAVE_CAMPUS
1	Juan Pérez	CEM
2	Guillermo Hernández	CEM
3	Pablo Ramírez	CCM

CLAVE_CAMPUS	NOMBRE_CAMPUS
CEM	Campus Estado de México
CCM	Campus Ciudad de México

Diseño de Base de Datos – Document Stores (MongoDB)

ALUMNOS

{“_id” : 1,
“nombre” : “Juan Pérez”
“id_salon” : [1,2]
“id_campus” : “CEM”}

SALONES

{“_id” : 1,
“nombre” : “A4308”}

CAMPUS

{“_id” : “CEM”,
“nombre” : “Campus Estado de México”}

- No es un buen diseño. Una “regla de dedo” en MongoDB es que si tu diseño se parece a lo que harías con una base de datos relacional, vas por mal camino. Esto es porque seguramente no estás aprovechando la ventaja que ofrece “Pre Join / Embeded Data”.

Diseño de Base de Datos – Document Stores (MongoDB)

Relaciones Uno a Uno

Ejemplo: Un alumno y su expediente: Un alumno tiene un expediente. ¿Diseñar como una colección para alumnos y una colección para expedientes, o el expediente está empotrado dentro de la colección de alumnos?

Consideraciones:

- Frecuencia de acceso a la información: Tal vez no es muy frecuente accesar al expediente del alumno.
- Tamaño de los documentos: Si la información del expediente es muy “pesada” (16 megas para MongoDB tamaño máximo de un documento) conviene dejarla en una colección separada.

Diseño de Base de Datos – Document Stores (MongoDB)

Relaciones Uno a Muchos

Ejemplo: Un campus tiene muchos alumnos (10,000 alumnos).

- Diseño 1: Empotrar la información de los alumnos en la colección “campus”.

Colección “campus” -> {nombre: “itesm-cem”, dirección: “carretera…”, alumnos: […,…,…]}
NO ES VIABLE por la cantidad de elementos que puede tener el arreglo “alumnos”.

- Diseño 2: Empotrar la información del campus en la colección “alumnos”.

Colección “alumnos” -> {nombre: “Juan Pérez”, campus: {nombre: “itesm-cem”, dirección: “carretera…”}}
NO ES VIABLE porque se duplica TODA la información de “campus” en muchos documentos (10,000) y abre muchas posibilidades a inconsistencias. Se complica mantener actualizada la información de elemento campus en todos los documentos.

- Diseño 3: Crear una colección para “alumnos” y otra para “campus” y ligarlas por medio del “_id” de “campus”.

Colección “campus” -> {“_id” : “itesm-cem”, dirección : “carretera…”}
Colección “alumnos” –> {nombre : “fulanito”, “campus” : “itesm-cem”}
ES VIABLE. Aunque rompe la “regla de dedo”, pero dada la cantidad de información que se puede tener en los casos anteriores, es la opción más viable.

Diseño de Base de Datos – Document Stores (MongoDB)

Relaciones Uno a Pocos

Ejemplo: Un alumno tiene pocos salones.

- Se recomienda empotrar la colección “pocos” en la colección “uno”:

  alumnos-> {“_id” : “A012345678”, “nombre” : “Juan Pérez”, salones: [{“_id” : “A5401”, dato1: “…”, …}, {…}, {…}]}

Consideraciones:

- Frecuencia de cambio de la información contenida en los documentos empotrados (en éste caso de cada salón). Si de antemano sabemos que la información propia de los salones va cambiar constantemente entonces éste no es el mejor enfoque y habría que crear una colección independiente para los salones.
- Tamaño de los documentos: Si la información del salón es muy “pesada” (16 megas para MongoDB tamaño máximo de un documento) conviene dejarla en una colección separada.

Diseño de Base de Datos – Document Stores (MongoDB)

Relaciones Muchos a Muchos

Ejemplo: Spotify. Un usuario tiene muchas canciones y una canción tiene muchos usuarios.

- Se recomienda establecer la relación por medio de los “_id” de los documentos:

  usuarios -> {“_id” : “usuario123”, “nombre” : “Juan Pérez”, canciones: [5,78,1098,23,152]}
  canciones -> {“_id” : 1098, “titulo” : “The Globalist”, “autor”: “Muse”, “album” : “Drones”}

Consideraciones:

- Se tiene una gran cantidad de elementos en ambos lados de la relación, lo que complicaría el diseño y su control en código si se empotran los documentos.

Diseño de Base de Datos – Document Stores (MongoDB)

Relaciones Pocos a Pocos

Ejemplo: Un alumno tiene pocos profesores y un profesor tiene pocos alumnos.

- Dos opciones:

1. Utilizar el enfoque “muchos a muchos”.
2. Empotrar la información dentro de los documentos.

Consideraciones:

- Si el diseño de los documentos es complejo y a su vez tienen otros documentos empotrados, se recomienda utilizar la opción 1.
- Si el diseño de los documentos es simple y se sabe de antemano que no se volverán complejos, se recomienda utilizar la opción 2 para aprovechar las ventajas de “Pre Join / Embeded Data”.

MongoDB: Consultas

Documentación de Referencia: https://docs.mongodb.com/manual/tutorial/query-documents/

The following example selects from the `inventory` collection all documents where the `status` equals `"D"`:

copy

```
db.inventory.find( { status: "D" } )
```

The following example retrieves all documents from the `inventory` collection where `status` equals either `"A"` or `"D"`:

copy

```
db.inventory.find( { status: { $in: [ "A", "D" ] } } )
```

MongoDB: Consultas

The following example retrieves all documents in the `inventory` collection where the `status` equals `"A"` and `qty` is less than (`$lt`) `30`:

copy

```
db.inventory.find( { status: "A", qty: { $lt: 30 } } )
```

The following example retrieves all documents in the collection where the `status` equals `"A"` or `qty` is less than (`$lt`) `30`:

copy

```
db.inventory.find( { $or: [ { status: "A" }, { qty: { $lt: 30 } } ] } )
```

MongoDB: Consultas

In the following example, the compound query document selects all documents in the collection where the `status` equals `"A"` and *either* `qty` is less than (`$lt`) `30` *or* `item` starts with the character `p`:

copy

```
db.inventory.find( {
     status: "A",
     $or: [ { qty: { $lt: 30 } }, { item: /^p/ } ]
} )
```

MongoDB: Consultas en Documentos Anidados

Se tiene la siguiente información en la base de datos:

copy

```
db.inventory.insertMany( [
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status: "A" }
]);
```

MongoDB: Consultas en Documentos Anidados

For example, the following query selects all documents where the field `size` equals the document `{ h: 14, w: 21, uom: "cm" }`:

```
db.inventory.find( { size: { h: 14, w: 21, uom: "cm" } } )
```

copy

The following example selects all documents where the field `uom` nested in the `size` field equals `"in"`:

```
db.inventory.find( { "size.uom": "in" } )
```

copy

MongoDB: Consultas en Documentos Anidados

The following query uses the less than operator (`$lt`) on the field `h` embedded in the `size` field:

```
db.inventory.find( { "size.h": { $lt: 15 } } )
```

The following query selects all documents where the nested field `h` is less than `15`, the nested field `uom` equals `"in"`, and the `status` field equals `"D"`:

```
db.inventory.find( { "size.h": { $lt: 15 }, "size.uom": "in", status: "D" } )
```

MongoDB: Consultas en Arreglos

Se tiene la siguiente información en la base de datos:

copy

```
db.inventory.insertMany([
   { item: "journal", qty: 25, tags: ["blank", "red"], dim_cm: [ 14, 21 ] },
   { item: "notebook", qty: 50, tags: ["red", "blank"], dim_cm: [ 14, 21 ] },
   { item: "paper", qty: 100, tags: ["red", "blank", "plain"], dim_cm: [ 14, 21 ] },
   { item: "planner", qty: 75, tags: ["blank", "red"], dim_cm: [ 22.85, 30 ] },
   { item: "postcard", qty: 45, tags: ["blue"], dim_cm: [ 10, 15.25 ] }
]);
```

MongoDB: Consultas en Arreglos

The following example queries for all documents where the field `tags` value is an array with exactly two elements, `"red"` and `"blank"`, in the specified order:

copy

```
db.inventory.find( { tags: ["red", "blank"] } )
```

If, instead, you wish to find an array that contains both the elements `"red"` and `"blank"`, without regard to order or other elements in the array, use the `$all` operator:

copy

```
db.inventory.find( { tags: { $all: ["red", "blank"] } } )
```

MongoDB: Consultas en Arreglos

The following example queries for all documents where `tags` is an array that contains the string `"red"` as one of its elements:

copy

```
db.inventory.find( { tags: "red" } )
```

For example, the following operation queries for all documents where the array `dim_cm` contains at least one element whose value is greater than `25`.

copy

```
db.inventory.find( { dim_cm: { $gt: 25 } } )
```

MongoDB: Consultas en Arreglos

The following example queries for documents where the `dim_cm` array contains elements that in some combination satisfy the query conditions; e.g., one element can satisfy the greater than `15` condition and another element can satisfy the less than `20` condition, or a single element can satisfy both:

copy

```
db.inventory.find( { dim_cm: { $gt: 15, $lt: 20 } } )
```

Use `$elemMatch` operator to specify multiple criteria on the elements of an array such that at least one array element satisfies all the specified criteria.

The following example queries for documents where the `dim_cm` array contains at least one element that is both greater than (`$gt`) `22` and less than (`$lt`) `30`:

copy

```
db.inventory.find( { dim_cm: { $elemMatch: { $gt: 22, $lt: 30 } } } )
```

MongoDB: Consultas en Arreglos

Se puede hacer referencia a un elemento en específico del arreglo por medio de su posición. En el siguiente ejemplo se hace referencia a la posición 1 del arreglo dim_cm utilizando la notación de punto:

The following example queries for all documents where the second element in the array `dim_cm` is greater than `25`:

copy

```
db.inventory.find( { "dim_cm.1": { $gt: 25 } } )
```

MongoDB: Consultas en Arreglos

Use the `$size` operator to query for arrays by number of elements. For example, the following selects documents where the array `tags` has 3 elements.

copy

```
db.inventory.find( { "tags": { $size: 3 } } )
```

MongoDB: Consultas en Arreglos de Documentos

Se tiene la siguiente información en la base de datos:

```
db.inventory.insertMany( [
  { item: "journal", instock: [ { warehouse: "A", qty: 5 }, { warehouse: "C", qty: 15 } ] },
  { item: "notebook", instock: [ { warehouse: "C", qty: 5 } ] },
  { item: "paper", instock: [ { warehouse: "A", qty: 60 }, { warehouse: "B", qty: 15 } ] },
  { item: "planner", instock: [ { warehouse: "A", qty: 40 }, { warehouse: "B", qty: 5 } ] },
  { item: "postcard", instock: [ { warehouse: "B", qty: 15 }, { warehouse: "C", qty: 35 } ] }
]);
```

MongoDB: Consultas en Arreglos de Documentos

The following example selects all documents where an element in the `instock` array matches the specified document:

```
db.inventory.find( { "instock": { warehouse: "A", qty: 5 } } )
```

copy

The following example selects all documents where the `instock` array has at least one embedded document that contains the field `qty` whose value is less than or equal to `20`:

```
db.inventory.find( { 'instock.qty': { $lte: 20 } } )
```

copy

MongoDB: Consultas en Arreglos de Documentos

Se puede hacer referencia a un elemento en específico del arreglo por medio de su posición. En el siguiente ejemplo se hace referencia a la posición 0 del arreglo instock utilizando la notación de punto:

The following example selects all documents where the **instock** array has as its first element a document that contains the field **qty** whose value is less than or equal to **20**:

copy

```
db.inventory.find( { 'instock.0.qty': { $lte: 20 } } )
```

MongoDB: Consultas en Arreglos de Documentos

Use `$elemMatch` operator to specify multiple criteria on an array of embedded documents such that at least one embedded document satisfies all the specified criteria.

The following example queries for documents where the **`instock`** array has at least one embedded document that contains both the field **`qty`** equal to `5` and the field **`warehouse`** equal to `A`:

copy

```
db.inventory.find( { "instock": { $elemMatch: { qty: 5, warehouse: "A" } } } )
```

The following example queries for documents where the **`instock`** array has at least one embedded document that contains the field **`qty`** that is greater than `10` and less than or equal to `20`:

copy

```
db.inventory.find( { "instock": { $elemMatch: { qty: { $gt: 10, $lte: 20 } } } } )
```

MongoDB: Consultas en Arreglos de Documentos

For example, the following query matches documents where any document nested in the `instock` array has the `qty` field greater than `10` and any document (but not necessarily the same embedded document) in the array has the `qty` field less than or equal to `20`:

copy

```
db.inventory.find( { "instock.qty": { $gt: 10,  $lte: 20 } } )
```

The following example queries for documents where the `instock` array has at least one embedded document that contains the field `qty` equal to `5` and at least one embedded document (but not necessarily the same embedded document) that contains the field `warehouse` equal to `A`:

copy

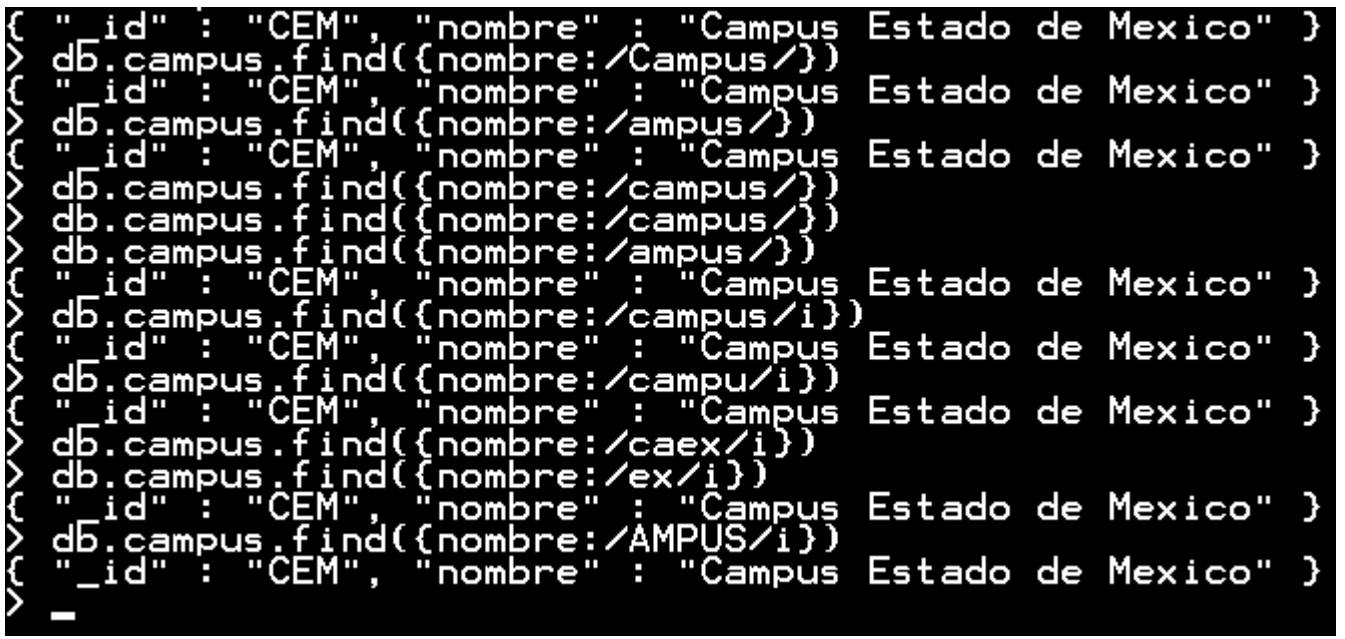
```
db.inventory.find( { "instock.qty": 5, "instock.warehouse": "A" } )
```

MongoDB: Búsquedas Ignore Case

Ejemplo:

db.campus.find(nombre:/campus/i)

‘/’ son comodines, el equivalente a ‘%’ en MySQL



Sin ’i’

Con ’i’

Práctica Grupal