

mongoDB

SQUEMA DESIGN

Enrique Barra

REPASO – QUÉ ES MONGODB

- MongoDB es una tecnología de base de datos NoSQL orientada a documentos
- Una **base de datos** contiene **colecciones**
- Una **colección** esta formada por **documentos**
- Cada **documento** esta compuesto de campos
- Las colecciones pueden ser **indexadas**, lo cual mejora la búsqueda y ordenamiento

REPASO – DIFERENCIAS CON RELACIONAL

- No tienen esquema declarado
 - Aunque todos los documentos veremos que tienen un esquema similar y que conviene pensarlo bien antes
- No hay join
- No hay constraints
- Aunque tenga transactions (desde v4.0) no hay que abusar, implican bloquear varias colecciones y esperas => disparan los tiempos
- La diferencia principal viene del hecho de que las bases de datos relacionales definen columnas a nivel de la tabla mientras que bases de datos orientadas a documentos definen sus campos a nivel de documento

REPASO – PARECIDOS CON RELACIONAL

- Colecciones son como las tablas
- Documentos son como las filas
- Campos del documento con como las columnas
- Tiene índices
- Ya tiene transacciones multidocumentos (aunque cuidado con abusar)

TESTS CONOCIMIENTOS PREVIOS

- Cuál de las siguientes afirmaciones es cierta sobre MongoDB?
 - A. MongoDB es orientado a documentos.
 - B. MongoDB soporta SQL.
 - C. MongoDB tiene schema dinámico.
 - D. MongoDB soporta joins.

TESTS CONOCIMIENTOS PREVIOS

- Cuál de las siguientes afirmaciones es cierta sobre MongoDB?
 - A. MongoDB es orientado a documentos.
 - B. MongoDB soporta SQL.
 - C. MongoDB tiene schema dinámico.
 - D. MongoDB soporta joins.
- En inglés se suele decir schemaless, pero es mejor traducirlo por esquema dinámico

TESTS CONOCIMIENTOS PREVIOS

- Qué característica omite MongoDB para preservar la escalabilidad?
 - A. Joins
 - B. Indices
 - C. Indices secundarios
 - D. Transacciones entre múltiples documentos

TESTS CONOCIMIENTOS PREVIOS

- Qué característica omite MongoDB para preservar la escalabilidad?
 - A. Joins
 - B. Indices
 - C. Indices secundarios
 - D. Transacciones entre múltiples documentos
- Joins no escalan horizontalmente
- Indices dan muy buena performance.
- Los índices secundarios también están soportados
- Transacciones, desde MongoDB 4.0 se soportan sobre multiples documentos.

SCHEMA DESIGN - INTRODUCCIÓN

- En Bases de datos relacionales el modelo de datos es agnóstico de la aplicación
- En Mongo y NOSQL en general estará **muy relacionado modelo de datos y la aplicación**
- Pensaremos mucho en esta relación para diseñar el esquema
 - Qué datos están relacionados y se suelen acceder juntos
 - Qué datos se escriben todo el tiempo
 - Qué datos se leen todo el tiempo
- Concepto clave: Pre-join/embed data
 - Nos permite “vivir sin restricciones”
 - Nos facilita operaciones atómicas y no necesitaremos transacciones
- O'Reilly Webcast: MongoDB Schema Design: How to Think Non-Relational <https://www.youtube.com/watch?v=PIWVFUtBV1Q>

SCHEMA DESIGN - ÍNDICE

- Datos sin relaciones
- Relación 1 a 1
- Relación 1 a N
 - N grande
 - N pequeño
- Relación M a N
- Representar árboles
- Validación

SCHEMA DESIGN - DATOS SIN RELACIONES

- En muchos casos en las colecciones de MongoDB no hay relaciones, está todo dentro de la misma colección
- Ejemplos:

COLECCIÓN LOGS/SESIONES

```
{  
  user_agent: "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101  
Firefox/47.0 Mozilla/5.0 (Macintosh; Intel Mac OS X x.y; rv:42.0) Gecko/20100101  
Firefox/42.0",  
  action: "GET",  
  URL: "http://educainternet.es/workshops/5",  
  datetime: "1489400125995",  
  time_spent_ms: "32500"  
}
```

COLECCIÓN TRAFICO

```
{  
  car: "Toyota Auris",  
  position: { lng: 55.5, lat: 42.3 },  
  city: "Madrid",  
  street: "Calle 30",  
  speed: "12",  
  updated_at: "1489400125995"  
}
```

SCHEMA DESIGN – RELACIÓN 1 A 1

- Podemos ir a embed
- Podemos crear dos documentos con un id
 - pero no existe join, lo tendremos que programar por software
- Dependerá de nuestra app:
 - Cuantas veces se accede a ese contenido, se accede junto o separado?
 - Puede existir un documento sin el otro (por ejemplo curriculum y empleado)
 - ¿Cómo se espera que crezca, al actualizar un documento se actualiza el otro?
 - Tamaño: máximo tamaño de documento 16MB (aunque existe GridFS)
 - Atomicidad: puede nuestra app ser tolerante a un fallo de actualización?

RELACIÓN 1 A 1 SIN EMBED (CON REFERENCIA)

COLECCIÓN USERS

```
{  
  _id: ObjectId("507f1f77bcf86cd799439011"),  
  name: "Enrique Perez",  
  fecha_nacimiento: ISODate("1982-09-19T06:01:17.171Z")  
}
```

COLECCIÓN DIRECCIONES

```
{  
  _id: ObjectId("46cd799439011507f1f77b33"),  
  user_id: ObjectId("507f1f77bcf86cd799439011"),  
  calle: "Gran Via 12",  
  city: "Madrid",  
  pais: "España",  
  cp: "28026"  
}
```

RELACIÓN 1 A 1 CON EMBED

COLECCIÓN USERS

```
{
  _id: ObjectId("507f1f77bcf86cd799439011"),
  name: "Enrique Perez",
  fecha_nacimiento: ISODate("1982-09-19T06:01:17.171Z"),
  direccion: {
    calle: "Gran Via 12",
    city: "Madrid",
    pais: "España",
    cp: "28026"
  }
}
```

RELACIÓN 1 A N (CON N GRANDE)

- Con N **grande**
- Según la app que estemos desarrollando si se accede a las vistas de los “N” sueltas se harán dos colecciones
- Ej:
 - Ciudadanos y ciudad
 - Editorial y libros
- Convendrá añadir un índice sobre los campos de relación entre las colecciones

Opción 1: En la editorial añadimos un array de ids de los libros

COLECCIÓN EDITORIALES

```
{
  name: "O'Reilly Media",
  founded: 1980,
  location: "CA",
  books: [12346789, 234567890, ...]
}
```

COLECCIÓN LIBROS

```
{
  _id: 123456789,
  title: "MongoDB: The Definitive Guide",
  author: [ "Kristina Chodorow", "Mike Dirolf" ],
  published_date: ISODate("2010-09-24"),
  pages: 216,
  language: "English"
}

{
  _id: 234567890,
  title: "50 Tips and Tricks for MongoDB Developer",
  author: "Kristina Chodorow",
  published_date: ISODate("2011-05-06"),
  pages: 68,
  language: "English"
}
```


Opción 2: Para evitar que el array books crezca y cambie a cada libro que se añada, se puede añadir mejor publisher_id

```
{  
  _id: "oreilly",  
  name: "O'Reilly Media",  
  founded: 1980,  
  location: "CA"  
}
```

COLECCIÓN EDITORIALES

```
{  
  _id: 123456789,  
  title: "MongoDB: The Definitive Guide",  
  author: [ "Kristina Chodorow", "Mike Dirolf" ],  
  published_date: ISODate("2010-09-24"),  
  pages: 216,  
  language: "English",  
  publisher_id: "oreilly"  
}  
  
{  
  _id: 234567890,  
  title: "50 Tips and Tricks for MongoDB Developer",  
  author: "Kristina Chodorow",  
  published_date: ISODate("2011-05-06"),  
  pages: 68,  
  language: "English",  
  publisher_id: "oreilly"  
}
```

COLECCIÓN LIBROS

RELACIÓN 1 A N (CON N PEQUEÑO)

- En el caso de 1 a POCOS
 - Suele ser típico embed (desde el POCOS al uno)
- Por ejemplo post y comentarios
 - Haremos una única colección post
 - Cada post dentro tendrá sus comentarios
 - Cada comentario a su vez tendrá dentro otros comentarios (si se permiten réplicas)
- Por ejemplo estudiante y profesores
 - En el estudiante tendremos un campo “teachers” que es un array de sus profesores
- Por ejemplo usuario y direcciones de email o direcciones de casa, trabajo etc.

```

{
  id: ObjectId("507f1f77bcf86cd799439011"),
  discussion_id: ObjectId("507f1f77bcf86cd799439012"),
  parent_id: ObjectId("507f1f434ddddd799439014"),
  slug: 'nombre_corto',
  posted: ISODate("2015-11-19T06:01:17.171Z"),
  title: 'Mi opinion',
  text: 'Este post explica mi opinión de ...',
  author: {
    id: ObjectId("507f1f77bcf86cd799423543"),
    name: 'Ricky'
  },
  comments: [
    { posted: ISODate("2015-12-19T06:01:17.171Z"),
      author: { id: ObjectId("507f1f77bcf86cd799433224"), name: 'Jose' },
      text: 'Me parece muy bien tu opinión. Mandame un email.',
      replies: [ {}, {} ]
    },
    { posted: ISODate("2015-12-19T06:02:17.171Z"),
      author: { id: ObjectId("507f1f77bcf86cd7994334233"), name: 'Enrique' },
      text: 'Yo no estoy deacuerdo.',
      replies: [ {} ]
    }
  ]
}

```

COLECCIÓN POSTS

COLECCIÓN USUARIOS

```
{
  _id: "joe",
  name: "Joe Bookreader",
  addresses: [
    {
      street: "123 Fake Street",
      city: "Faketon",
      state: "MA",
      zip: "12345"
    },
    {
      street: "1 Some Other Street",
      city: "Boston",
      state: "MA",
      zip: "12345"
    }
  ]
}
```

SCHEMA DESIGN – RELACIÓN M A N

○ Relación M a N

- Ejs: libros y autores. Profesores, estudiantes y asignaturas
- Esta relación es totalmente dependiente de la app que estemos haciendo
- Se puede incrustar documentos “autores” en los documentos “libros”, pero para añadir un autor al sistema tendrá que existir un libro suyo. Al revés igual.
- Es muy normal en este caso que existan dos colecciones separadas y en cada una poner un array de ids de documentos de la otra colección (a esto se llama LINK en MongoDB)

Relación M a N

ESTUDIANTES

```
{  
  id: ObjectId("507f1f77bcf86cd799439011"),  
  nombre: "Enrique Perez",  
  notas: [ {}, {}],  
  slug: 'nombre_corto',  
  fecha_matriculacion: ISODate("2015-11-19T06:01:17.171Z"),  
  fecha_nacimiento: ISODate("1982-09-19T06:01:17.171Z"),  
  asignaturas_matriculadas: [345, 234, 898, 888, 341, 123],  
  profesores: [1, 5, 7]  
}
```

ASIGNATURAS

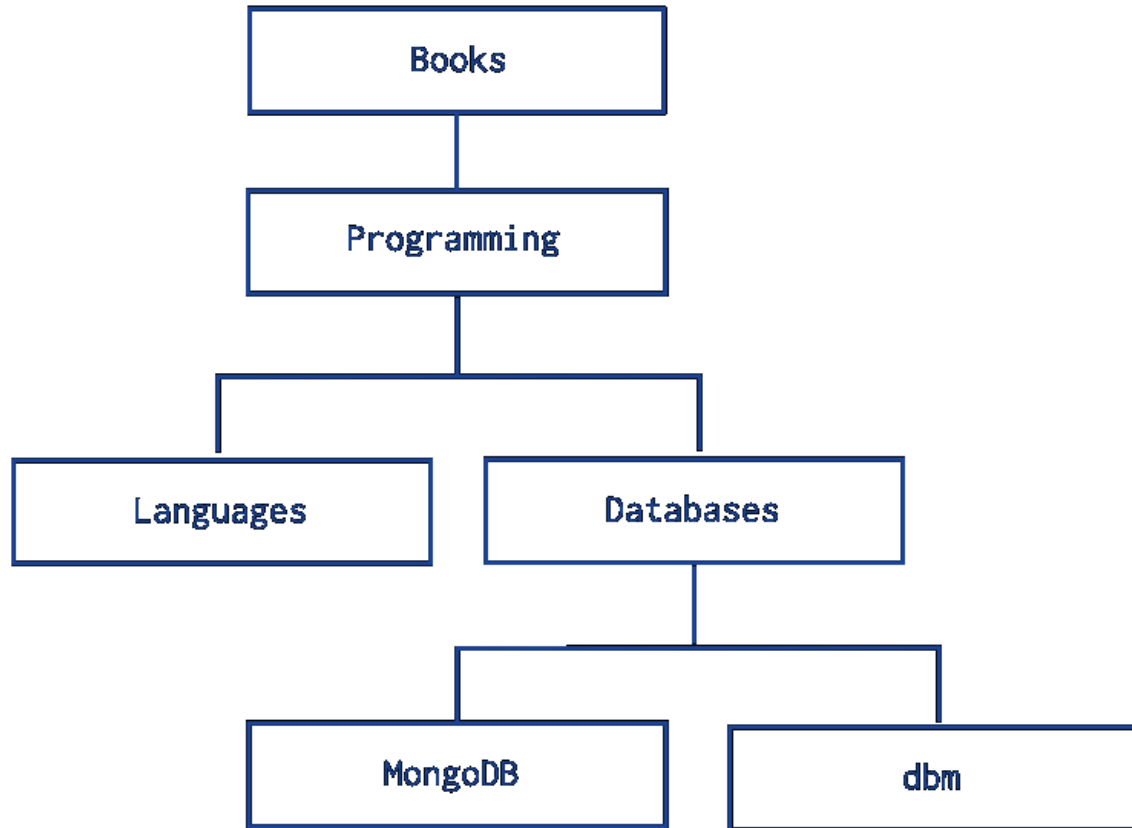
```
{  
  id: ObjectId("507f1f77bcf86cd799435654"),  
  nombre: "Bases de datos",  
  slug: "BBDD",  
  profesores: [1, 3],  
  tipo: "troncal",  
  curso: "segundo",  
  comentarios: "Asignatura de master"  
}
```

PROFESORES

```
{  
  id: 1,  
  nombre: "Joaquin Solis",  
  slug: 'joaquinsol',  
  fecha_nacimiento: ISODate("1982-09-19T06:01:17.171Z")  
}
```

REPRESENTAR ÁRBOLES

- Ejemplo: Productos y categorías



OPCIONES PARA REPRESENTAR ÁRBOLES

○ Parent References

- { _id: "MongoDB", parent: "Databases" }

○ Child References

- { _id: "Databases", children: ["MongoDB", "dbm"] }
- { _id: "MongoDB", children: [] }

○ Array of Ancestors

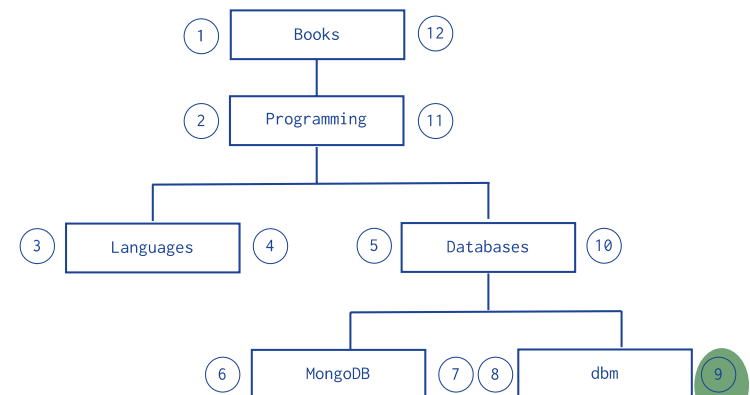
- { _id: "MongoDB", ancestors: ["Books", "Programming", "Databases"], parent: "Databases" }

○ Materialized Paths

- { _id: "MongoDB", path: ",Books,Programming,Databases," }

○ Nested Sets

- { _id: "MongoDB", parent: "Databases", left: 6, right: 7 }



MODEL TREE STRUCTURES WITH AN ARRAY OF ANCESTORS

```
{ _id: "MongoDB", ancestors: [ "Books", "Programming",  
"Databases" ], parent: "Databases" }
```

```
{ _id: "dbm", ancestors: [ "Books", "Programming", "Databases" ],  
parent: "Databases" }
```

```
{ _id: "Databases", ancestors: [ "Books", "Programming" ],  
parent: "Programming" }
```

```
{ _id: "Languages", ancestors: [ "Books", "Programming" ],  
parent: "Programming" }
```

```
{ _id: "Programming", ancestors: [ "Books" ], parent: "Books" }
```

```
{ _id: "Books", ancestors: [ ], parent: null }
```

VALIDACIÓN

- MongoDB proporciona la capacidad de realizar la validación del esquema durante las actualizaciones e inserciones.
- Las reglas de validación son por colección.
- Se pueden configurar de muy laxas a muy estrictas
 - Que se permita guardar documentos inválidos pero quede registrado en el log
 - Que se permitan campos extra y valide solo alguno
 - Saltarse la validación (`bypassDocumentValidation`) si el usuario tiene el permiso o privilegio concedido

SCHEMA DESIGN – DIFERENCIAS CON EL MODO TRADICIONAL (RDBMS)

| Tradicional | MongoDB |
|-------------------------------------|--|
| No importa la aplicación | Siempre depende de la aplicación |
| Solo importan los datos | No solo importan los datos, sino cómo se usan |
| Solo es relevante durante el diseño | Es relevante durante toda la historia de la aplicación |

PATRONES

- <https://www.mongodb.com/blog/post/building-with-patterns-a-summary>

| | | Use Case Categories | | | | | | |
|----------|---------------------|---------------------|--------------------|--------------------|--------|-----------------|---------------------|-------------|
| | | Catalog | Content Management | Internet of Things | Mobile | Personalization | Real-Time Analytics | Single View |
| Patterns | Approximation | ✓ | | ✓ | ✓ | | ✓ | |
| | Attribute | ✓ | ✓ | | | | | ✓ |
| | Bucket | | | ✓ | | | ✓ | |
| | Computed | ✓ | | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Document Versioning | ✓ | ✓ | | | ✓ | | ✓ |
| | Extended Reference | ✓ | | | ✓ | | ✓ | |
| | Outlier | | | ✓ | ✓ | ✓ | | |
| | Preallocated | | | ✓ | | | ✓ | |
| | Polymorphic | ✓ | ✓ | | ✓ | | | ✓ |
| | Schema Versioning | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| | Subset | ✓ | ✓ | | ✓ | ✓ | | |
| | Tree and Graph | ✓ | ✓ | | | | | |

A decorative graphic on the left side of the slide. It features several vertical stripes in shades of green and grey. Overlaid on these stripes are several green circles of different sizes. One large circle is positioned near the top, and several smaller circles are arranged below it, some overlapping the stripes.

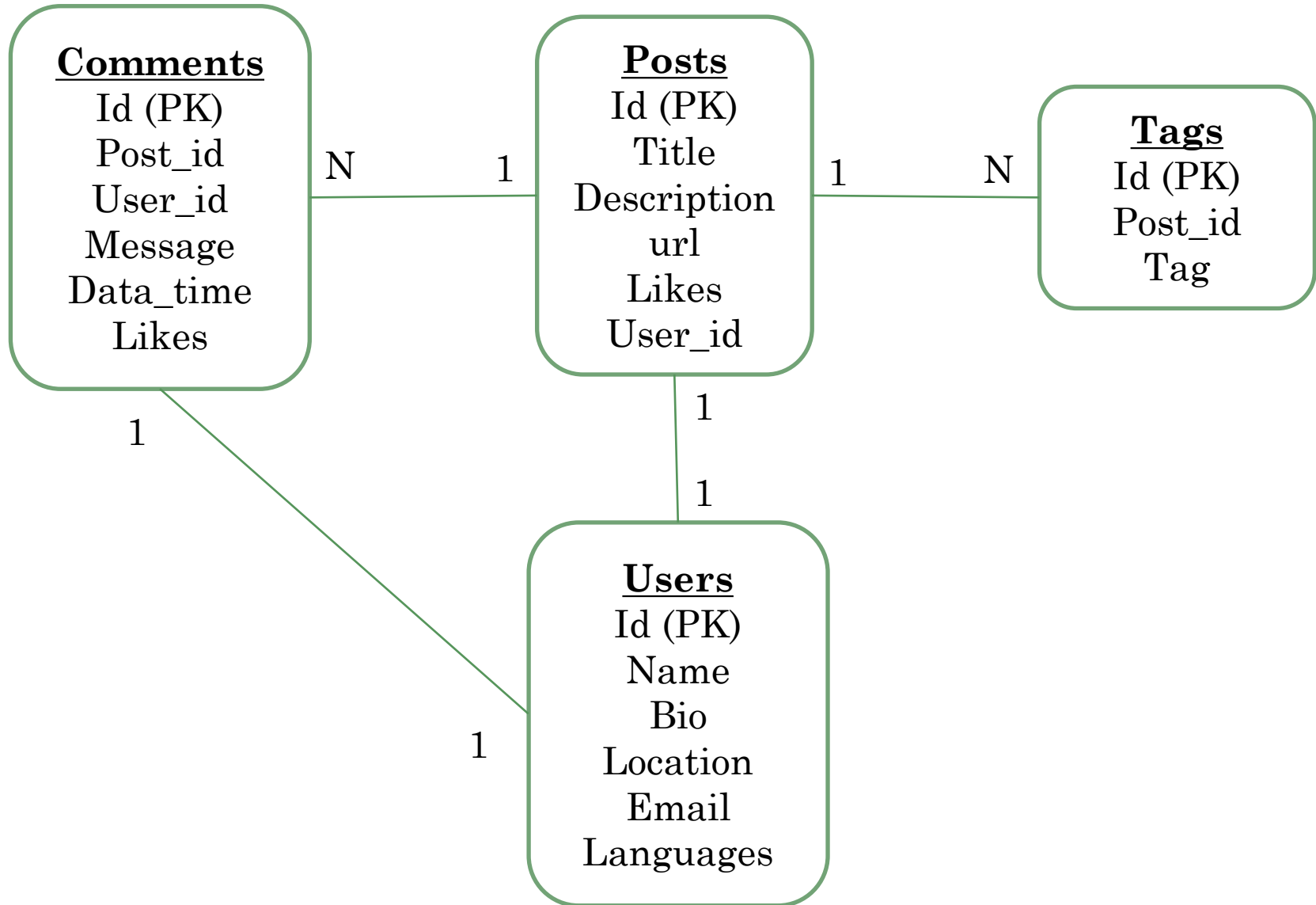
EJEMPLOS

29

EJEMPLO 1 - BLOG

- Supongamos que un cliente necesita un diseño de base de datos para su blog y quiere ver las diferencias entre relacional (RDBMS) y MongoDB
- Requisitos:
 - Cada post tiene un título único, descripción y URL
 - Cada post tiene una o más tags
 - Cada post tiene el nombre de su autor y el número total de likes
 - Cada post tiene comentarios hechos por usuarios con su nombre, mensaje, likes y fechas
 - En cada post puede haber cero o más comentarios

RDBMS



USERS

```
{
  _id: USER_ID,
  name: "Enrique Barra",
  bio: "...",
  location:"Madrid ...",
  email: "ebarra@dit.upm.es",
  languages: ["Spanish",
"English"],
  ...,
}
```

POSTS

```
{
  _id: POST_ID,
  title: TITLE_OF_POST,
  description: POST_DESCRIPTION,
  by: USER_ID,
  url: URL_OF_POST,
  tags: [TAG1, TAG2, TAG3],
  likes: TOTAL_LIKES,
  comments: [
    {
      user_id: USER_ID1,
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    },
    {
      user_id: USER_ID2,
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    }
  ]
}
```


USERS

```
{
  _id: USER_ID,
  name: "Enrique Barra",
  bio: "...",
  location:"Madrid ...",
  email: "ebarra@dit.upm.es",
  languages: ["Spanish",
"English"],
  ...,
}
```

POSTS (WITH USER NAME INCLUDED)

```
{
  _id: POST_ID,
  title: TITLE_OF_POST,
  description: POST_DESCRIPTION,
  by:{ id: USER_ID, name: "Enrique
Barra"},
  url: URL_OF_POST,
  tags: [TAG1, TAG2, TAG3],
  likes: TOTAL_LIKES,
  comments: [
    {
      user :{ id: USER_ID1, name:
"Pepe Perez"},
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    },
    {
      user :{ id: USER_ID2, name:
"Jose Risco"},
      message: TEXT,
      dateCreated: DATE_TIME,
      like: LIKES
    }
  ]
}
```

EJEMPLO 2 – PRODUCTOS Y PARTES

- Tenemos que modelar un sistema de pedidos, donde tenemos productos y las piezas que lo forman
- Cada producto está formado por cientos de piezas y cada pieza puede estar en distintos productos (relación M a N)

```
> db.products.findOne()
{
  name: 'left-handed smoke shifter',
  manufacturer: 'Acme Corp',
  catalog_number: 1234,
  parts: [ // array of references to Part documents
    ObjectID('AAAA'), // reference to the #4 grommet above
    ObjectID('F17C'), // reference to a different Part
    ObjectID('D2AA'),
    // etc
  ]
}

> db.parts.findOne()
{
  _id : ObjectID('AAAA'),
  partno : '123-aff-456',
  name : '#4 grommet',
  qty: 94,
  cost: 0.94,
  price: 3.99
}
```

- Como será el join a nivel de aplicación

```
// Fetch the Product document identified by this catalog number  
> product = db.products.findOne({catalog_number: 1234});
```

```
// Fetch all the Parts that are linked to this Product  
> product_parts = db.parts.find({_id: { $in : product.parts } }  
) .toArray() ;
```

EJEMPLO 3 – LOGS Y MÁQUINAS

- Queremos modelar un sistema de logging de un cloud
- Cada máquina emite muchos/muchísimos logs (1 a N grande)
- Como hay una limitación de 16MB por documento no tendría sentido meter todos los logs en el documento del host o máquina. Además cada vez que llega un log no queremos guardar todo el documento entero una y otra vez

```
> db.hosts.findOne()  
{  
  _id : ObjectId('AAAB'),  
  name : 'goofy.example.com',  
  ipaddr : '127.66.66.66'  
}
```

```
>db.logmsg.findOne()  
{  
  time : ISODate("2014-03-28T09:42:41.382Z"),  
  message : 'cpu is on fire!',  
  host: ObjectId('AAAB')      // Reference to the Host document  
}
```

- Cómo se buscarían los últimos 5000 logs de una máquina

```
// find the parent 'host' document
```

```
> host = db.hosts.findOne({ipaddr : '127.66.66.66'});
```

```
// assumes unique index
```

```
// find the most recent 5000 log message documents linked to that host
```

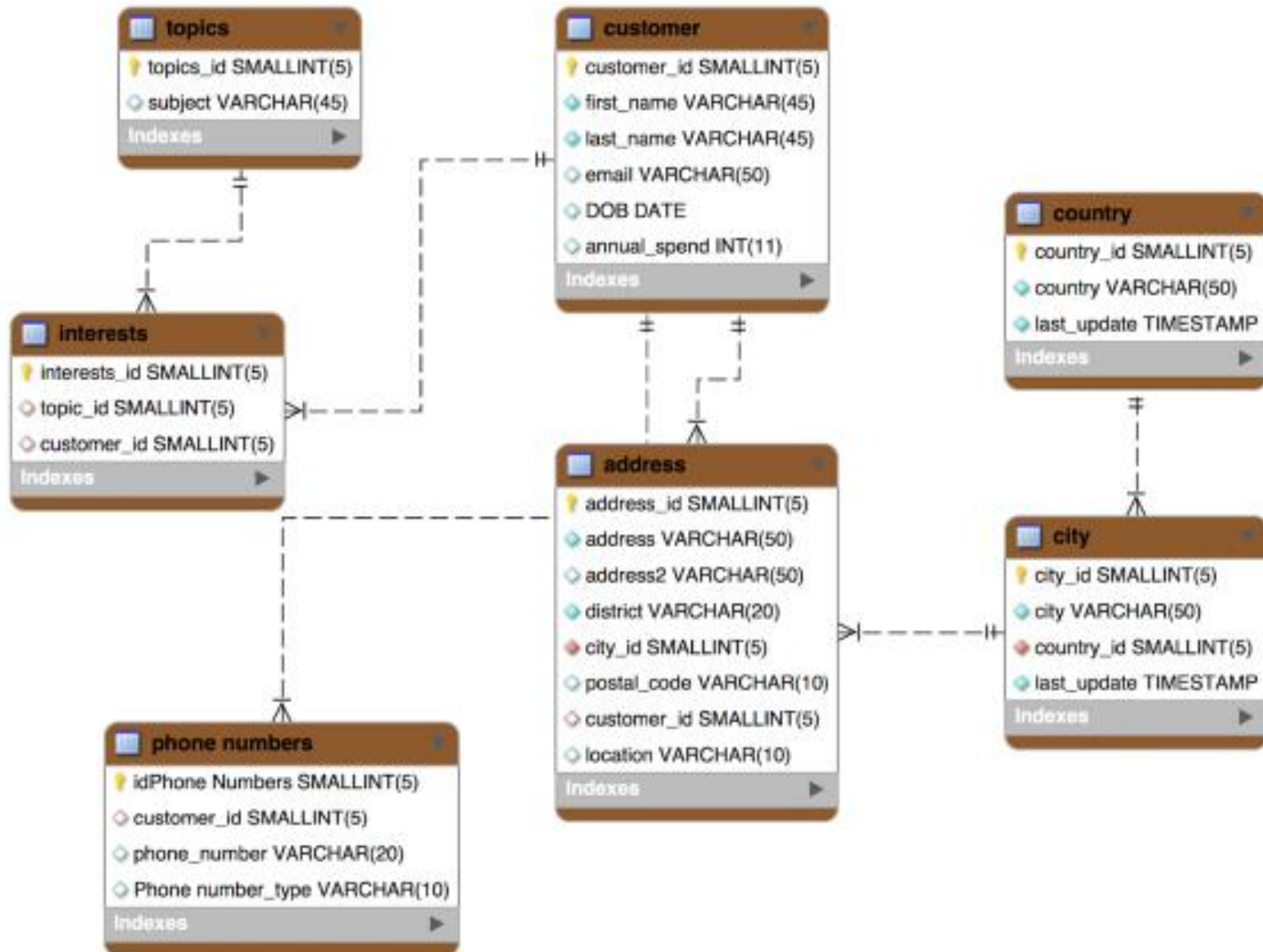
```
> last_5k_msg = db.logmsg.find({host: host._id}).sort({time : -1}).limit(5000).toArray()
```

SE PODRÍA HACER CON UN PATRÓN

- Ver el **patrón** bucket como ejemplo de logs y máquinas o sensores:
- <https://www.mongodb.com/blog/post/building-with-patterns-the-bucket-pattern>
- Se hace un doc cada hora y en él se hacen incluso cálculos al cerrarlo y abrir la siguiente hora

```
{
  sensor_id: 12345,
  start_date: ISODate("2019-01-31T10:00:00.000Z"),
  end_date: ISODate("2019-01-31T10:59:59.000Z"),
  measurements: [
    {
      timestamp: ISODate("2019-01-31T10:00:00.000Z"),
      temperature: 40
    },
    {
      timestamp: ISODate("2019-01-31T10:01:00.000Z"),
      temperature: 40
    },
    ...
    {
      timestamp: ISODate("2019-01-31T10:42:00.000Z"),
      temperature: 42
    }
  ],
  transaction_count: 42,
  sum_temperature: 2413
}
```


EJEMPLO 4 – CLIENTE Y SUS DATOS



USER

```
{
  "_id": ObjectId("5ad88534e3632e1a35a58d00"),
  "name": { "first": "John", "last": "Doe" },
  "address": [
    { "location": "work",
      "address": {
        "street": "16 Hatfields",
        "city": "London",
        "postal_code": "SE1 8DJ"},
      "geo": { "type": "Point", "coord": [
        51.5065752, -0.109081]}}},
    + {...}
  ],
  "phone": [
    { "location": "work",
      "number": "+44-1234567890"},
    + {...}
  ],
  "dob": ISODate("1977-04-01T05:00:00Z"),
  "interests": ["tennis", "football", ...]
}
```

A decorative graphic on the left side of the slide. It features several vertical stripes in shades of green and grey. Overlaid on these stripes are several green circles of different sizes. One large circle is positioned near the top left, and several smaller circles are arranged in a descending pattern towards the bottom left. The text 'EJEMPLOS DE APPS' is positioned to the right of these circles.

EJEMPLOS DE APPS

TAREA PARA CLASE

- Modelar el esquema para AirBnb
- Tenemos:
 - Usuarios (pueden ser viajero o propietario)
 - Casas
 - Reservas
 - Comentarios de las reservas
 - Comentarios sobre el usuario
 - Likes de los usuarios


BÚSQUEDA

12:29

0,2kB/s

99


← Madrid · Alojamientos



SOL

Lollipop Flats: Count's secret lair in Plaza Mayor

★ 4,65 (34)



EXPLORAR

GUARDADOS

VIAJES

MENSAJES

ENTRAR

PERFIL

12:30


0,1kB/s


98

←

Hola, soy Maria Teresa

Se unió en junio, 2017


 Verificada

 275 evaluaciones

Acerca de

“

Soy una persona a quien le gusta conocer otras personas

 Vive en Madrid, España

PERFIL (SCROLL)

12:30

0,5kB/s

98

Anuncio de Maria Teresa



HABITACIÓN PRIVADA · 1 CAMA

Madrid centro habitación con dos balcones

★★★★★ 275

275 evaluaciones

De los huéspedes

Muy buena ubicación en casa de Teresa. Ella y su familia muy atentos a nuestras inquietudes, nos ayudaron a ubicarnos. Siempre amables.

 Alessis, Cali, Colombia

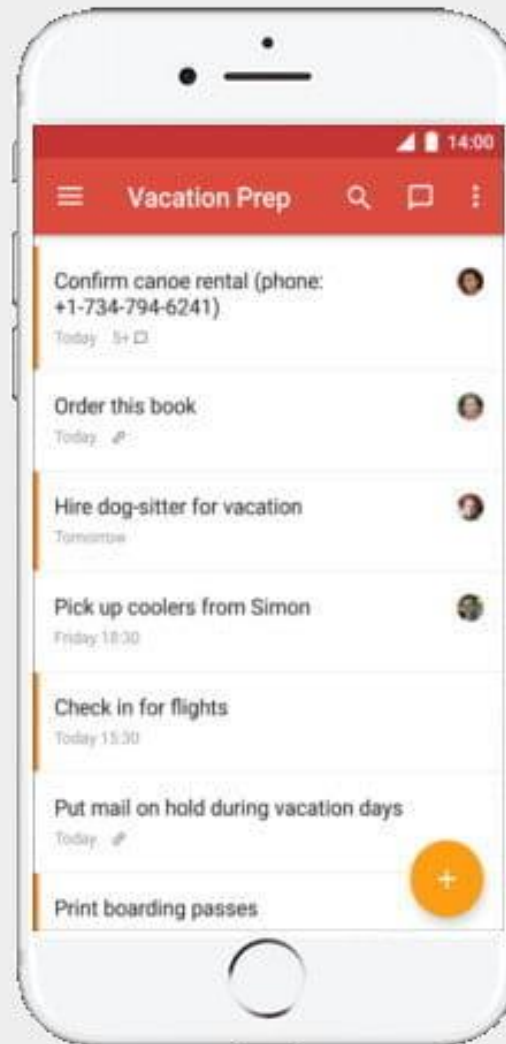
USERS

```
{
  _id: 12
  name: "Enrique Barra",
  bio: "...",
  location:"Madrid ...",
  email: "ebarra@dit.upm.es",
  languages: ["Spanish", "English"],
  ...,
  trips: [
    {
      home_id: 56,
      start_date: DATETIME,
      end_date: DATETIME,
      reservation_date: DATETIME,
      ...
    }, . . . ],
  likes: [
    {
      home_id: 56,
      start_date: DATETIME,
      end_date: DATETIME,
      reservation_date: DATETIME,
      ...
    }, . . . ],
  reviews: [. . .]
}
```

HOMES

```
{
  _id: 34
  owner_id: 12
  name: "Casa Rural Kike",
  description: "...",
  location:"Madrid ...",
  price: 100€,
  rooms: 3,
  ...,
  reviews: [
    {
      user_id: 333,
      message: "TEXT",
      trip: 2
    },
    . . .
  ]
}
```

APP LISTA DE TAREAS



PRÁCTICA 1

- **Ver enunciado**