Gestión de datos

Modelos de datos y modelado (2)

Lorena Etcheverry (lorenae@fing.edu.uy)
Instituto de Computación, FING, UdelaR

Recapitulemos

Modelo de datos: definición

Los modelos de datos son **lenguajes** usados para especificar y manipular Bases de Datos

Un Modelo de Datos permite expresar:

- Estructuras: Elementos de los problemas.
- Restricciones: Reglas que deben cumplir los datos para que la base sea considerada válida.
- Operaciones: Insertar, borrar y consultar la BD.

Clasificación de los modelos de datos: nivel de abstracción

Conceptuales: Representan la realidad independientemente de cualquier implementación de la base de datos

Lógicos: Son implementados en un manejador de bases de datos particular.

Físicos: Corresponden a cómo está implementado el manejador de bases de datos (estructuras de datos)

Modelos Conceptuales

Nos concentraremos en el modelo Entidad-Relación

Para profundizar/repasar referirse al material recomendado en el EVA:

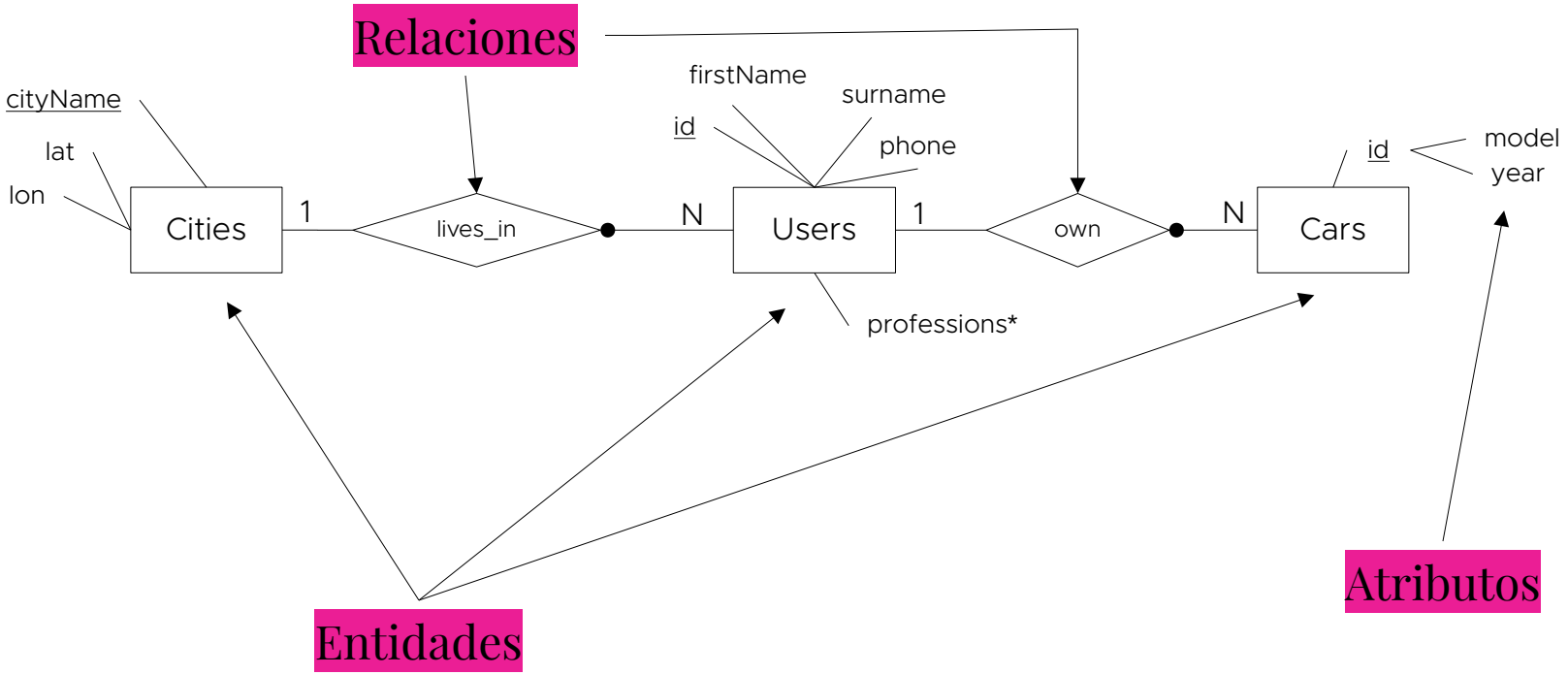
- los videos del curso Fundamentos de Bases de Datos (disponibles en OpenFING)
- Capítulo 7 (Fundamentals of Database Systems, Elmasri and Navathe, 6th edition)

Ejemplo Modelo Entidad-relación

Queremos modelar una base de datos de propietarios de automóviles. De cada usuario se conoce su nombre y apellido, un número de teléfono y una o más profesiones. Cada usuario se identifica por un número de usuario, que es único. Se conoce además la ciudad en donde vive cada usuario, y para cada ciudad (identificada por su nombre) se conocen sus coordenadas (lat y long). Además interesa registrar los vehículos que posee cada usuario. Cada vehículo se identifica por el modelo y el año.

1- identifiquemos entidades (conceptos relevantes de la realidad)
2- identifiquemos atributos de las entidades
3- relaciones entre entidades y restricciones

Un posible diseño conceptual



Modelos lógicos

El **modelo Relacional** (Codd, 1970) es un ejemplo famoso y exitoso de modelo lógico.

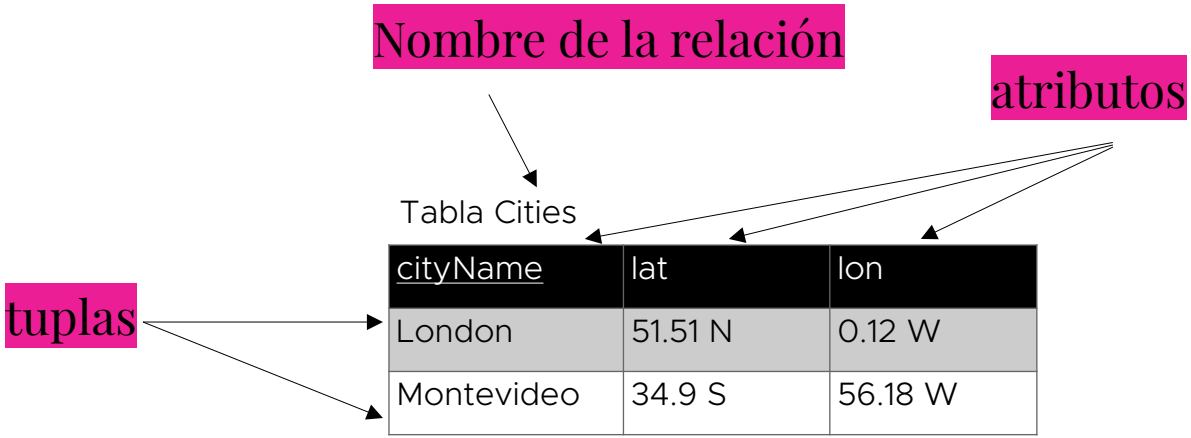
Los datos se representan como una colección de **relaciones** (tablas).

Cada relación tiene un nombre y un conjunto de atributos.

Las instancias de la relación (valores) son conjuntos de tuplas (sin repetidos, sin orden).

Además hay restricciones (ej: clave primaria)

Ejemplo Modelo Relacional



Ejemplo Modelo Relacional (cont)

Tabla Cities

cityName	lat	lon
London	51.51 N	0.12 W
Montevideo	34.9 S	56.18 W

Tabla Users

id	firstName	surname	phone	city
1	Paul	Miller	5111111	London
2	Laura	Rodriguez	333333	Montevideo

Tabla User_professions

userId	profession
1	banking
1	finance
1	trader

Tabla Cars

userId	model	year
1	Bentley	1973
1	Rolls Royce	1965

Diseño lógico de BDs relacionales

- Hay heurísticas que permiten obtener un diseño a partir del modelo conceptual, garantizando cierta calidad del diseño obtenido
  - Pasaje de MER a ER (ver material)
- Esta calidad es independiente de las consultas a realizar.
- Las Formas Normales garantizan propiedades para todas las instancias

Pero hay más modelos lógicos ...

- bases de datos documentales
- bases de datos de grafos

Una base de datos de documentos
es una base **no relacional**
(noSQL)
que almacena datos como
documentos estructurados,
típicamente
XML y **JSON**

XML: la vedette de los 2000s

Un lenguaje de marcado para representar datos.

XML 1.0 W3C recommendation en Febrero 1998.

¿para qué se pensó y para que se usa/usó?

Separar datos de presentación en la web 2.0

Agregar metadatos y esquema (dar estructura).

Intercambio de datos entre aplicaciones.

Interoperabilidad!

Estándares asociados a XML

XPath

Una sintaxis para recuperar partes del documento: filtros, comodines.

XQuery

Un lenguaje de consulta sobre XML. El SQL de XML!

XML Schema

Un documento XML especial que describe la estructura de otro documento XML

XSLT (eXtensible Stylesheet Language Transformations)

Un lenguaje de transformaciones. Permite generar otros formatos (ej: HTML)

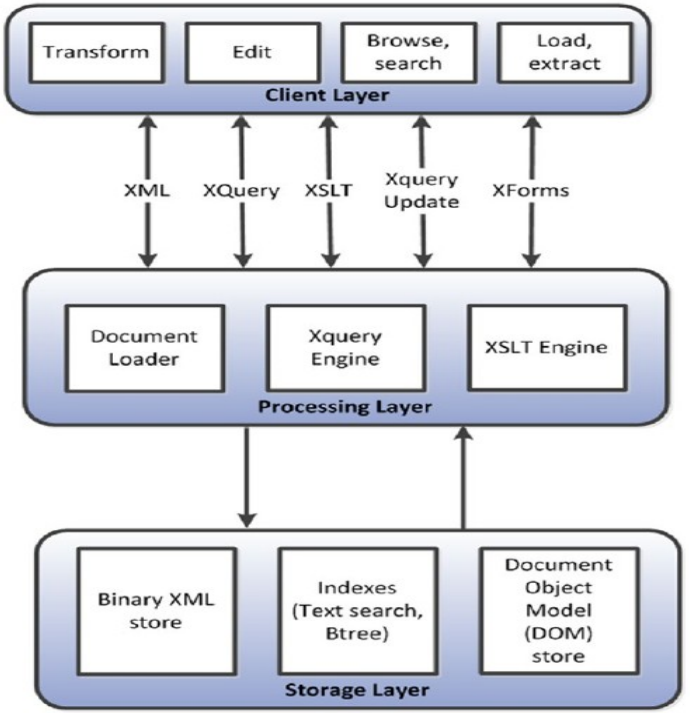
DOM (Document Object Model)

Una API orientada a objetos para interactuar con documentos XML.

Y como muchas aplicaciones generaban XML …

Surgen las BDs de documentos para gestionar colecciones de documentos XML.

Muchas soluciones comerciales.



pero XML tiene algunos inconvenientes...

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup
                Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
            languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

"Some languages can be read by humans, but not by machines,

while others can be read by machines but not by humans.

XML solves this problem by being readable to neither."

Post by deleted, Reddit [1]

[1] https://www.reddit.com/r/ProgrammerHumor/comments/1mbgih/xml/

Mientras tanto, en el mundo de los desarrolladores de aplicaciones web ...

Javascript se impone como lenguaje de programación tanto del lado del cliente como del servidor (ej: Node.js).

AJAX: mensajería asíncrona entre cliente y servidor (originalmente pensado sobre XML).

Aparece Javascript Object Notation (JSON) como mecanismo para serializar objetos.

JSON: The Fat-Free alternative to XML [1]

[1]http://www.json.org/xml.html

Estructura de un doc JSON

objeto

clave

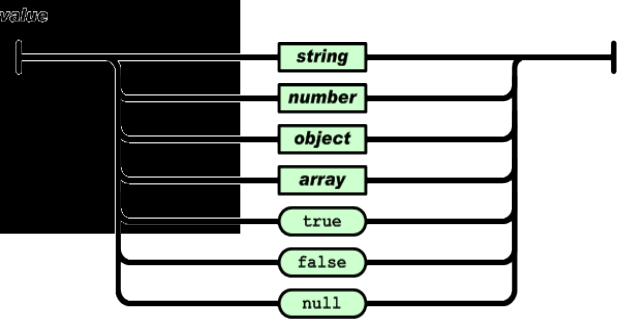
valor

arreglo (lista de valores)

```
{ "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup
                        Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to
                     create markup languages such as
DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}
```

Las *claves* son *strings*.

Los *valores* pueden ser cualquier elemento de la notación.



XML vs JSON (1)


xml
Término de búsqueda
json
Término de búsqueda
+ Añadir comparación
Todo el mundo
2004 - hoy
Todas las categorías
Búsqueda web
Interés a lo largo del tiempo
100
75
50
25
Media
1 ene 2004
1 mar 2011
mar 2022
xml 10
json 18
Los valores de este punto tienen datos parciales (incompletos)

XML vs JSON (2)

```
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup
                Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
            languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

```
{ "glossary": {
    "title": "example glossary",
    "GlossDiv": {
      "title": "S",
      "GlossList": {
        "GlossEntry": {
          "ID": "SGML",
          "SortAs": "SGML",
          "GlossTerm": "Standard Generalized Markup
                        Language",
          "Acronym": "SGML",
          "Abbrev": "ISO 8879:1986",
          "GlossDef": {
            "para": "A meta-markup language, used to
                     create markup languages such as
                     DocBook.",
            "GlossSeeAlso": ["GML", "XML"]
          },
          "GlossSee": "markup"
        }
      }
    }
  }
}
```

Algunas ventajas de JSON

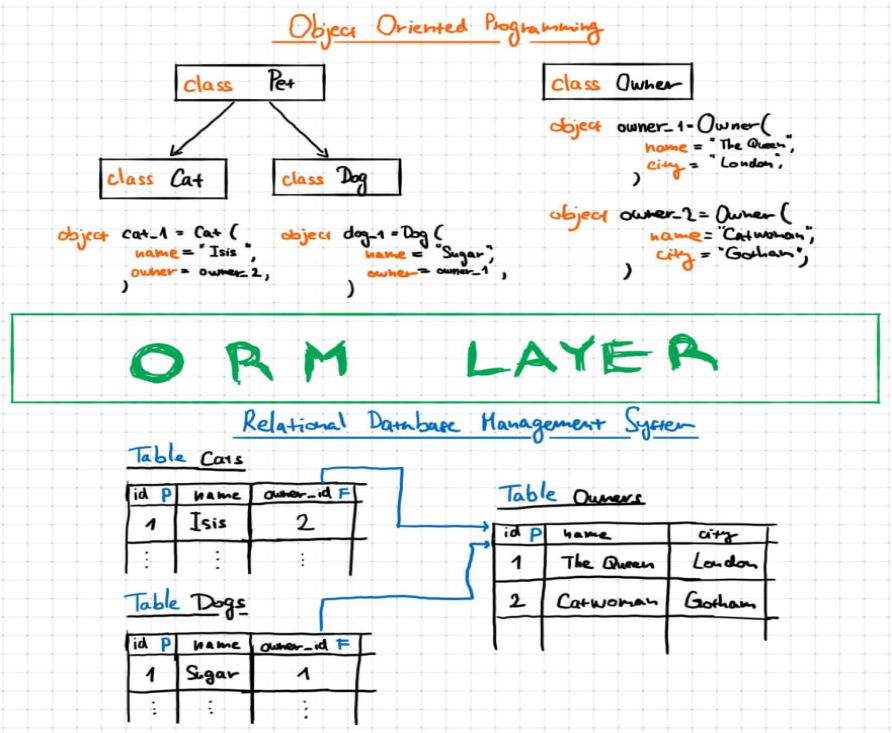
- JSON es **menos verboso**: más liviano para el intercambio de datos y más rápido su parseo.

- JSON representa **directamente** el objeto: más sencillo para el desarrollador

- No tengo un esquema fijo (salvo q use JSON Schema), pero tengo **algo de estructura**.
- Si desarrollo en Javascript, intercambio JSON y almaceno JSON tengo un *full stack* **Javascript**.

- Si desarrollo OO y almaceno JSON **no tengo** ***impedance mismatch*** entre el modelo OO y el relacional.

Object- relational impedance mismatch



Describe la dificultad que surge al tratar de integrar sistemas de bases de datos relacionales con lenguajes de programación orientados a objetos.

Algunos “problemas”

1. problemas de granularidad

2. problema de la herencia

3. problema de la identidad

4. problema de la asociación y navegación de datos

Ireland, C., & Bowers, D. (2015, May). Exposing the myth: object-relational impedance mismatch is a wicked problem. In DBKDA 2015, The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications (pp. 21-26). IARIA XPS Press.

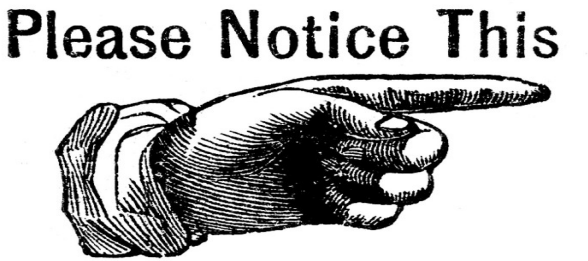
Diseño de bases de datos de documentos

BD relacional

BD documental

Tabla	Colección
Fila (tupla)	Documento
Columna (atributo)	Campo (*field*)
SQL Join	Documentos embebidos y referencias
SQL Group-by (agregación)	*Aggregation pipeline*

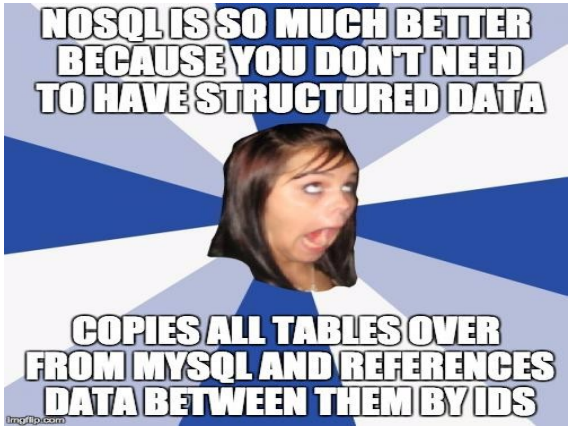
- Los documentos dentro de una colección pueden tener campos diferentes (diferente esquema).
- Se *puede* exigir la validación de cierto esquema definido con JSON Schema
  - Ejemplos de esquemas



La ausencia de un esquema fijo (*schema-less*) no quiere decir que no exista una etapa de diseño y modelado

Diseño de BDs documentales

- No hay heurísticas que permitan pasar desde conceptual a documental como en el caso relacional que garanticen la calidad de la solución
- El diseño depende del tipo de operaciones que voy a hacer además de la naturaleza de los datos
- Diseñar “a la relacional” no suele ser lo mejor



Diseño de BDs documentales

¿Cuáles son las variables que tenemos?

- Decidir cómo se van a representar los elementos de la realidad.
- Decidir cómo representar las relaciones entre elementos.

Estrategias para representar relaciones

```
{
  "imdbID":"tt0944947",
  "Type":"series",
  "Title":"Game of Thrones",
  "Genre":"Adventure, Drama, Fantasy",
  "Actors":"Peter Dinklage, Lena Headey, Emilia Clarke, Kit Harington",
  "imdbRating":"9.5",
  "imdbVotes":"1,031,056"
}
{
  "imdbID":"tt1877832",
  "Type":"movie"
  "Title":"X-Men: Days of Future Past",
  "Year":"2014",
  "Genre":"Action, Adventure, Fantasy",
  "Actors":"Hugh Jackman, Michael Fassbender, Jennifer I
  "imdbRating":"8.0",
  "imdbVotes":"514,203"
}
```

Estrategia 1:
documentos embebidos

```
{
  "imdbID":"tt0944947",
  "Type":"series",
  "Title":"Game of Thrones",
  "Genre":"Adventure, Drama, Fantasy",
  "Actors":
       [
          { “imdbID”: “nm0227759” , “name”: "Peter Dinklage”},
          { “imdbID”: “nm0372176” , “name”: "Lena Headey”},
          { “imdbID”: “nm3229685” , “name”: "Kit Harington"}
         ]
  "imdbRating":"9.5",
  "imdbVotes":"1,031,056"
}
{
  "imdbID":"tt1877832",
  "Type":"movie"
  "Title":"X-Men: Days of Future Past",
  "Year":"2014",
  "Genre":"Action, Adventure, Fantasy",
  "Actors":
       [
          { “imdbID”: “nm0413168” , “name”: "Hugh Jackman”},
          { “imdbID”: “nm1055413”, “name”: "Michael Fassbender”},
          { “imdbID”: “nm2225369”, “name”: "Jennifer Lawrence”},
          { “imdbID”: “nm0227759” , “name”: "Peter Dinklage”}
         ]
  "imdbRating":"8.0",
  "imdbVotes":"514,203"
}
```

Documentos embebidos

VENTAJAS

- Recupero toda la información relevante en una sola consulta.
  - Evita implementar joins en el código de la aplicación o utilizar $lookup.
- Actualizo la información relacionada como una única operación atómica.
  - Por defecto, todas las operaciones CRUD sobre un mismo documento son compatibles con ACID.

LIMITACIONES

- Duplicación de datos
- Los documentos grandes suponen más sobrecarga si la mayoría de los campos no son relevantes.
  - Se puede aumentar el rendimiento de las consultas limitando el tamaño de los docs
- MongoDB tiene un límite de tamaño de documento de 16 MB.
  - Si estoy colocando demasiados datos dentro de un único documento, podría alcanzar este límite.

Estrategia 2:
documentos referenciados

```
{
  "imdbID":"tt0944947",
  "Type":"series",
  "Title":"Game of Thrones",
  "Genre":"Adventure, Drama, Fantasy",
  "Actors":
       ["nm0227759", "nm0372176", "nm3229685" ]
  "imdbRating":"9.5",
  "imdbVotes":"1,031,056"
}
{
  "imdbID":"tt1877832",
  "Type":"movie"
  "Title":"X-Men: Days of Future Past",
  "Year":"2014",
  "Genre":"Action, Adventure, Fantasy",
  "Actors":
       [ "nm0413168" ,"nm1055413","nm2225369","nm0227759"]
  "imdbRating":"8.0",
  "imdbVotes":"514,203"
}
...
{ "id":"nm0227759",
  "name":"Peter Dinklage",
  "description":"Actor, Battle of the Bastards"
}
```

Documentos referenciados

VENTAJAS

- Evito duplicación de datos
- Menor tamaño en los documentos
- La información a la que se accede con poca frecuencia no se retorna en cada consulta.

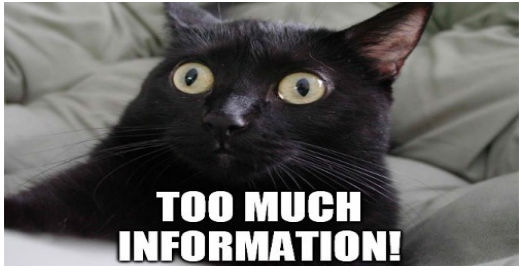
LIMITACIONES

- Para recuperar toda la información relevante debo utilizar $lookup.
- La actualización puede impactar a más de un documento (transacción).
- Aunque el operador de transacción está disponible desde la versión 4.0, es un anti-patrón depender demasiado de su uso.

Aspectos a tener en cuenta en el diseño

- Atomicidad de las operaciones (ACID a nivel de documento)
- Restricción de tamaño máximo de cada documento.
- Restricciones de tamaño y cantidad de documentos en las colecciones
- Operaciones a realizar y su frecuencia
- Ciclo de vida del dato (permanente, volátil?)
- Necesidades de particionamiento (sharding)

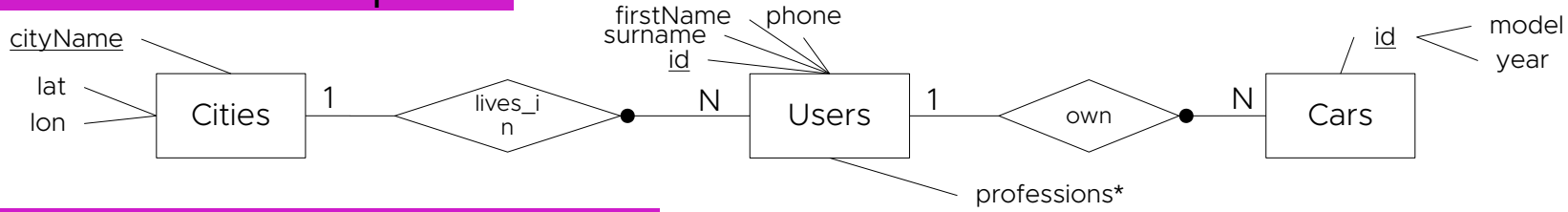
Estos aspectos condicionan los diseños posibles



¿Cómo combinamos todos estos ingredientes?

- Aplicando recomendaciones básicas de diseño
- Considerando patrones de diseño
- Usando metodologías emergentes que buscan optimizar costos

Diseño conceptual



Diseño lógico relacional

Tabla Cities

cityName	loc_x	loc_y
London	51.51 N	0.12 W
Montevideo	34.9 S	56.18 W

Tabla User_professions

userId	profession
1	banking
1	finance
1	trader

Tabla Users

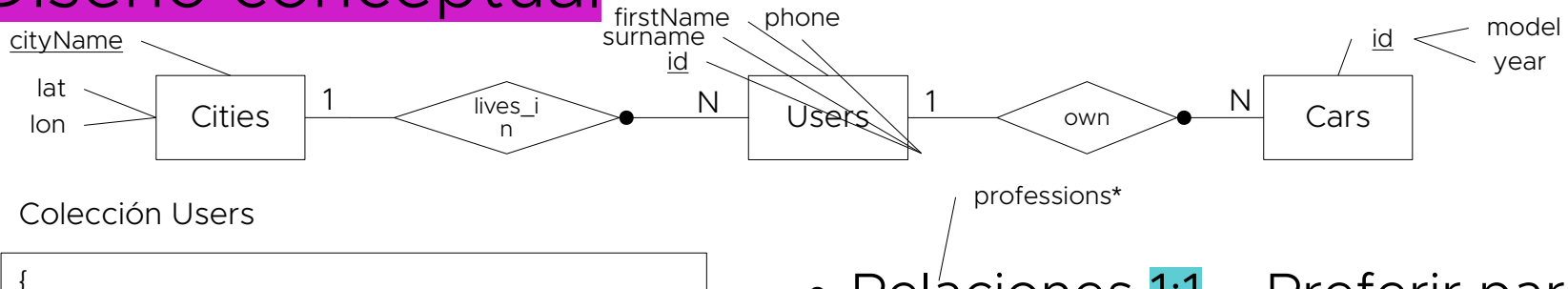
id	firstName	surname	phone	city
1	Paul	Miller	5111111	London

Tabla Cars

userId	model	year
1	Bentley	1973
1	Rolls Royce	1965

Ahora construyamos un diseño lógico documental, asumiendo que las consultas están centradas en usuarios

Diseño conceptual



Colección Users

```
{
   "first_name": "Paul",
   "surname": "Miller",
   "cell": "5111111",
   "city": "London",
   "location": [51.51, 0.12],
   "profession": ["banking", "finance", "trader"],
   "cars": [
      {
         "model": "Bentley",
         "year": 1973
      },
      {
         "model": "Rolls Royce",
         "year": 1965
      }
   ]
}
```

- Relaciones 1:1 – Preferir parejas clave valor en el documento
- Relaciones 1:N
  - 1 a pocos – Preferir documentos embebidos