

# BASE DE DATOS DE DOCUMENTO

**Kelley Nicolás, López Fernanda, Martínez Diego,  
Medina Ashmed, Reyes Yareni**

# INTRODUCCIÓN

- Flexibilidad de NoSQL
- Administrar estructuras de datos más complejas (no key - value)
- No definir estructura común



# CARACTERÍSTICAS SIMILARES A LAS BASES DE DATOS RELACIONALES

- Consultar
- Filtrar



# ¿QUÉ ES UN DOCUMENTO?

Ejemplo de Documento de HTML

Almacenan 2 tipos de información:

- Comandos de contenido
- Comandos de formato

Utilizan:

- Etiquetas predefinidas para formato

## The Structure of HTML Documents

HTML documents combine content, such as text and images, with layout instructions, such as heading and table formatting commands.

### Major Headings Look Like This

Major headings are used to indicate the start of a high level section. Each high level section may be divided into subsections.

### Minor Headings Indicate Subsections

Minor headings are useful when you have a long major section and want to visually break it up into more manageable pieces for the reader.

### Summary

HTML combines structure and content. Other standards for structuring combinations of structure and content include XML and JSON.

# UTILIZANDO JAVASCRIPT OBJECT NOTATION (JSON)

Una muestra de registro de  
cliente es:

```
{  
  "customer_id":187693,  
  "name": "Kiera Brown",  
  "address" : {  
    "street" : "1232 Sandy Blvd.",  
    "city" : "Vancouver",  
    "state" : "Washington",  
    "zip" : "99121"  
  },  
  "first_order" : "01/15/2013",  
  "last_order" : " 06/27/2014"  
}
```

# LA ESTRUCTURA DE LOS OBJETOS JSON :

- Sintaxis simples
- Datos en pares clave-valor
- Documentos en pares nombre-valor (,)
- Principio y fin con { }
- Nombres son cadenas
- Valores pueden ser varios
- Valores de matrices [ ]
- Valores de objetos en pares {clave-valor} entre llaves

# EN XML :

Una muestra de registro de cliente es:

```
<customer_record>
<customer_id>187693</customer_id>
  <name>"Kiera Brown"</name>
  <address>
    <street>"1232 Sandy Blvd."</street>
    <city>"Vancouver"</city>
    <state>"Washington"</state>
    <zip>"99121"</zip>
  </address>
  <first_order>"01/15/2013"</first_order>
  <last_order>"06/27/2014"</last_order>
</customer_record>
```



## Ventajas:

- Formato muy estructurado y fácil
- Válido mediante Schemas(XSD)
- Se pueden definir estructuras completas y reutilizables

## Desventajas:

- Formato estricto
- Más tiempo de procesamiento
- Un error = documento inválido



## Ventajas:

- ★ Formato simple
- ★ Velocidad de procesamiento alta
- ★ Archivos de menor tamaño

## Desventajas:

- ❑ Estructura enredosa
- ❑ No fácil de interpretar a simple vista





# DOCUMENTOS Y PARES CLAVE - VALOR

- Atributos relacionados se administran dentro de un mismo objeto.
- Imitar algunos aspectos (atributos, entidades, identificador único)
- Documentos (así como las tablas relacionales) tienen muchos atributos dentro en un solo objeto.

Last Purchase Date > (Today () - 180)

## Key Value

Customer\_List = Return Customer\_ID  
Where Last Purchase Date >  
(Today () - 180);

Customer\_Name = Return Customer Name  
Value Where  
Customer\_ID in  
Customer\_List;

Customer\_Address = Return Customer Address  
Value Where  
Customer\_ID in  
Customer\_List;

Bases de datos de documentos - código para consultar varios atributos

# GESTIONAR MÚLTIPLES DOCUMENTOS EN COLECCIONES

Documentos generalmente agrupados en colecciones de documentos similares.

Parte Clave: Decidir cómo organizar mis documentos en colecciones.



# EMPEZANDO CON LAS COLECCIONES

- ❑ “Listas de documento”
- ❑ Diseñadores de bases de datos de documentos optimizan la base de datos de documentos para agregar, eliminar, actualizar y buscar rápidamente
- ❑ Escalabilidad
- ❑ Documentos de la misma colección no mismas estructuras, pero deben compartir alguna estructura común

```
{  
  {  
    "customer_id":187693,  
    "name": "Kiera Brown"  
    "address" : {  
      "street" : "1232 Sandy Blvd.",  
      "city" : "Vancouver",  
      "state" : "WA",  
      "zip" : "99121"
```

<http://freepdf-books.com>

```
    "first_order" : "01/15/2013",  
    "last_order" : " 06/27/2014"  
  }  
}
```

DOCUMENTOS CON ESTRUCTURA SIMILAR

```
{  
  "customer_id":187694,  
  "name": "Bob Brown",  
  "address" : {  
    "street" : "1232 Sandy Blvd.",  
    "city" : "Vancouver",  
    "state" : "WA",  
    "zip" : "99121"  
  },  
  "first_order" : "02/25/2013",  
  "last_order" : " 05/12/2014"  
}
```

DOCUMENTOS CON ESTRUCTURA SIMILAR

```
{
  "customer_id":179336,
  "name": "Hui Li",
  "address" : {
    "street" : "4904 Main St.",
    "city" : "St Louis",
    "state" : "MO",
    "zip" : "99121"
  },
  "first_order" : "05/29/2012",
  "last_order" : " 08/31/2014",
  "loyalty_level" : "Gold",
  "large_purchase_discount" : 0.05,
  "large_purchase_amount" : 250.00
}
```

DOCUMENTOS CON ESTRUCTURA SIMILAR

```
{  
  "customer_id":290981,  
  "name": "Lucas Lambert",  
  "address" : {  
    "street" : "974 Circle Dr.",  
    "city" : "Boston",  
    "state" : "MA",  
    "zip" : "02150"  
  },  
  "first_order" : "02/14/2014",  
  "last_order" : " 02/14/2014",  
  "number_of_orders" : 1,  
  "number_of_returns" : 1  
}
```

DOCUMENTOS CON ESTRUCTURA SIMILAR

# TIPS PARA DISEÑAR COLECCIONES



- ★ Son conjuntos de documentos
- ★ No imponen coherencia
- ★ En documentos de la colección se puede guardar diferentes tipos de documentos en una sola colección
- ★ Deben guardar documentos del mismo tipo de entidad



## EVITAR LOS TIPOS DE ENTIDADES ALTAMENTE ABSTRACTOS

- event\_type muy abstracta porque datos de flujo de click web con registro del servidor no tienen casi nada en común.
- Solo comparten ID y datetime.

```
{ "id" : 12334578,  
  "datetime" : "201409182210",  
  "session_num" : 987943,  
  "client_IP_addr" : "192.168.10.10",  
  "user_agent" : "Mozilla / 5.0",  
  "referring_page" : "http://www.example.com/page1"  
}  
  
{ "id" : 31244578,  
  "datetime" : "201409172140",  
  "event_type" : "add_user",  
  "server_IP_addr" : "192.168.11.11",  
  "descr" : "User jones added with sudo privileges"  
}
```

Si almacenamos estos dos en una misma colección, habría que agregar un indicador de tipo para hacer que el código pueda distinguir fácilmente entre un documento de eventos de registro del servidor y uno de flujo de clics web.

# INDICADOR DE TIPO AGREGADO:

- doc\_type

No mezclar el tipo de entidades entre el documento de eventos de registro del servidor y el documento de flujos de clics web

```
{ "id" : 12334578,  
  "datetime" : "201409182210",  
  "doc_type": "click_stream",
```

<http://freepdf-books.com>

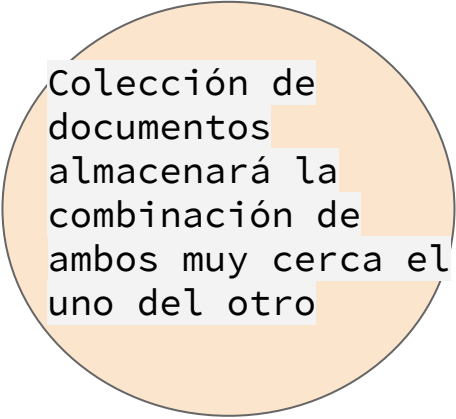
```
  "client_IP_addr" : "192.168.10.10",  
  "user_agent" : "Mozilla / 5.0",  
  "referring_page" : "http://www.example.com/page1"  
}
```

```
{ "id" : 31244578,  
  "datetime" : "201409172140"  
  "doc_type" : "server_log"  
  "event_type" : "add_user"  
  "server_IP_addr" : "192.168.11.11"  
  "descr" : "User jones added with sudo privileges"  
}
```

Filtrar colecciones muchas veces es más lento que trabajar directamente con varias colecciones.

650,000 documentos clickstream

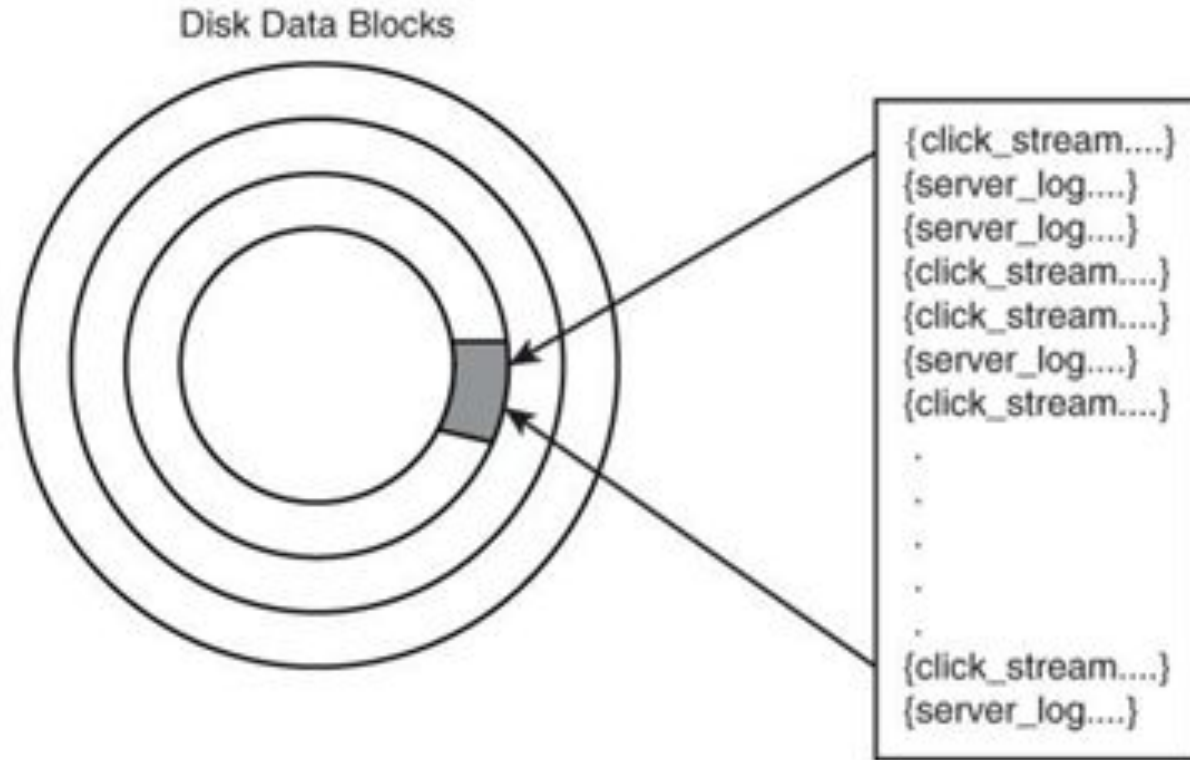
350,000 eventos de registro del servidor



Colección de documentos almacenará la combinación de ambos muy cerca el uno del otro

The diagram consists of two light orange rounded rectangular boxes on the left, one containing '650,000 documentos clickstream' and the other '350,000 eventos de registro del servidor'. A large right-facing curly bracket groups these two boxes. To the right of the bracket is a larger light orange circle. Inside the circle, text explains that the collection will store the combination of both, very close to each other.

Si se está usando el almacenamiento en disco, puede haber recuperación de datos de ambos.

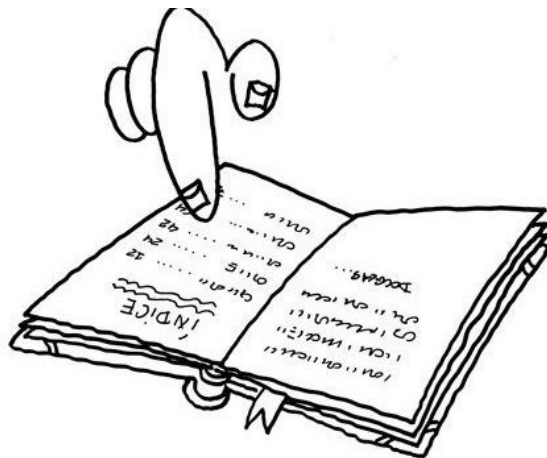


MEZCLA DE TIPOS DE DOCUMENTOS LLEVA A MUCHOS TIPOS DE DOCUMENTOS EN 1 BLOQUE DE DATOS DE DISCO PROVOCANDO INEFICIENCIA PORQUE DATOS SE LEEN DE DISCO PERO APLICACIÓN QUE FILTRA LOS DOCUMENTOS NO LOS USA SEGÚN EL TIPO QUE SON.

# CARDINALIDAD DE EN TÉRMINOS DE BASES DE DATOS RELACIONALES

# RELACIONALES

- ❖ Depende del tamaño de la colección, el índice y el número de tipos de documentos.
- ❖ Considerar el uso de índice.



# FUNCIONES SEPARADAS PARA MANIPULAR DIFERENTES TIPOS DE DOCUMENTOS

- Para saber si una colección debe dividirse en varias colecciones checa tu código de aplicación.
- Código de aplicación que manipula una colección debe tener código aplicado a todos los documentos y otra cierta cantidad de código especializado a uno en particular.
- Ejemplo...



```
If (doc_type = 'click_stream'):  
    process_click_stream (doc)  
Else  
    process_server_log (doc)
```

### Lower-Level Branching

```
book.title = doc.title  
book.author = doc.author  
book.year = doc.publication_year  
book.publisher = doc.publisher  
book.descr = book.title + book.author + book.year + book.publisher  
if (doc.ebook = true);  
    book.descr = book.descr + doc.ebook_size
```

SE DEBEN CREAR COLECCIONES SEPARADAS SI HAY RAMIFICACIONES DE ALTO NIVEL EN LAS FUNCIONES QUE MANIPULAN A LOS DOCUMENTOS, BIFURCACIÓN EN NIVELES INFERIORES ES COMÚN CUANDO HAY DOCUMENTOS CON ATRIBUTOS OPCIONALES.

# UTILIZAR SUBTIPOS DE DOCUMENTOS CUANDO LAS ENTIDADES SE AGREGUEN O COMPARTAN CON FRECUENCIA CÓDIGO SUSTANCIAL

1. ¿Cómo decidir cómo organizar estos datos en uno o más colección de documentos?
2. Empezar por cómo se utilizarán los datos
3. Preguntar a cliente lo siguiente:

- Product name

<http://amazon.com>

- SKU (stock keeping unit)
- Product dimensions
- Shipping weight
- Average customer review score
- Standard price to customer
- Cost of product from supplier

Each of the product types will have specific fields. Books will have fields with information about

- Author name
- Publisher
- Year of publication
- Page count

The CDs will have the following fields:

- Artist name
- Producer name
- Number of tracks
- Total playing time

The small kitchen appliances will have the following fields:

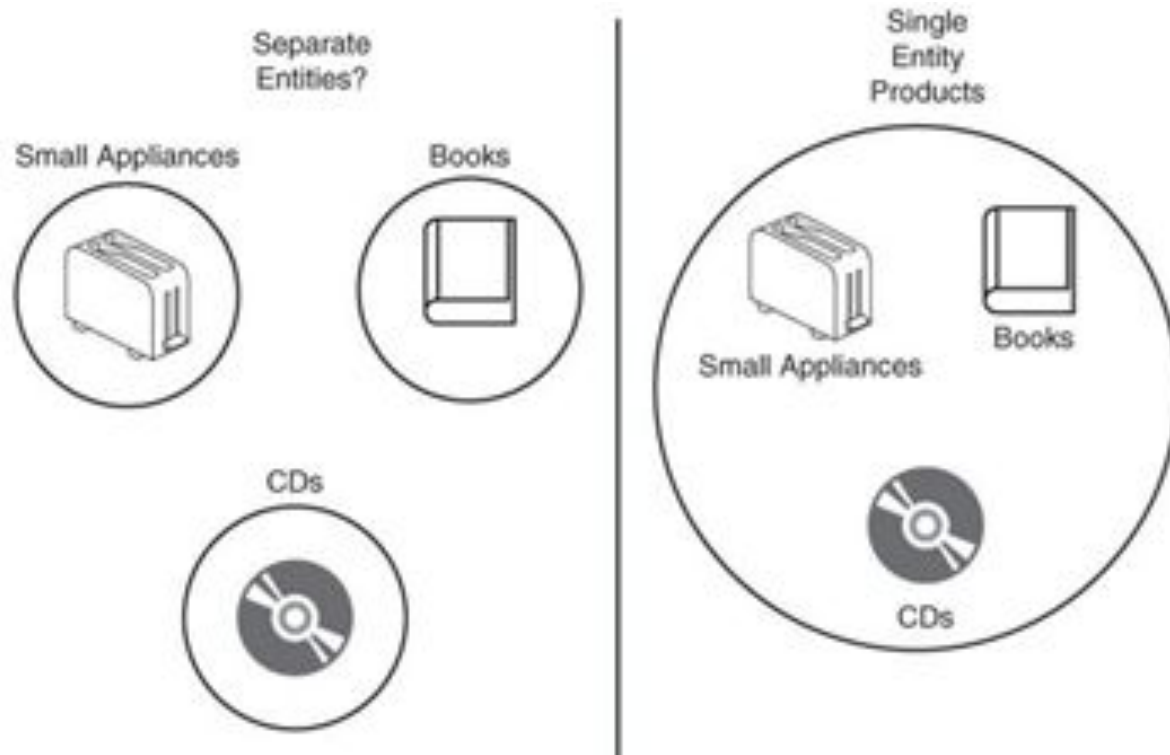
- Color
- Voltage
- Style



# CONSULTA/ PREGUNTAS AL CLIENTE:



- ★ ¿Cuál es el # promedio de productos comprados por cada cliente?
- ★ ¿Cuál es el mayor # de productos comprados?
- ★ ¿Cuáles son los 20 productos más populares (por estado) del cliente?
- ★ ¿Cuál es el precio estándar a cliente-costo del producto del proveedor?
- ★ ¿Cuántos de cada tipo de producto se vendieron en los últimos 30 días?



LAS CONSULTAS AYUDAN A GUIAR LA ORGANIZACIÓN DE DOCUMENTOS Y COLECCIONES

# ¡PARA RESUMIR! (PARTE 1)

1. Evitar tipos de documentos demasiado abstractos
2. Para procesar diferentes subtipos de documentos, considera separar los tipos de documentos en diferentes colecciones
3. Un diseño de colección malo puede afectar negativamente el rendimiento
4. A veces puedes agrupar algo diferente (pequeños electrodomésticos y libros) si son tratados como iguales (ambos son productos)

## ¡PARA RESUMIR! (PARTE 2)

5. Define tus estructuras y tipos de registro antes de usarlo en tu programa
6. Ajusta los modelos de datos que definen tablas, columnas y otras estructuras de datos



# OPERACIONES BÁSICAS EN BASE DE DATOS DE DOCUMENTO

- Inserción
- Eliminar
- Actualización
- Recuperación



# INSERTAR DOCUMENTOS EN UNA COLECCIÓN

- Serie de operaciones diferentes
- Agregar documentos a una colección
- Ejemplo:

```
db.books.insert( {“title”:” Mother Night”, “author”: “Kurt Vonnegut, Jr.”} )
```

- Ejemplo :

```
db.books.insert( {book_id: 1298747, “title”:“Mother Night”, “author”: “Kurt Vonnegut, Jr.”} )
```

# INSERTAR DOCUMENTOS EN UNA COLECCIÓN

```
db.books.insert(  
  [  
    {"book_id": 1298747,  
     "title": "Mother Night",  
     "author": "Kurt Vonnegut, Jr."},  
    {"book_id": 639397,  
     "title": "Science and the Modern World",  
     "author": "Alfred North Whitehead"},  
    {"book_id": 1456701,  
     "title": "Foundation and Empire",  
     "author": "Isaac Asimov"}  
  ]  
)
```

- Inserciones masivas

- Ejemplo :

```
db.books.insert( {"book_id": 1298747, "title": "Mother  
Night", "author": "Kurt Vonnegut, Jr."} )
```

```
db.books.insert( {"book_id": 639397, "title": "Science and  
the Modern World", "author": "Alfred North Whitehead"} )
```

```
db.books.insert( {"book_id": 1456701, "title": "Foundation and  
Empire", "autor": "Isaac Asimov"} )
```

- Útil para cargar un número de documentos a la vez

# ELIMINAR DOCUMENTOS DE UNA COLECCIÓN

- `db.books.remove ()`
- Colección sigue existiendo
- Lista de claves y valores
- `{"book_id": 639397}`
- Para eliminar :  
`db.books.remove({"book_id": 639397})`
- Tener cuidado al eliminar  
`db.books.remove({"author": "Kurt Vonnegut, Jr."})`

```
db.books.insert(  
  [  
    {"book_id": 1298747,  
      "title": "Mother Night",  
      "author": "Kurt Vonnegut, Jr."},  
  
    {"book_id": 1298770,  
      "title": "Cat's Cradle",  
      "author": "Kurt Vonnegut, Jr."},  
  
    {"book_id": 639397,  
      "title": "Science and the Modern  
        World",  
      "author": "Alfred North  
        Whitehead"},  
  
    {"book_id": 1456701,  
      "title": "Foundation and Empire",  
      "author": "Isaac Asimov"}  
  ]  
)
```



# ACTUALIZACIÓN DE DOCUMENTOS EN UNA COLECCIÓN

- 2 parámetros para actualizar:

Consulta de documentos

Conjunto de claves y valores a actualizar

- Actualizar a Kurt : `{"book_id": 1298747}`
- Operador `$set`
- Ejemplo: `db.books.update ({"book_id": 1298747}, {$set {"quantity" : 10 }})`
- Agrega clave si no existe

# RECUPERAR DOCUMENTOS DE UNA COLECCIÓN

- Método de búsqueda
- `db.books.find()`
- Especificar criterios
- Ejemplo: `db.books.find({"author": "Kurt Vonnegut, Jr."})`

# RECUPERAR DOCUMENTOS DE UNA COLECCIÓN

- Condicionales y operadores booleanos

ejemplo: `Db.books.find( { "quantity" : { "$gte" : 10, "$lt" : 50 } } )`

- Las condicionales y booleanos admitidos en MongoDB son:

- `$lt`—Menos que
- `$lte`—Menos que o igual a
- `$gt`—Mayor que
- `$gte`—Mayor que o igual a
- `$in`—Consulta de valores de una sola llave
- `$or`—Consulta de valores en claves múltiples
- `$not`—Negación

# AUSENCIA DE ESQUEMA (SCHEMALESS)

- Estructura de documentos no tiene estructura específica
- Más Flexibilidad y más responsabilidad
- Se puede agregar nuevos valor de llave en cualquier momento
- Estructuras ∈ {  
colección

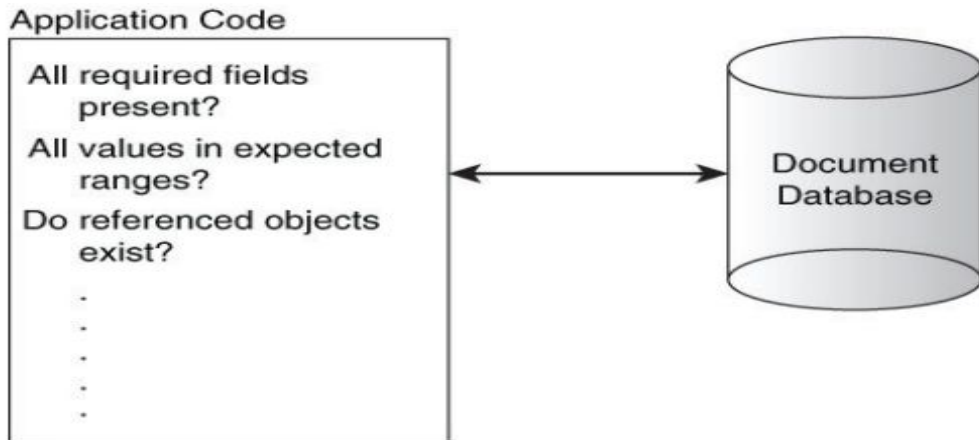
Ejemplo

flexibilidad: and {  
    'employeeName' : 'Robert Lucas',  
    'department' : 'Finance'  
    'startDate' : '21-May-2009',  
    'certifications' : 'CPA'  
}

```
{ 'employeeName' : 'Janice Collins',  
  'department' : 'Software engineering',  
  'startDate' : '10-Feb-2010',  
  'pastProjectCodes' : [ 189847, 187731, 176533, 154812]  
}
```

# AUSENCIA DE ESQUEMA (SCHEMALESS)

- Más libertad en la estructura de documentos pero el SDAD no aplica reglas basadas en la estructura
- Código de aplicación = Reglas sobre los datos
- Sobre el tiempo la llaves y información puede cambiar
- Código de validación de datos y manejo de errores



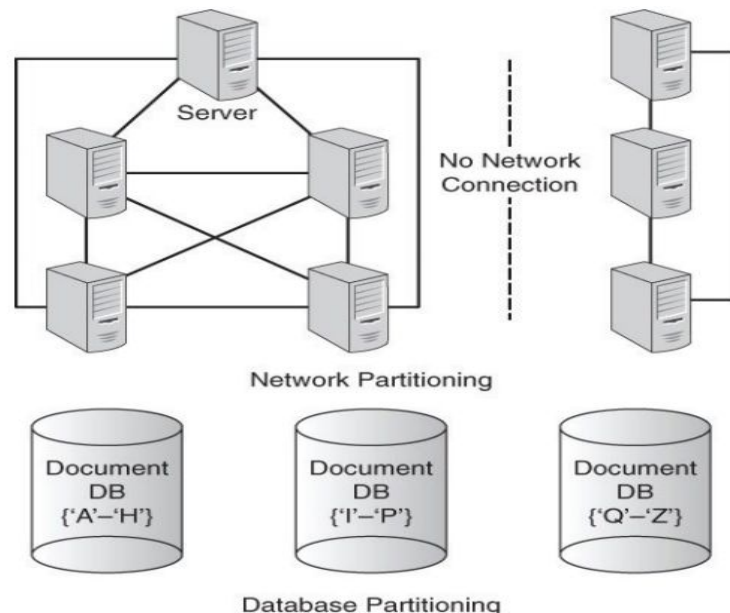
# ESQUEMA POLIMORFO

- El documento en la colección tiene muchas formas
- Muchas estructuras de documentos en un conjunto de documentos creando una colección

```
{  
  {'a': 1,  
   'b': 2,  
   'c': 3  
  }  
  
  {'a': 7,  
   'b': 8,  
   'd': 10  
  }  
  
  {'a': 20,  
   'e': 30,  
   'f': 40,  
   'g': 50  
  }  
}
```

# ¿QUE ES PARTICIÓN?

- Partición es la separación en partes de una red que no son accesibles entre sí.
- División de una bd de documentos y distribución a diferentes servidores.



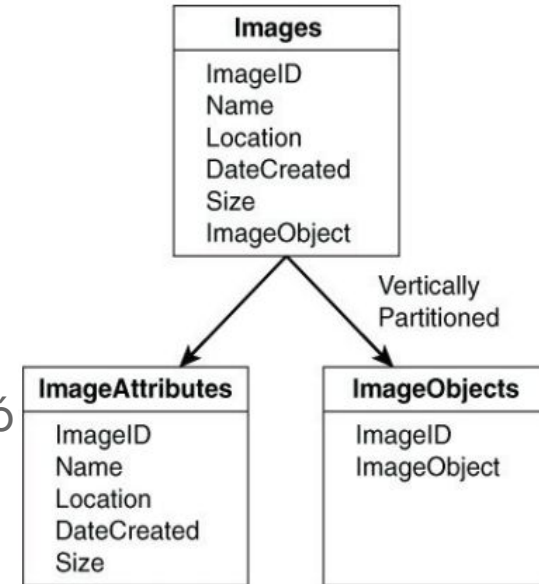
# PARTICIÓN VERTICAL

- Separar las columnas de una tabla relacional en múltiples tablas separadas

Ejemplo: Buscar imagen por características

Si no se realiza la separación, la imagen se forzará a aparecer al mismo tiempo que se ingresa la información

- Separar es mas eficiente en la recuperación de datos

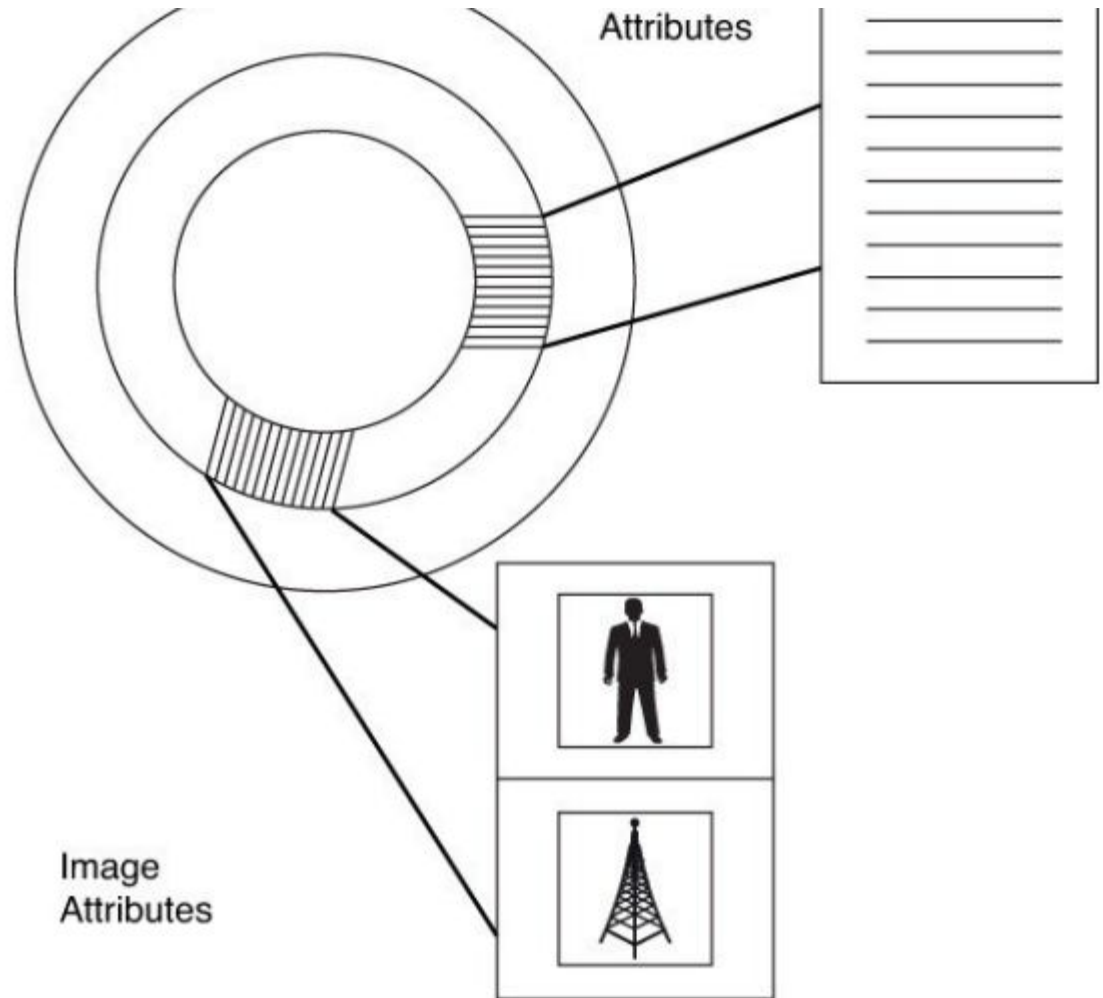




# APP IMAGEN

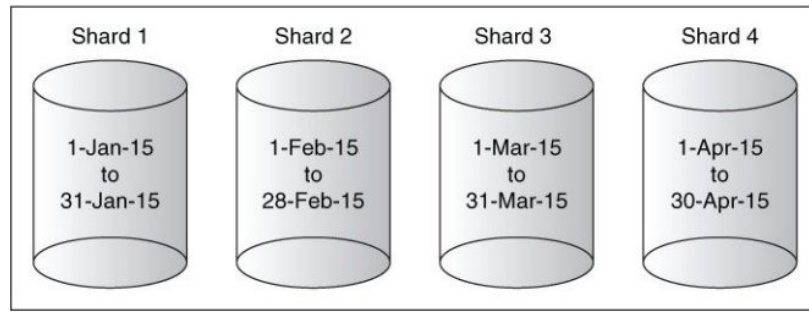
El objeto imagen no se recupera junto con los datos de atributos

- Más usado en bd relacional que en bd de documentos
- Métodos para usarlo en bd no relacionales



# PARTICIÓN HORIZONTAL

- Se divide por documento en una bd de documentos o en filas en una bd relacional (fragmentación)
- Estos fragmentos se almacenan en servidores separados
- Un fragmento único puede almacenarse en varios servidores si una bd o cluster está configurada para replicar datos
- Si se replica o no los datos, un servidor en un grupo de bd de documentos tendrá un solo fragmento por servidor
- Se implementa con una shard key y un método de partición
- Ahorra dinero y tiempo



# SEPARANDO DATOS CON SHARD KEYS

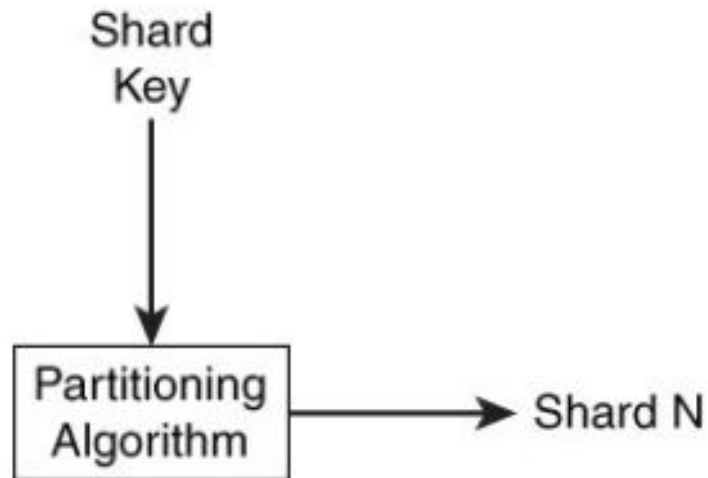
- Shard key es una o más llaves o campos que existe en todos los documentos de una colección que es usada para separar documentos.

Ejemplo:

- ID (Del documento único)
- Nombre
- Fecha (Creación del documento)
- Categoría o Tipo

# SEPARANDO DATOS CON SHARD KEYS

- Shard key especifica en que particion de las existentes en la tabla se colocaran los datos que se inserten
- El algoritmo de partición:
  1. Shard key como entrada
  2. Fragmento apropiado para la llave



# DISTRIBUCIÓN DE DATOS CON ALGORITMO DE PARTICIÓN

Diferentes funciones para hacer partición horizontal:

- Range: shard key se determina por un rango de valores, que determina la partición donde se almacenará un valor
- Hash: shard key es una función hash, aplicada sobre una columna, distribución equitativa de los registros sobre las diferentes particiones
- List: shard key es una lista de valores para determinar a cada una de las particiones

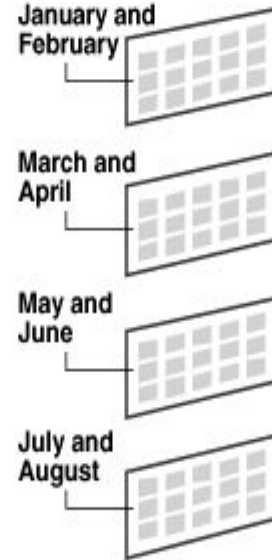
# DISTRIBUCIÓN DE DATOS CON ALGORITMO DE PARTICIÓN

- Electronic—northeast
- Electronics—southeast
- Electronics—midwest
- Electronics—southwest
- Electronics—northwest
- Appliances—northeast
- Appliances—southeast
- Appliances—midwest
- And so forth...

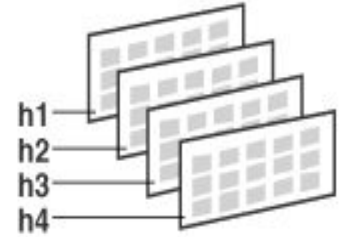
List  
Partitioning



Range  
Partitioning



Hash  
Partitioning



Tipos de Particionado en Oracle

# DISTRIBUCIÓN DE DATOS CON ALGORITMO DE PARTICIÓN

- La fragmentación es un proceso fundamental para grandes bd de documentos y un mejor rendimiento
- La partición vertical es posible en base de datos de documentos
- La partición horizontal es ampliamente usada en la base de datos de documentos
- Los desarrolladores pueden elegir la shared key para la fragmentacion
-

# MODELADO DE DATOS Y PROCESAMIENTO DE CONSULTAS

## Características:

Flexibles

Documentos y variaciones en misma colección

## Requisitos:

Lista de consultas en BD





# MODELADO DE DATOS Y PROCESAMIENTO DE CONSULTAS

BD relacional

Entidades, relaciones, atributos,  
etc

BD de documento

Lista de consultas



Normalización

Problemas

Desnormalización

Mala integración de consultas

Salida del procesador



# MODELADO DE DATOS Y PROCESAMIENTO DE CONSULTAS

## BD de documento VS. BD relacionales

Procesos menos formales

Implementan procesadores de consultas

→ Secuencia óptima de pasos

→ Recuperar datos especificados



# NORMALIZACIÓN

Objetivo:

Organizar datos para evitar anomalías y/o reducirlas.

¿Cómo?      Añade datos repetidos a misma tabla.

Atributos adicionales se relacionan.

Anomalía: inconsistencia de datos.

# NORMALIZACIÓN

Diseño de documentos.

Normalizado:

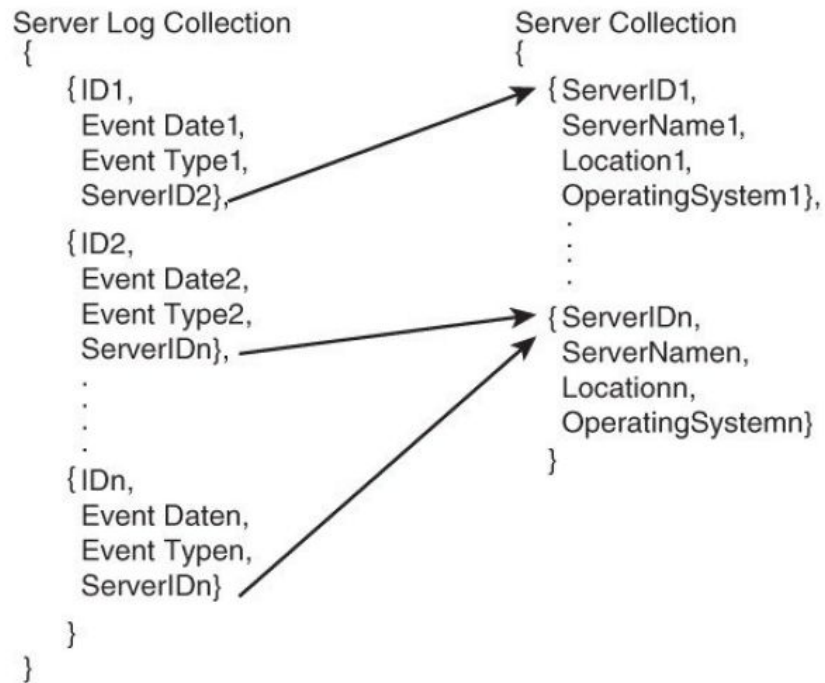
Varias colecciones  
que almacenan datos.

- ❏ Referencias a otros documentos
  - ❏ Buscar información adicional

# EJEMPLO

Un documento registra el servidor.

Una colección puede contener información adicional del servidor para no repetir en documento.



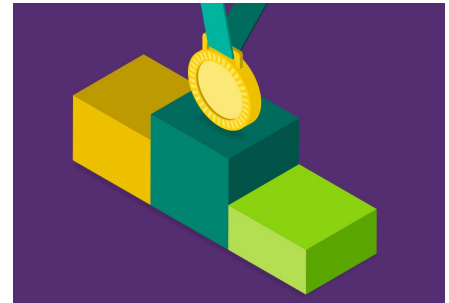
# DENORMALIZACIÓN

Normalización = problemas de rendimiento.

→ Información en varias tablas.

¿Formas eficientes de unir información?

Mejorar rendimiento de unión



# DENORMALIZACIÓN

BD altamente normalizada.



Bajo rendimiento



Desnormalización.



Mejora rendimiento.

\*Datos obtenidos de colección o doc.

# PROCESADOR DE CONSULTAS

¿Cómo obtener información?

A partir de un dato específico.



¿Cómo funciona?

Consultas de entrada



Datos



Secuencia para recuperar datos.



# PROCESADOR DE CONSULTAS

Key-value vs. BD de Documento

1. No necesita procesadores de consulta
2. Valores con clave.



# PROCESADOR DE CONSULTAS

¿Varias condiciones?

Jerarquía de criterios



Ejemplo

Fecha de creación vs. tipo de documento

→ Regresa el de menor cantidad de docs.

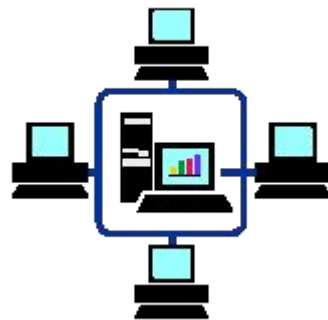
# PROCESADOR DE CONSULTAS

Comparten lenguaje con BD NoSQL

Fragmentación:

Divide BD grandes en varios servidores

→ Mejora rendimiento

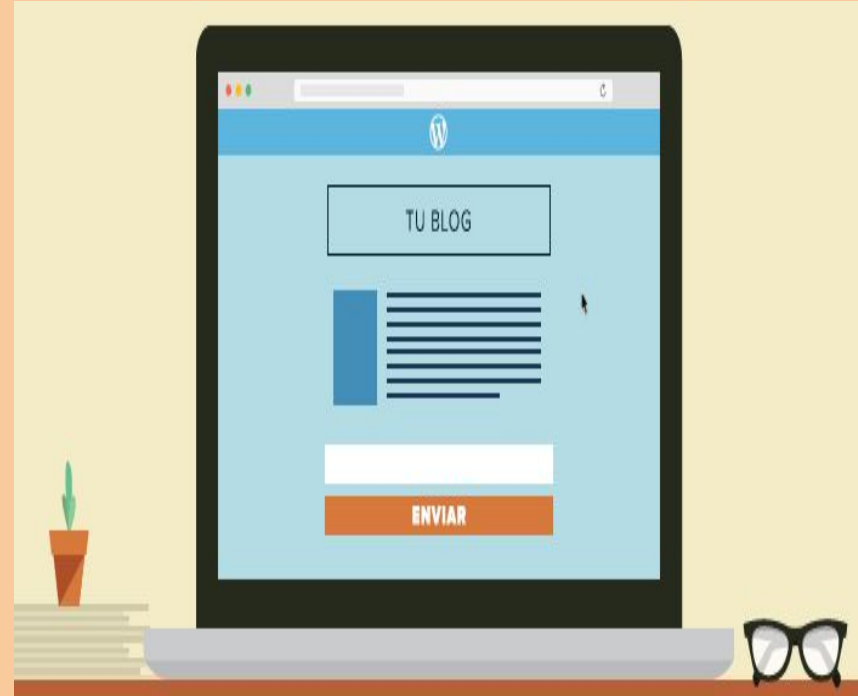


# DISEÑO DE BD DE DOCUMENTO

Cuenta con estructura flexible : JSON y XML

Factor clave: Flexibilidad.

Objetivo: Elegir el mejor modelo de BD de documento dependiendo necesidades.



# BALANCE ENTRE NORMALIZACIÓN Y DESNORMALIZACIÓN

Relación 1:M

Ejemplo: 1 cliente y varios pedidos.

Relación M:M

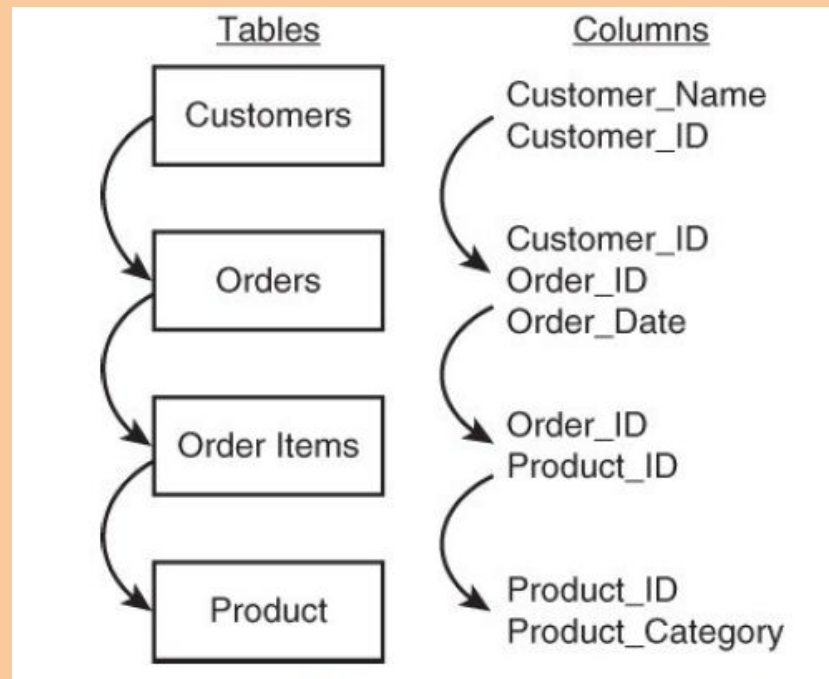
Ejemplo: Muchos clientes y muchas promociones.

→ Se busca almacenar datos relacionados en el mismo doc.

# EJECUCIÓN DE UNIONES

Ejemplo:

Clientes que  
compraron accesorios  
electrónicos en los  
últimos 12 meses.



¿Como mejorar rendimiento?

Índices por año  
Índices por artículo  
Ordenar tablas

# NOSOTROS COMO MODELADORES DE BASES DE DATOS DE DOCUMENTO

- A diferencia de los modeladores de bases de datos relacionales, es más importante prevenir la anomalía de datos, incluso si esto implica sacrificar flexibilidad o escalabilidad.

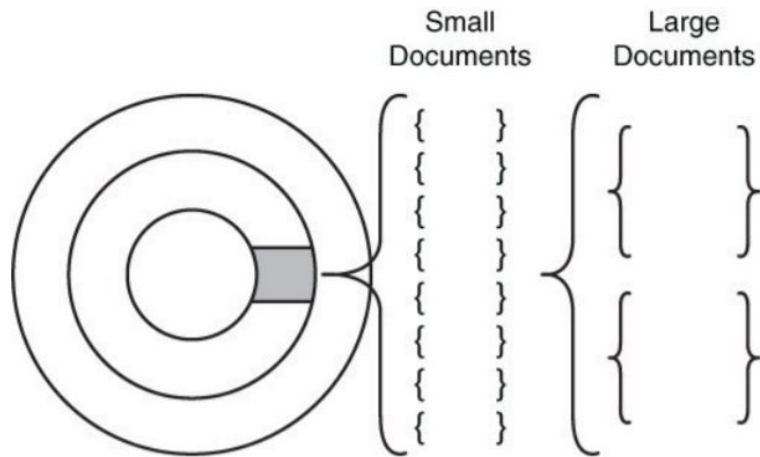
# ¿QUÉ HACER PARA PREVENIR LA ANOMALÍA DE DATOS?

- Regular el uso excesivo de “joins” en los documentos.
- Planear para documentos cambiantes.
- Evitar estar cambiando a los archivos muy pesados.



# NO USAR DEMASIADOS "JOINS" ENTRE LOS DOCUMENTOS.

- **Joins:** Forma en la que un documento se relaciona con otro.



Documentos largos pueden recuperar documentos más pequeños para trabajar con ellos.

# JOIN ENTRE DOS DOCUMENTOS

Número de documentos

```
for doc1 in collection1:  
    for doc2 in collection2:  
        <do something with both documents>
```

Collection 1
10 docs.

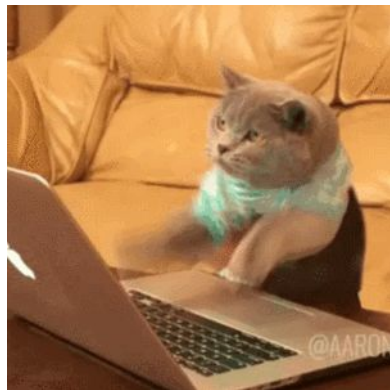
Collection 2
10 docs.

=

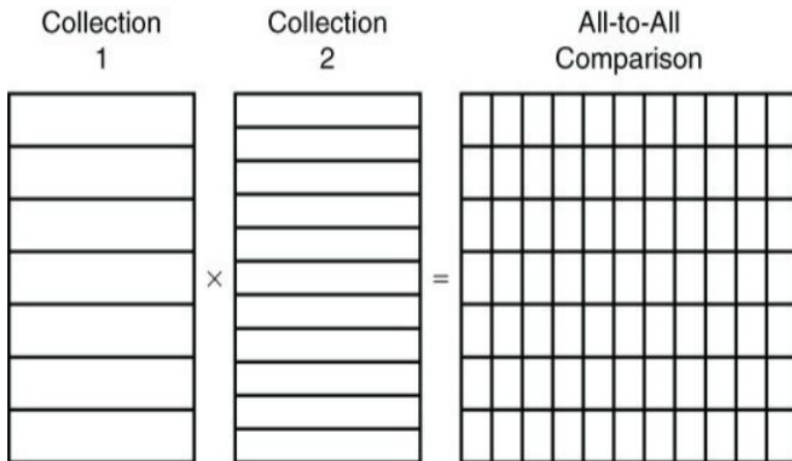
100 veces.

# ¿QUÉ TIENE DE MALO USAR JOINS ENTRE DOCUMENTOS?

Supongamos que la empresa comienza a crecer...

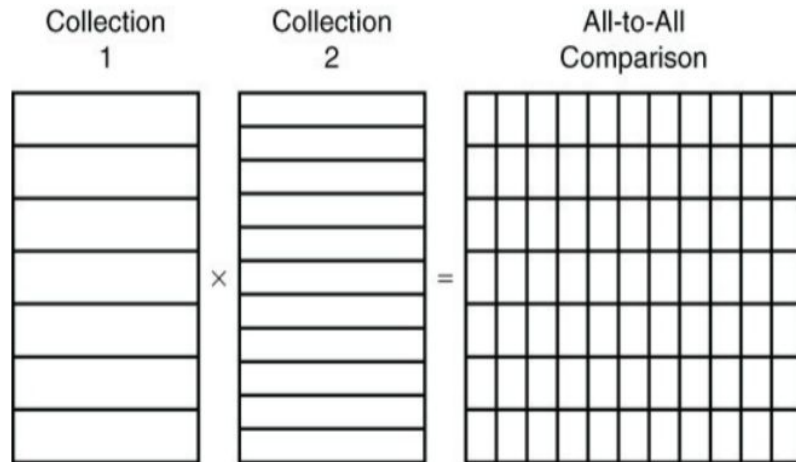


Collection 1	Collection 2	
100,000 docs.	500,000 docs.	= $(5 \times 10^{10})$ veces.

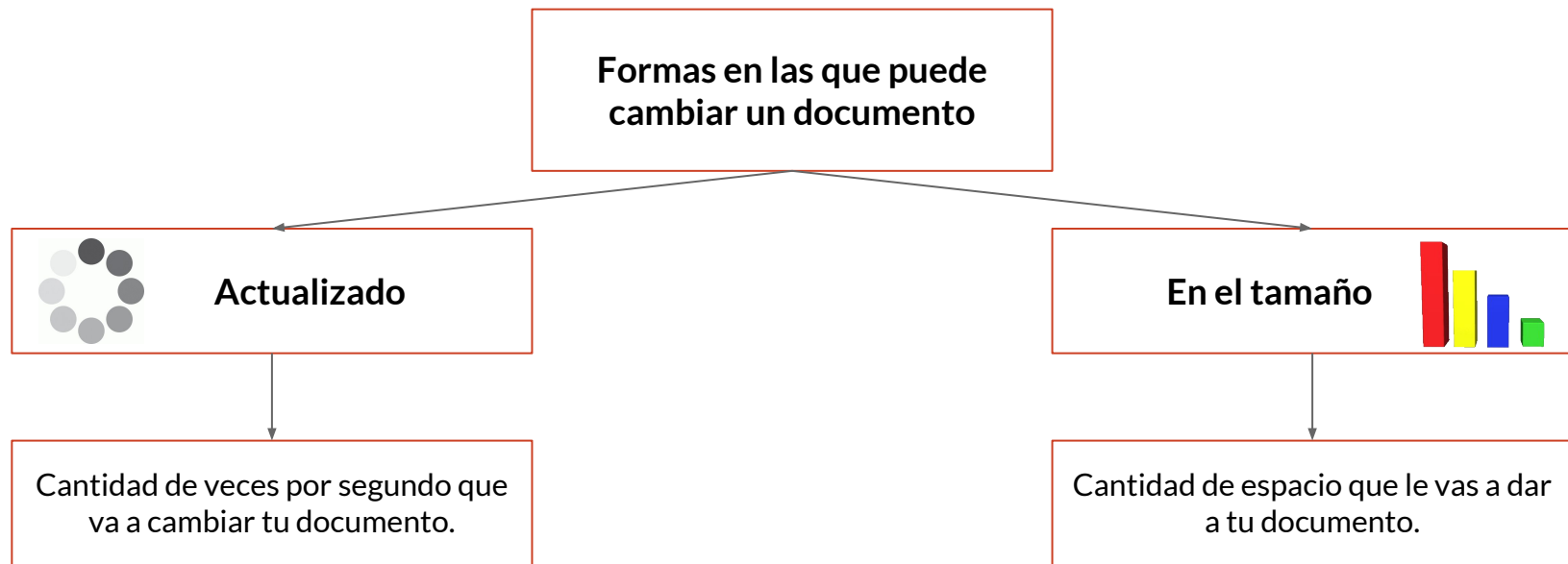


# ¿CÓMO ELIMINAR LOS JOINS?

- Índices.
- Filtrado de información.
- Clasificaciones.



# PLANEAR PARA DOCUMENTOS CAMBIANTES



# EVITAR CAMBIAR ARCHIVOS MUY PESADOS

- Considerar el ciclo de vida de un documento
- Planificar siempre para un crecimiento anticipado.
- Crear un documento con el espacio suficiente para la vida completa de este.

Vendedores = array(15)

Compradores = array(15)

Utilería = array(20)

==>

Vendedores = array(20)

Compradores = array(20)

Utilería = array(20)

# BIBLIOGRAFÍA

- Dan Sullivan. (2015). Document Databases. En NoSQL for mere mortals(630). UEA: Pearson.