

Bases de Datos II  
Bases de Datos NoSQL / Práctica con MongoDB.

**Grupo 1**

**Integrantes**

- Ladislao Bordon
- Lautaro Joaquin Gutierrez
- Facundo Feliú

**Ayudante**

- Natalia Ortu

**Parte I**

**Sección 1: Bases de Datos NoSQL y Relacionales**

Si bien las BBDD NoSQL tienen diferencias fundamentales con los sistemas de BBDD Relacionales o RDBMS, algunos conceptos comunes se pueden relacionar. Responda las siguientes preguntas, considerando MongoDB en particular como Base de Datos NoSQL.

1. ¿Cuáles de los siguientes conceptos de RDBMS existen en MongoDB? En caso de no existir, ¿hay alguna alternativa? ¿Cuál es?

- Base de Datos
- Tabla / Relación
- Fila / Tupla
- Columna

- Base de Datos: este concepto existe también en MongoDB.

- Tabla / Relación: este concepto no existe en MongoDB. La alternativa son las 'Colecciones' y la diferencia con las 'Tablas' es que son grupos de documentos en lugar de grupos de registros.

- Fila / Tupla: este concepto no existe en MongoDB. La alternativa es 'Documento'.

- Columna: este concepto no existe en MongoDB. La alternativa son los 'Campos' que son pares clave/valor

2. Existen claves foráneas en MongoDB? ¿Qué diferencias existen con las bases de datos de tipo relacional?

El concepto de clave foránea no existe en MongoDB, pero se puede lograr referenciar otros objetos de manera manual mediante la adición de un campo que contenga un objectId referenciando a otro objeto. O sea, el valor del campo "\_id" del otro objeto. A esto se lo llama referencia. Otra forma es incorporar toda la información que se relacione en un único documento, aunque causaría repetición de la información, a esto se lo llaman documentos embebidos

3. Para acelerar las consultas, MongoDB tiene soporte para índices. ¿Qué tipos de índices soporta?

- Single Field: se aplican a un solo campo de una colección. Para declararlo se usa un comando similar a este: `db.users.ensureIndex( { "user_id" : -1 } )`. El número indica que queremos que el índice se ordene de forma descendente. Si quisiéramos un orden ascendente, el parámetro debe ser un 1.

- Compound index: el índice se generará sobre varios campos. `db.users.ensureIndex( { "user_name" : 1, "age":-1 } )` El índice que se generará con la instrucción anterior, agrupará los datos primero por el campo `user_name` y luego por el campo `age`. Es decir, se generaría algo así:

"Antonio", 35

"Antonio", 18

"María", 56

"María", 30

"María", 21

"Pedro", 19

"Unai", 34

"Unai", 27

Podemos usar los índices compuestos para consultar uno o varios de los campos, sin ser necesario incluirlos todos. En el ejemplo anterior el índice se puede utilizar siempre que se hagan consultas sobre `user_name` o sobre `user_name` y `age`. Este índice compuesto ordenará el campo `user_name` de la misma manera que si creamos un índice simple. Lo que no podemos hacer es buscar sólo sobre el campo "age". Para ese caso tendríamos que crear un índice específico. Si el índice compuesto tuviera tres campos, podríamos utilizarlo al consultar sobre el primer campo, sobre el primer y segundo campo, o sobre los tres campos.

- Multikey index: se aplican a campos que tienen información contenida en arreglos. se crea una entrada para cada uno de los elementos del arreglo.

- Geospatial index: La indexación geoespacial de MongoDB permite ejecutar de manera eficiente consultas espaciales en una colección que contiene formas y puntos geoespaciales.

- Text index: MongoDB proporciona índices de texto para admitir consultas de búsqueda de texto sobre el contenido de un string. Los índices de texto pueden incluir cualquier campo

cuyo valor sea un string o una matriz de strings. Una colección solo puede tener un índice de búsqueda de texto, pero ese índice puede cubrir varios campos.

- Hashed index: Los índices hash mantienen entradas con hash de los valores del campo indexado. Los índices hash admiten la fragmentación mediante claves de fragmentación hash. La fragmentación basada en hash utiliza un índice hash de un campo como clave de partición para dividir los datos en su clúster fragmentado. El uso de una clave fragmentada con hash para fragmentar una colección da como resultado una distribución más uniforme de los datos. Los índices hash utilizan una función hash para calcular el hash del valor del campo de índice. La función hash colapsa los documentos incrustados y calcula el hash para el valor completo, pero no admite índices de varias claves (es decir, matrices). Específicamente, crear un índice hash en un campo que contiene una matriz o intentar insertar una matriz en un campo indexado hash devuelve un error.

4. En MongoDB existen dos tipos de vistas, explique brevemente cuáles son y qué diferencias existen entre ellas. Además mencione algunos casos donde podría utilizarlas.

En MongoDB existen dos tipos de vistas: **vistas normales** y **vistas materializadas**.

Las vistas normales son consultas predefinidas basadas en agregaciones, no almacenan datos y son de solo lectura.

Las vistas materializadas no existen de forma nativa, pero se pueden simular guardando el resultado de una consulta en una colección, lo que permite un acceso más rápido aunque los datos pueden quedar desactualizados.

Las vistas normales son útiles para filtrar o resumir información sin duplicar datos, mientras que las materializadas se usan cuando se necesitan consultas rápidas sobre datos que no cambian con frecuencia.

5. Los documentos de una colección pueden diferir en la cantidad y sus tipos de campos, ¿existen algunas formas de validar los elementos a insertar en una colección para evitar esta disparidad?

Sí, en MongoDB **existen mecanismos para validar los documentos** que se insertan en una colección, y así **evitar disparidades en la cantidad o tipo de campos**.

MongoDB permite usar **reglas de validación de esquemas** mediante **validadores**. Estas reglas se definen al momento de crear una colección o se pueden agregar después con el comando `collMod`. Se escriben usando expresiones de tipo JSON Schema.

Con esta validación se puede:

- Restringir qué campos deben existir.
- Especificar el tipo de dato de cada campo (string, número, booleano, etc.).
- Establecer valores permitidos, rangos, longitudes mínimas/máximas, entre otros.

Además, se puede definir el **nivel de validación**:

- **"strict"**: rechaza cualquier documento que no cumpla las reglas.
- **"moderate"**: permite documentos inválidos si provienen de operaciones que no pasaron por validación directa (por ejemplo, actualizaciones internas).

6. MongoDB tiene soporte para transacciones, pero no es igual que el de los RDBMS. ¿Cuál es el alcance de una transacción en MongoDB?

En las bases de datos relacionales, cuando se realiza alguna operación CRUD, toda la tabla se bloquea, lo que impide el acceso a otros usuarios que quieran realizar operaciones en la misma tabla. Sin embargo, en MongoDB, solo se bloquea el documento específico que se va a modificar, lo que significa que otros usuarios aún pueden acceder y modificar otros documentos simultáneamente. Esto hace que MongoDB sea compatible con el modelo ACID a nivel de documento.

A partir de la versión 4.0 de MongoDB, se introdujeron transacciones ACID a nivel multi-documento. Esto significa que ahora es posible realizar transacciones que involucren múltiples documentos y garantizar la consistencia e integridad de los datos en esas transacciones. Sin embargo, se estima que alrededor del 90% de los modelos de datos que utilizan el enfoque de documentos en MongoDB no necesitarán transacciones, ya que el sistema proporciona un buen nivel de aislamiento y coherencia en las operaciones a nivel de documento.

Aunque la mayoría de los casos pueden ser manejados sin transacciones, existen situaciones en las que las transacciones multi-documento o multi-colección son la mejor opción. Las transacciones en MongoDB funcionan de manera similar a las transacciones en otras bases de datos. Para utilizar transacciones, primero se abre una sesión para utilizarla para ejecutar operaciones CRUD entre documentos, colecciones e incluso shards (fragmentos de datos distribuidos).

7. Las relaciones entre documentos en MongoDB pueden establecerse mediante documentos embebidos o referencias. Investigue cómo se implementa cada una y analice las ventajas y desventajas de cada una, comparándola con la forma estándar de establecer relaciones en una base de datos relacional.

Método	Descripción	Ventajas	Desventajas	Comparación con RDBMS
--------	-------------	----------	-------------	-----------------------

<b>Documentos Embebidos</b>	Un documento contiene otro documento dentro de sí.	- Consultas rápidas y atómicas	- Puede duplicar datos	Más flexible que RDBMS, no requiere joins
		- Simplicidad al leer datos relacionados	- Dificultad para actualizar datos embebidos	Menos normalizado, puede generar inconsistencias
<b>Referencias</b>	Un documento guarda el <code>_id</code> de otro para relacionarlos.	- Mantiene datos normalizados	- Requiere consultas adicionales (joins manuales)	Similar a claves foráneas, pero sin integridad automática
		- Evita duplicación de datos	- Puede afectar rendimiento por múltiples consultas	RDBMS tiene integridad referencial automática
<b>Relaciones en RDBMS</b>	Relaciones mediante claves foráneas y joins entre tablas.	- Integridad referencial garantizada	- Consultas con joins pueden ser lentas	Más rígido y estructurado, con mejor control
		- Datos normalizados y consistentes	- Menos flexible ante cambios frecuentes	

8. Tomando como referencia el modelo de los trabajos prácticos anteriores y suponiendo que este podría mapearse a una base de datos en MongoDB, proponga algunos casos donde la relación sería conveniente ser mapeada como referencia y otros como documentos embebidos. Justifique la elección.

Purchase tiene a la Review embebida ya que una review es escrita por el usuario que realiza la Purchase y no tiene relación con otra purchase. Está fuertemente conectada a la compra que la contiene. Además, se accede frecuentemente junto con la Purchase y de esta manera podemos ahorrarnos el tener la review como una colección aparte, solo existe dentro de la compra.

Purchase tiene a los itemservice embebidos ya que la compra está formada por itemservices y casi nunca se necesita la compra sin los productos que la conforman, además cada item service está ligado a una única compra y no tiene existencia fuera de la misma.

Purchase tiene a la Route embebida ya que la compra debe mantener la ruta tal como estaba en el momento de la reserva. Por lo tanto, tener la Route embebida dentro de la Purchase permite mantener ese histórico.

Route tiene por referencia a las stops ya que las stops pueden ser reutilizadas en múltiples rutas (una misma parada puede estar en muchas rutas). Además, podrían actualizarse frecuentemente. Por lo tanto, mantenerlas como referencias evita duplicación de datos y permite actualizaciones consistentes.

Route tiene por referencia a los Driver User y a los Tour Guide ya que los conductores y guías pueden estar asignados a múltiples rutas y su información puede cambiar. Por lo tanto, deben mantenerse como entidades independientes.

Supplier tiene por referencia a los services ya que un supplier puede tener múltiples Services, y estos pueden cambiar independientemente. Se requiere independencia de acceso y modificación, por lo que usar referencias es lo correcto

## Sección 2: Operaciones CRUD básicas

Descargue la última versión de MongoDB desde el sitio oficial. Ingrese al cliente de línea de comando para realizar los siguientes ejercicios.

9. Cree una nueva base de datos llamada "tours", y una colección llamada "recorridos".

i. En esta nueva colección y utilizando el comando correspondiente inserte un nuevo documento (un recorrido) con los siguientes atributos:

```
{ "nombre": "City Tour", "precio": 200, "stops": ["Diagonal Norte", "Avenida de Mayo", "Plaza del Congreso" ], "totalKm":5 }
```

```
> db.recorridos.insertOne({
  nombre: "City Tour",
  precio: 200,
  stops: ["Diagonal Norte", "Avenida de Mayo", "Plaza del Congreso"],
  totalKm: 5
})
< {
  acknowledged: true,
  insertedId: ObjectId('6840bc6939d110a773f14e2d')
}
```

ii. Recupere la información del producto usando el comando `db.products.find()` (puede agregar la función `.pretty()` al final de la expresión para ver los datos indentados). Notará que no se encuentran exactamente los atributos que insertó. ¿Cuál es la diferencia?

```
> db.recorridos.find().pretty()
< {
  _id: ObjectId('682e2a3aabe5324b2403b190'),
  nombre: 'City Tour',
  precio: 200,
  stops: [
    'Diagonal Norte',
    'Avenida de Mayo',
    'Plaza del Congreso'
  ],
  totalKm: 5
}
tours >
```

**Aclaración:** El documento no es exactamente el mismo porque el `ObjectId` es distinto al del inciso anterior, esto pasó porque al principio cuando comenzamos con la práctica no sabíamos que teníamos que ir sacando captura de cada paso, por lo que no le sacamos captura al momento de insertar el documento por primera vez.

#### Diferencia:

MongoDB **automáticamente** agrega un campo `_id` a cada documento insertado si no se lo especifica. Este campo contiene un valor único (tipo `ObjectId`) que identifica el documento internamente.

10. Agregue a la colección creada en el punto anterior, utilizando un solo comando, los documentos especificados en el documento “material\_adicional\_1.json”. Sobre ellos realice las siguientes operaciones:



```
tours> db.recorridos.insertMany([  
  {"nombre": "Historical Adventure",  
    "precio": 300,  
    "stops": [  
      "Avenida de Mayo",  
      "Plaza del Congreso",  
      "Río de la Plata",  
      "Teatro Colón",  
      "Av 9 de Julio",  
      "Plaza Italia"  
    ],  
    "totalKm": 10  
  },  
  {  
    "nombre": "Architectural Expedition",  
    "precio": 500,  
    "stops": [  
      "Usina del Arte",  
      "Puerto Madero",  
      "Museo Nacional de Bellas Artes",  
      "Museo Moderno",  
      "Museo de Arte Latinoamericano",  
      "Teatro Colón",  
      "San Telmo",  
      "Recoleta"  
    ],  
    "totalKm": 12  
  },  
  {
```

```

< {
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('68360474a5397d3512919996'),
    '1': ObjectId('68360474a5397d3512919997'),
    '2': ObjectId('68360474a5397d3512919998'),
    '3': ObjectId('68360474a5397d3512919999'),
    '4': ObjectId('68360474a5397d351291999a'),
    '5': ObjectId('68360474a5397d351291999b'),
    '6': ObjectId('68360474a5397d351291999c'),
    '7': ObjectId('68360474a5397d351291999d'),
    '8': ObjectId('68360474a5397d351291999e'),
    '9': ObjectId('68360474a5397d351291999f'),
    '10': ObjectId('68360474a5397d35129199a0'),
    '11': ObjectId('68360474a5397d35129199a1'),
    '12': ObjectId('68360474a5397d35129199a2'),
    '13': ObjectId('68360474a5397d35129199a3')
  }
}

```

i.

Actualizar el recorrido “Cultural Odyssey” para que su total de kilómetros sea 12.

```

tours> db.recorridos.updateOne(
  { nombre: "Cultural Odyssey" },
  { $set: { totalKm: 12 } }
)

```

ii.

Actualizar el listado de Stops del recorrido con nombre “Delta Tour” para agregar “Tigre”.

```

> db.recorridos.updateOne(
  { nombre: "Delta Tour" },
  { $push: { stops: "Tigre" } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

Para verificar que se agrega hacemos

```

< {
  _id: ObjectId('68360474a5397d3512919998'),
  nombre: 'Delta Tour',
  precio: 800,
  stops: [
    'Río de la Plata',
    'Bosques de Palermo',
    'Delta',
    'Tigre'
  ],
  totalKm: 8
}

```

iii. Aumentar un 10% el precio de todos los recorridos.

Primero mostramos los precios originales de los recorridos

```
> db.recorridos.find(
  {},
  { nombre: 1, precio: 1, _id: 0 }
).pretty()
< {
  nombre: 'City Tour',
  precio: 200.000000000000003
}
{
  nombre: 'Historical Adventure',
  precio: 300
}
{
  nombre: 'Architectural Expedition',
  precio: 500
}
{
  nombre: 'Delta Tour',
  precio: 800.000000000000001
}
{
  nombre: 'Nature Escape',
  precio: 400.000000000000006
}
{
  nombre: 'Artistic Journey',
  precio: 600
}
```

```
{
  nombre: 'Tango Experience',
  precio: 350.000000000000006
}
{
  nombre: 'Gastronomic Delight',
  precio: 700.00000000000001
}
{
  nombre: 'Urban Exploration',
  precio: 450.000000000000006
}
{
  nombre: 'Museum Tour',
  precio: 550
}
{
  nombre: 'Historic Landmarks',
  precio: 250
}
{
  nombre: 'Temporal Route',
  precio: 500
}
{
  nombre: 'City Lights',
  precio: 500
}
```

```
{
  nombre: 'Cultural Odyssey',
  precio: 650.00000000000001
}
{
  nombre: 'Riverfront Ramble',
  precio: 350.00000000000006
}
```

aplico el aumento de todos los recorridos

```
> db.recorridos.updateMany(
  {},
  { $mul: { precio: 1.10 } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 15,
  modifiedCount: 15,
  upsertedCount: 0
}
```

ahora verifico que en todas se aumento un 10%

```
{
  nombre: 'City Tour',
  precio: 220.000000000000003
}
{
  nombre: 'Historical Adventure',
  precio: 330
}
{
  nombre: 'Architectural Expedition',
  precio: 550
}
{
  nombre: 'Delta Tour',
  precio: 880.00000000000001
}
{
  nombre: 'Nature Escape',
  precio: 440.000000000000006
}
{
  nombre: 'Artistic Journey',
  precio: 660
}
{
  nombre: 'Tango Experience',
  precio: 385.000000000000006
}
```

```
{
  nombre: 'Gastronomic Delight',
  precio: 770.00000000000001
}
{
  nombre: 'Urban Exploration',
  precio: 495.00000000000006
}
{
  nombre: 'Museum Tour',
  precio: 605
}
{
  nombre: 'Historic Landmarks',
  precio: 275
}
{
  nombre: 'Temporal Route',
  precio: 550
}
{
  nombre: 'City Lights',
  precio: 550
}
{
  nombre: 'Cultural Odyssey',
  precio: 715.00000000000001
}
```



```
{
  nombre: 'Riverfront Ramble',
  precio: 385.000000000000006
}
```

iv. Eliminar el recorrido con nombre "Temporal Route"

elimino el recorrido

```
> db.recorridos.deleteOne(
  { nombre: "Temporal Route" }
)
< {
  acknowledged: true,
  deletedCount: 1
}
```

y para verificar que fue borrado hago lo siguiente

```
> db.recorridos.find({ nombre: "Temporal Route" }).pretty()
<
```

me devuelve vacío por que está borrado

v. Crear el array de etiquetas (tags) para la ruta "Urban Exploration". Agregue el elemento "Gastronomía" a dicho arreglo.

```

> db.recorridos.updateOne(
  { nombre: "Urban Exploration" },
  { $set: { tags: ["Gastronomía"] } }
)
< {
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}

```

para verificar que se agregó hago

```

> db.recorridos.find(
  { nombre: "Urban Exploration" },
  { nombre: 1, tags: 1, _id: 0 }
).pretty()
< {
  nombre: 'Urban Exploration',
  tags: [
    'Gastronomía'
  ]
}

```

11. Sobre la colección generada en el punto anterior realice las siguientes consultas utilizando la función find( )

i. Obtenga la ruta con nombre "Museum Tour"

```
> db.recorridos.find(
  { nombre: "Museum Tour" }
).pretty()
< {
  _id: ObjectId('68360474a5397d351291999e'),
  nombre: 'Museum Tour',
  precio: 605,
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Planetario',
    'Bosques de Palermo',
    'San Telmo',
    'La Boca - Caminito',
    'Recoleta',
    'El Monumental (Estadio River Plate)',
    'Av 9 de Julio'
  ],
  totalKm: 13
}
```

ii. Las rutas con precio superior a \$600

```
> db.recorridos.find(
  { precio: { $gt: 600 } }
).pretty()
< {
  _id: ObjectId('68360474a5397d3512919998'),
  nombre: 'Delta Tour',
  precio: 880.00000000000001,
  stops: [
    'Río de la Plata',
    'Bosques de Palermo',
    'Delta',
    'Tigre'
  ],
  totalKm: 8
}
{
  _id: ObjectId('68360474a5397d351291999a'),
  nombre: 'Artistic Journey',
  precio: 660,
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Usina del Arte',
    'Planetario',
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
    'Av 9 de Julio'
  ]
}
```

```
    'Av 9 de Julio'
  ],
  totalKm: 15
}
{
  _id: ObjectId('68360474a5397d351291999c'),
  nombre: 'Gastronomic Delight',
  precio: 770.00000000000001,
  stops: [
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
    'Recoleta',
    'Costanera Sur'
  ],
  totalKm: 9
}
{
  _id: ObjectId('68360474a5397d351291999e'),
  nombre: 'Museum Tour',
  precio: 605,
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Planetario',
    'Bosques de Palermo',
    'San Telmo',
    'La Boca - Caminito',
```

```

        'La Boca - Caminito',
        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Av 9 de Julio'
    ],
    totalKm: 13
}
{
    _id: ObjectId('68360474a5397d35129199a2'),
    nombre: 'Cultural Odyssey',
    precio: 715.00000000000001,
    stops: [
        'Bosques de Palermo',
        'San Telmo',
        'La Boca - Caminito',
        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Av 9 de Julio',
        'Plaza Italia'
    ],
    totalKm: 12
}

```

iii. Las rutas con precio superior a \$500 y con un total de kilómetros mayor a 10.

```
> db.recorridos.find(
  {
    $and: [
      { precio: { $gt: 500 } },
      { totalKm: { $gt: 10 } }
    ]
  }
).pretty()
< {
  _id: ObjectId('68360474a5397d3512919997'),
  nombre: 'Architectural Expedition',
  precio: 550,
  stops: [
    'Usina del Arte',
    'Puerto Madero',
    'Museo Nacional de Bellas Artes',
    'Museo Moderno',
    'Museo de Arte Latinoamericano',
    'Teatro Colón',
    'San Telmo',
    'Recoleta'
  ],
  totalKm: 12
}
{
  _id: ObjectId('68360474a5397d351291999a'),
  nombre: 'Artistic Journey',
  precio: 660,
```

```
    precio: 660,  
    stops: [  
      'Museo Nacional de Bellas Artes',  
      'Teatro Colón',  
      'Usina del Arte',  
      'Planetario',  
      'San Telmo',  
      'La Boca - Caminito',  
      'Belgrano - Barrio Chino',  
      'Av 9 de Julio'  
    ],  
    totalKm: 15  
  }  
  {  
    _id: ObjectId('68360474a5397d351291999e'),  
    nombre: 'Museum Tour',  
    precio: 605,  
    stops: [  
      'Museo Nacional de Bellas Artes',  
      'Teatro Colón',  
      'Planetario',  
      'Bosques de Palermo',  
      'San Telmo',  
      'La Boca - Caminito',  
      'Recoleta',  
      'El Monumental (Estadio River Plate)',  
      'Av 9 de Julio'  
    ],
```



```

        'Av 9 de Julio'
    ],
    totalKm: 13
}
{
    _id: ObjectId('68360474a5397d35129199a2'),
    nombre: 'Cultural Odyssey',
    precio: 715.00000000000001,
    stops: [
        'Bosques de Palermo',
        'San Telmo',
        'La Boca - Caminito',
        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Av 9 de Julio',
        'Plaza Italia'
    ],
    totalKm: 12
}

```

iv. Las rutas que incluyan el stop “San Telmo”

```
> db.recorridos.find(
  { stops: "San Telmo" }
).pretty()
< {
  _id: ObjectId('68360474a5397d3512919997'),
  nombre: 'Architectural Expedition',
  precio: 550,
  stops: [
    'Usina del Arte',
    'Puerto Madero',
    'Museo Nacional de Bellas Artes',
    'Museo Moderno',
    'Museo de Arte Latinoamericano',
    'Teatro Colón',
    'San Telmo',
    'Recoleta'
  ],
  totalKm: 12
}
{
  _id: ObjectId('68360474a5397d351291999a'),
  nombre: 'Artistic Journey',
  precio: 660,
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Usina del Arte',
    'Planetario',
```

```
    'Planetario',
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
    'Av 9 de Julio'
  ],
  totalKm: 15
}
{
  _id: ObjectId('68360474a5397d351291999b'),
  nombre: 'Tango Experience',
  precio: 385.000000000000006,
  stops: [
    'Avenida de Mayo',
    'Plaza del Congreso',
    'Paseo de la Historieta',
    'San Telmo',
    'La Boca - Caminito',
    'Recoleta'
  ],
  totalKm: 10
}
{
  _id: ObjectId('68360474a5397d351291999c'),
  nombre: 'Gastronomic Delight',
  precio: 770.00000000000001,
  stops: [
    'San Telmo',
```

```
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
    'Recoleta',
    'Costanera Sur'
  ],
  totalKm: 9
}
{
  _id: ObjectId('68360474a5397d351291999d'),
  nombre: 'Urban Exploration',
  precio: 495.000000000000006,
  stops: [
    'Avenida de Mayo',
    'Museo Moderno',
    'Paseo de la Historieta',
    'Usina del Arte',
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
    'Recoleta',
    'El Monumental (Estadio River Plate)',
    'Costanera Sur',
    'Plaza Italia'
  ],
  totalKm: 14,
  tags: [
    'Gastronomía'
```

```
{
  _id: ObjectId('68360474a5397d351291999e'),
  nombre: 'Museum Tour',
  precio: 605,
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Planetario',
    'Bosques de Palermo',
    'San Telmo',
    'La Boca - Caminito',
    'Recoleta',
    'El Monumental (Estadio River Plate)',
    'Av 9 de Julio'
  ],
  totalKm: 13
}
{
  _id: ObjectId('68360474a5397d351291999f'),
  nombre: 'Historic Landmarks',
  precio: 275,
  stops: [
    'Avenida de Mayo',
    'Plaza del Congreso',
    'Usina del Arte',
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
  ]
}
```

```

        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Costanera Sur',
        'Plaza Italia'
    ],
    totalKm: 23
}
{
    _id: ObjectId('68360474a5397d35129199a2'),
    nombre: 'Cultural Odyssey',
    precio: 715.00000000000001,
    stops: [
        'Bosques de Palermo',
        'San Telmo',
        'La Boca - Caminito',
        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Av 9 de Julio',
        'Plaza Italia'
    ],
    totalKm: 12
}

```

v. Las rutas que incluyan el stop “Recoleta” y no el stop “Plaza Italia”

```
> db.recorridos.find(
  {
    stops: "Recoleta",
    stops: { $not: { $eq: "Plaza Italia" } }
  }
).pretty()
< {
  _id: ObjectId('682e2a3aabe5324b2403b190'),
  nombre: 'City Tour',
  precio: 220.000000000000003,
  stops: [
    'Diagonal Norte',
    'Avenida de Mayo',
    'Plaza del Congreso'
  ],
  totalKm: 5
}
{
  _id: ObjectId('68360474a5397d3512919997'),
  nombre: 'Architectural Expedition',
  precio: 550,
  stops: [
    'Usina del Arte',
    'Puerto Madero',
    'Museo Nacional de Bellas Artes',
    'Museo Moderno',
    'Museo de Arte Latinoamericano',
    'Teatro Colón',
```

```
    'Teatro Colón',
    'San Telmo',
    'Recoleta'
  ],
  totalKm: 12
}
{
  _id: ObjectId('68360474a5397d3512919998'),
  nombre: 'Delta Tour',
  precio: 880.00000000000001,
  stops: [
    'Río de la Plata',
    'Bosques de Palermo',
    'Delta',
    'Tigre'
  ],
  totalKm: 8
}
{
  _id: ObjectId('68360474a5397d3512919999'),
  nombre: 'Nature Escape',
  precio: 440.00000000000006,
  stops: [
    'Delta',
    'Río de la Plata',
    'Av 9 de Julio',
    'Puerto Madero'
  ]
}
```



```
_id: ObjectId('68360474a5397d351291999a'),
nombre: 'Artistic Journey',
precio: 660,
stops: [
  'Museo Nacional de Bellas Artes',
  'Teatro Colón',
  'Usina del Arte',
  'Planetario',
  'San Telmo',
  'La Boca - Caminito',
  'Belgrano - Barrio Chino',
  'Av 9 de Julio'
],
totalKm: 15
}
{
  _id: ObjectId('68360474a5397d351291999b'),
  nombre: 'Tango Experience',
  precio: 385.000000000000006,
  stops: [
    'Avenida de Mayo',
    'Plaza del Congreso',
    'Paseo de la Historieta',
    'San Telmo',
    'La Boca - Caminito',
    'Recoleta'
  ],
  totalKm: 10
}
```

```
_id: ObjectId('68360474a5397d351291999c'),
nombre: 'Gastronomic Delight',
precio: 770.000000000000001,
stops: [
  'San Telmo',
  'La Boca - Caminito',
  'Belgrano - Barrio Chino',
  'Recoleta',
  'Costanera Sur'
],
totalKm: 9
}
{
  _id: ObjectId('68360474a5397d351291999e'),
  nombre: 'Museum Tour',
  precio: 605,
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Planetario',
    'Bosques de Palermo',
    'San Telmo',
    'La Boca - Caminito',
    'Recoleta',
    'El Monumental (Estadio River Plate)',
    'Av 9 de Julio'
  ],
  totalKm: 13
}
```

```
{
  _id: ObjectId('68360474a5397d35129199a3'),
  nombre: 'Riverfront Ramble',
  precio: 385.000000000000006,
  stops: [
    'Río de la Plata',
    'Bosques de Palermo',
    'El Monumental (Estadio River Plate)',
    'Costanera Sur'
  ],
  totalKm: 6
}
```

vi. El nombre y el total de km (si es que posee) de las rutas que incluyan el stop “Delta” y tenga un precio menor a 500

```
> db.recorridos.find(
  {
    stops: "Delta",
    precio: { $lt: 500 }
  },
  {
    nombre: 1,
    totalKm: 1,
    _id: 0
  }
).pretty()
< {
  nombre: 'Nature Escape'
}
```

vii. Las rutas que incluyen tanto “San Telmo” como “Recoleta” y “Avenida de Mayo” entre sus stops.

```
> db.recorridos.find(
  {
    stops: { $all: ["San Telmo", "Recoleta", "Avenida de Mayo"] }
  }
).pretty()
< {
  _id: ObjectId('68360474a5397d351291999b'),
  nombre: 'Tango Experience',
  precio: 385.000000000000006,
  stops: [
    'Avenida de Mayo',
    'Plaza del Congreso',
    'Paseo de la Historieta',
    'San Telmo',
    'La Boca - Caminito',
    'Recoleta'
  ],
  totalKm: 10
}
{
  _id: ObjectId('68360474a5397d351291999d'),
  nombre: 'Urban Exploration',
  precio: 495.000000000000006,
  stops: [
    'Avenida de Mayo',
    'Museo Moderno',
    'Paseo de la Historieta',
    'Usina del Arte',
```

```

        'Usina del Arte',
        'San Telmo',
        'La Boca - Caminito',
        'Belgrano - Barrio Chino',
        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Costanera Sur',
        'Plaza Italia'
    ],
    totalKm: 14,
    tags: [
        'Gastronomía'
    ]
}
{
    _id: ObjectId('68360474a5397d351291999f'),
    nombre: 'Historic Landmarks',
    precio: 275,
    stops: [
        'Avenida de Mayo',
        'Plaza del Congreso',
        'Usina del Arte',
        'San Telmo',
        'La Boca - Caminito',
        'Belgrano - Barrio Chino',
        'Recoleta',
        'El Monumental (Estadio River Plate)',
        'Costanera Sur',
    ]
}

```

`$all` se usa para verificar que un array contenga todos los elementos especificados, sin importar el orden ni si hay más.

viii. Solo el nombre de las rutas que dispongan de más de 5 stops

```
> db.recorridos.find(
  {
    $expr: { $gt: [ { $size: "$stops" }, 5 ] }
  },
  {
    nombre: 1,
    _id: 0
  }
).pretty()
< {
  nombre: 'Historical Adventure'
}
{
  nombre: 'Architectural Expedition'
}
{
  nombre: 'Artistic Journey'
}
{
  nombre: 'Tango Experience'
}
{
  nombre: 'Urban Exploration'
}
{
  nombre: 'Museum Tour'
```

```

{
  nombre: 'Historic Landmarks'
}
{
  nombre: 'City Lights'
}
{
  nombre: 'Cultural Odyssey'
}

```

ix. Las rutas que no tengan definida el total de sus kilómetros.

```

> db.recorridos.find(
  { totalKm: { $exists: false } }
).pretty()
< {
  _id: ObjectId('68360474a5397d3512919999'),
  nombre: 'Nature Escape',
  precio: 440.000000000000006,
  stops: [
    'Delta',
    'Río de la Plata',
    'Av 9 de Julio',
    'Puerto Madero'
  ]
}

```

x. Los nombres y el listado de stops de aquellas rutas que incluyen algún museo en sus recorridos.

```
> db.recorridos.find(
  { stops: { $elemMatch: { $regex: "Museo" } } },
  { nombre: 1, stops: 1, _id: 0 }
).pretty()
< {
  nombre: 'Architectural Expedition',
  stops: [
    'Usina del Arte',
    'Puerto Madero',
    'Museo Nacional de Bellas Artes',
    'Museo Moderno',
    'Museo de Arte Latinoamericano',
    'Teatro Colón',
    'San Telmo',
    'Recoleta'
  ]
}
{
  nombre: 'Artistic Journey',
  stops: [
    'Museo Nacional de Bellas Artes',
    'Teatro Colón',
    'Usina del Arte',
    'Planetario',
    'San Telmo',
    'La Boca - Caminito',
    'Belgrano - Barrio Chino',
    'Av 9 de Julio'
  ]
}
```



```
nombre: 'Urban Exploration',
stops: [
  'Avenida de Mayo',
  'Museo Moderno',
  'Paseo de la Historieta',
  'Usina del Arte',
  'San Telmo',
  'La Boca - Caminito',
  'Belgrano - Barrio Chino',
  'Recoleta',
  'El Monumental (Estadio River Plate)',
  'Costanera Sur',
  'Plaza Italia'
]
}
{
nombre: 'Museum Tour',
stops: [
  'Museo Nacional de Bellas Artes',
  'Teatro Colón',
  'Planetario',
  'Bosques de Palermo',
  'San Telmo',
  'La Boca - Caminito',
  'Recoleta',
  'El Monumental (Estadio River Plate)',
  'Av 9 de Julio'
]
```

```
{
  nombre: 'City Lights',
  stops: [
    'Avenida de Mayo',
    'Paseo de la Historieta',
    'Usina del Arte',
    'Recoleta',
    'El Monumental (Estadio River Plate)',
    'Costanera Sur',
    'Plaza Italia',
    'Museo de Arte Latinoamericano'
  ]
}
```

xi. Obtenga la cantidad de elementos que posee la colección.

```
> db.recorridos.countDocuments()
< 14
```

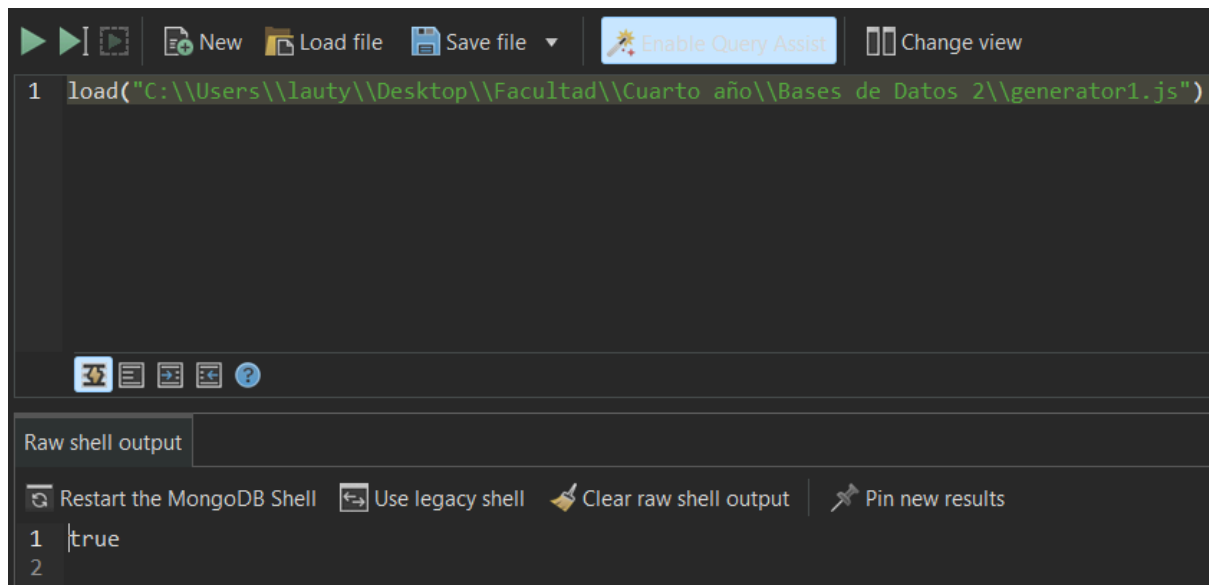
h

## Sección 3: Aggregation Framework

Aggregation Framework es una herramienta de MongoDB que nos permite ampliar la posibilidad de uso de su lenguaje de consulta. Así, podemos realizar consultas complejas, filtrado, agrupación y análisis de datos en tiempo real, utilizando una serie de etapas de procesamiento que se aplican secuencialmente a los documentos en una colección.

12. Cree una nueva base de datos llamada “tours2”.

Guarde el archivo llamado ‘generador1.js’ adjunto a esta práctica y ejecútelo con: `load(<ruta del archivo ‘generador1.js’)`. Si utiliza un cliente que lo permita (ej. Robo3T), se puede ejecutar directamente en el espacio de consultas. Examine las colecciones generadas.



Como quedo:



13. Utilizando Aggregation Framework, realice las siguientes operaciones:

Aclaró que las fotos de ejemplo de lo que retorna la consulta son parciales y no la totalidad de la respuesta.

i. Obtenga una muestra de 5 rutas aleatorias de la colección

```
> db.route.aggregate([
  { $sample: { size: 5 } }
])
< [
  {
    _id: ObjectId('6837b2a08caa32fbf5986093'),
    name: 'route26693',
    price: 529,
    totalKm: 9.01,
    stops: [
      82,
      63,
      49
    ]
  },
  {
    _id: ObjectId('6837b2738caa32fbf5980b6d'),
    name: 'route4895',
    price: 879,
    totalKm: 16.4,
```

ii. Extienda la consulta anterior para incluir en el resultado toda la información de cada una de las Stops. Note que puede ligarlas por su código.

```
> db.route.aggregate([
  { $sample: { size: 5 } },
  {
    $lookup: {
      from: "stop",
      localField: "stops",
      foreignField: "code",
      as: "stopDetails"
    }
  }
])
```

```

{
  _id: ObjectId('6837b2a98caa32fbf5987b85'),
  name: 'route33463',
  price: 289,
  totalKm: 15.04,
  stops: [
    12,
    114,
    69,
    36,
    50
  ],
  stopDetails: [
    {
      _id: ObjectId('6837b26c8caa32fbf597f7e4'),
      name: 'Cabildo de Buenos Aires',
      code: 12,
      description: 'Stop number 12'
    },
    {
      _id: ObjectId('6837b26c8caa32fbf597f7fc'),
      name: 'Centro Cultural Néstor Kirchner',
      code: 36,
      description: 'Stop number 36'
    },
    {
      _id: ObjectId('6837b26c8caa32fbf597f88a'),
      name: 'Teatro Colón',
      code: 50,
      description: 'Stop number 50'
    },
    {
      _id: ObjectId('6837b26c8caa32fbf597f81d'),
      name: 'Plaza Dorrego',
      code: 69,
      description: 'Stop number 69'
    },
    {
      _id: ObjectId('6837b26c8caa32fbf597f84a'),
      name: 'Jardín Botánico Carlos Thays',
      code: 114,
      description: 'Stop number 114'
    }
  ]
}
{
  _id: ObjectId('6837b2aa8caa32fbf5987e1b'),

```

iii. Obtenga la información de las Routes (incluyendo la de sus Stops) que tengan un

precio mayor o igual a 900

```
> db.route.aggregate([
  {
    $match: {
      price: { $gte: 900 }
    }
  },
  {
    $lookup: {
      from: "stop",
      localField: "stops",
      foreignField: "code",
      as: "stopDetails"
    }
  }
])
```

```
{
  _id: ObjectId('6837b26c8caa32fbf597f865'),
  name: 'route23',
  price: 901,
  totalKm: 8.25,
  stops: [
    103,
    94,
    91,
    45
  ],
  stopDetails: [
    {
      _id: ObjectId('6837b26c8caa32fbf597f805'),
      name: 'Floralis Genérica',
      code: 45,
      descriprion: 'Stop number 45'
    },
  ],
}
```

iv. Obtenga la información de las Routes que tengan 5 Stops o más.

```
> db.route.aggregate([
  {
    $match: {
      $expr: { $gte: [{ $size: "$stops" }, 5] }
    }
  }
])
```

```
{
  _id: ObjectId('6837b26c8caa32fbf597f852'),
  name: 'route4',
  price: 529,
  totalKm: 4.8,
  stops: [
    115,
    56,
    70,
    118,
    97
  ]
}
```

v. Obtenga la información de las Routes que tengan incluido en su nombre el string "111".

```
> db.route.aggregate([
  {
    $match: {
      name: /111/
    }
  }
])
```



```
< {
  _id: ObjectId('683efb0161c7a8e2d9441613'),
  name: 'route111',
  price: 311,
  totalKm: 19.72,
  stops: [
    34,
    27
  ]
}
{
  _id: ObjectId('683efb0361c7a8e2d94419fa'),
  name: 'route1110',
  price: 777,
  totalKm: 3.27,
  stops: [
    33,
    0,
    38,
    14
  ]
}
{
  _id: ObjectId('683efb0361c7a8e2d94419fb'),
  name: 'route1111',
  price: 400,
  totalKm: 11.87,
  stops: [
```

vi. Obtenga solo las Stops de la Route con nombre “Route100”

```
tours2> db.route.aggregate([
  {
    $match: { name: "route100" }
  },
  {
    $lookup: {
      from: "stop",
      localField: "stops",
      foreignField: "code",
      as: "stopDetails"
    }
  },
  {
    $project: {
      _id: 0,
      stopDetails: 1
    }
  }
])
```

```
< {  
  stopDetails: [  
    {  
      _id: ObjectId('683efb0161c7a8e2d9441574'),  
      name: 'Iglesia San Pedro Telmo',  
      code: 70,  
      desciprion: 'Stop number 70'  
    },  
    {  
      _id: ObjectId('683efb0161c7a8e2d944158e'),  
      name: 'Museo Islas Malvinas e Islas del Atlántico Sur',  
      code: 96,  
      desciprion: 'Stop number 96'  
    },  
    {  
      _id: ObjectId('683efb0161c7a8e2d9441594'),  
      name: 'Puente de la Mujer',  
      code: 102,  
      desciprion: 'Stop number 102'  
    },  
    {  
      _id: ObjectId('683efc1161c7a8e2d944d93b'),  
      name: 'Iglesia San Pedro Telmo',  
      code: 70,  
      desciprion: 'Stop number 70'  
    },  
  ],  
}
```

vii. Obtenga la información del Stop que más apariciones tiene en Routes.

```

tours2> db.route.aggregate([
  { $unwind: "$stops" },
  { $group: {
    _id: "$stops",
    count: { $sum: 1 }
  }},
  { $sort: { count: -1 } },
  { $limit: 1 },
  {
    $lookup: {
      from: "stop",
      localField: "_id",
      foreignField: "code",
      as: "stopInfo"
    }
  },
  {
    $unwind: "$stopInfo"
  },
  {
    $project: {
      _id: 0,
      code: "$_id",
      appearances: "$count",
      name: "$stopInfo.name",
      description: "$stopInfo.description"
    }
  }
])

```

```

< {
  code: 89,
  appearances: 1563,
  name: 'Basílica Nuestra Señora del Pilar',
  description: 'Stop number 89'
}

```

viii. Obtenga las Route que tengan un precio inferior a 150. A ellos agregué una nueva propiedad que especifique la cantidad de Stops que posee la Route. Cree una nueva colección llamada "rutas\_economicas" y almacene estos elementos.

```
tours2> db.route.aggregate([
  {
    $match: {
      price: { $lt: 150 }
    }
  },
  {
    $addFields: {
      stopCount: { $size: "$stops" }
    }
  },
  {
    $out: "rutas_economicas"
  }
])
```

_id	name	price	totalKm	stops	stopCount
ObjectId('683f0a3961c7a8e2d9459d40')	route14	105	16.2	Array (3)	3
ObjectId('683f0a3961c7a8e2d9459d4e')	route28	118	17.15	Array (4)	4
ObjectId('683f0a3961c7a8e2d9459d52')	route32	138	13.46	Array (4)	4
ObjectId('683f0a3961c7a8e2d9459d53')	route33	103	12.42		

ix. Por cada Stop existente en su colección, calcule el precio promedio de las Routes que la incluyen

Tener en cuenta: si la consulta se empieza a tornar difícil de leer, se pueden ir guardando los agregadores en variables, que no son más que objetos en formato JSON.

```

tours2> let unwindStops = { $unwind: "$stops" };
    let averageByStopCode = {
      $group: {
        _id: "$stops",
        avgPrice: { $avg: "$price" },
        count: { $sum: 1 }
      }
    };
    let lookupStop = {
      $lookup: {
        from: "stop",
        localField: "_id",
        foreignField: "code",
        as: "stopInfo"
      }
    };
    let unwindStopInfo = { $unwind: "$stopInfo" };
    let projectResult = {
      $project: {
        _id: 0,
        stopCode: "$_id",
        stopName: "$stopInfo.name",
        avgPrice: 1,
        count: 1
      }
    };

```

```

db.route.aggregate([
  unwindStops,
  averageByStopCode,
  lookupStop,
  unwindStopInfo,
  projectResult
])

```

```
< {
  avgPrice: 554.321083172147,
  count: 1551,
  stopCode: 100,
  stopName: 'Paseo de la Gloria'
}
{
  avgPrice: 553.1328671328671,
  count: 1430,
  stopCode: 8,
  stopName: 'Casa Rosada'
}
{
  avgPrice: 537.3928077455048,
  count: 1446,
  stopCode: 20,
  stopName: 'Paseo de la Historieta'
}
{
  avgPrice: 552.7814432989691,
  count: 1455,
  stopCode: 40,
  stopName: 'Colección de Arte Amalia Lacroze de Fortabat'
}
{
  avgPrice: 555.1712931618144,
  count: 1477,
  stopCode: 37,
```