Introducción a MongoDB

Big Data Aplicado

Dr. Francisco E. Cabrera

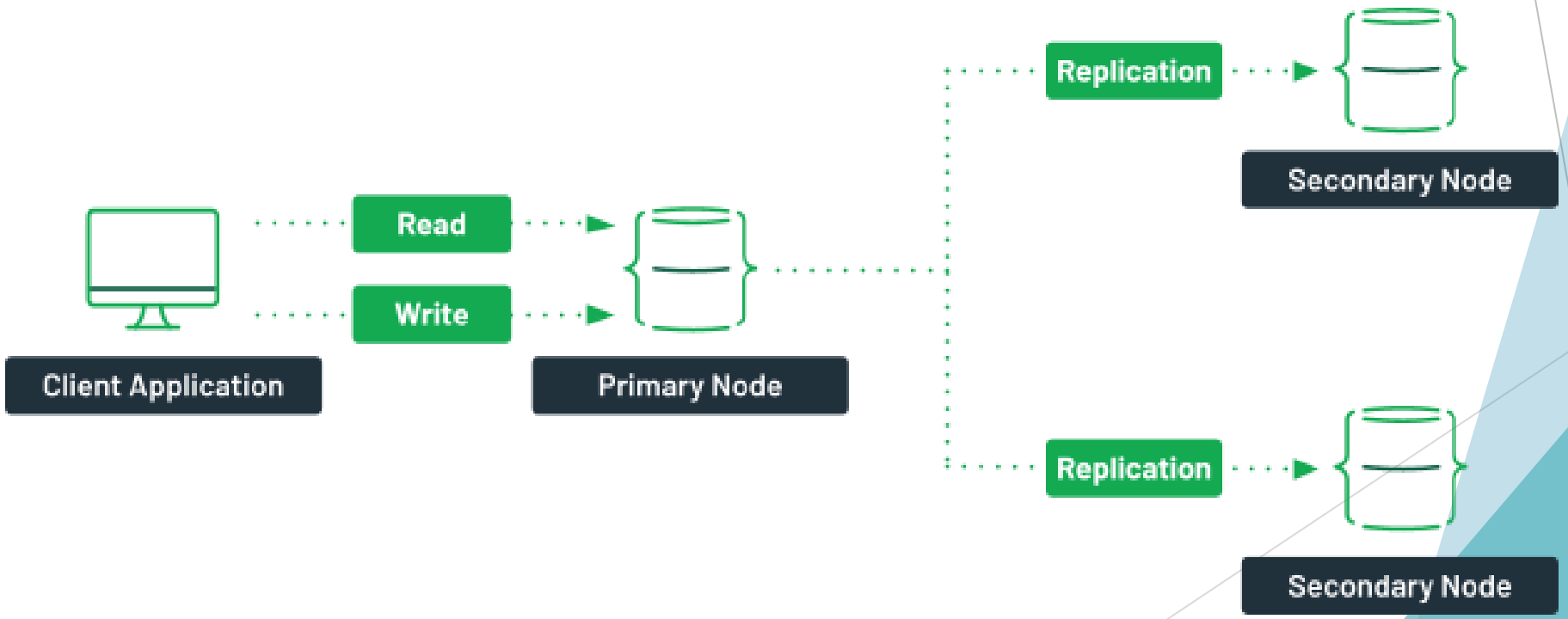
MongoDB

- Mongo DB es una base de datos NoSQL que permite gestionar datos semiestructurados en forma de documentos.
- Entre sus características destacan:
  - Escalabilidad: Gracias a la capacidad de partir la información mediante sharding repartiendo la carga entre distintos nodos de nuestro cluster.
  - Flexibilidad: Con la capacidad de tratar datos semiestructurados.
  - Replicación: Con la posibilidad de crear replicasets, que son conjutos de datos repetidos almacenados en distintos nodos de nuestro cluster.
- Documentación: https://www.mongodb.com/docs/

Comandos mas útiles de MongoDB

Cliente de Mongo

- Es la conexión entre la aplicación y la base de datos
- Se establece mediante el objeto MongoClient en PyMongo



Conexión mediante el cliente de Mongo

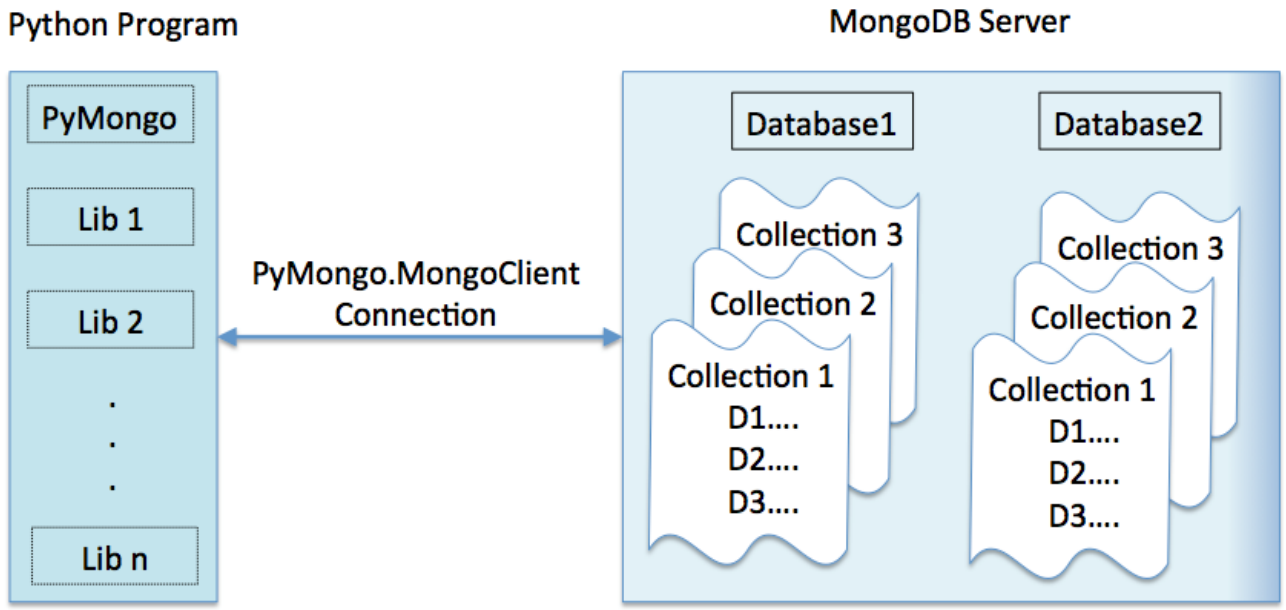
- Conectar a MongoDB
- Listar y Seleccionar Bases de Datos

```
# Conectar
from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27017/’)

# Listar y seleccionar
client.list_database_names()
db = client['mi_db']
```

Colecciones en MongoDB

- Es el equivalente a una tabla en bases de datos relacionales
- Contiene documentos que pueden tener diferentes estructuras



Colecciones

- Listar Colecciones
- Crear y Seleccionar una Colección
- Eliminar una Colección

```
# Listar
db.list_collection_names()

#Crear y Seleccionar
coleccion = db['mi_coleccion']
db.create_collection('nueva_coleccion’)

# Eliminar
db.drop_collection('nueva_coleccion')
```

Documentos de MongoDB

- Es el equivalente a una fila en bases de datos relacionales
- Se almacena en formato BSON (similar a JSON)

Operaciones CRUD: Insertar

- Insertar un Documento
- Insertar varios Documentos

```
# Insertar un documento
coleccion.insert_one({'nombre': 'Juan', 'edad': 30})

# Insertar varios documentos
docs = [{'nombre': 'Ana', 'edad': 25}, {'nombre': 'Pedro', 'edad': 35}]
coleccion.insert_many(docs)
```

Operaciones CRUD: Consultar

- Recuperar un Documento
- Recuperar todos los Documentos
- Filtrar con Condiciones

```
# Recuperar un documento
resultado = coleccion.find_one({'nombre': 'Juan’})

# Recuperar varios documentos
for doc in coleccion.find():
    print(doc)

# Filtrar con Condiciones
filtro = coleccion.find({'edad': {'$gt': 25}})
```

Operaciones CRUD: Actualizar

- Actualizar un Documento
- Actualizar varios Documentos

```
# Actualizar un documento
coleccion.update_one({'nombre': 'Juan'}, {'$set': {'edad': 31}})

# Actualizar varios documentos
coleccion.update_many({
    'edad': {'$lt': 30}},
    {'$set': {'categoria': 'joven’}
})
```

Operaciones CRUD: Eliminar

- Eliminar un documento
- Eliminar varios documentos

```
# Eliminar un documento
coleccion.delete_one({'nombre': 'Juan’})

# Eliminar varios documentos
coleccion.delete_many({'edad': {'$lt': 30}})
```

Operaciones avanzadas

- Ordenar resultados
- Limitar resultados
- Contar documentos

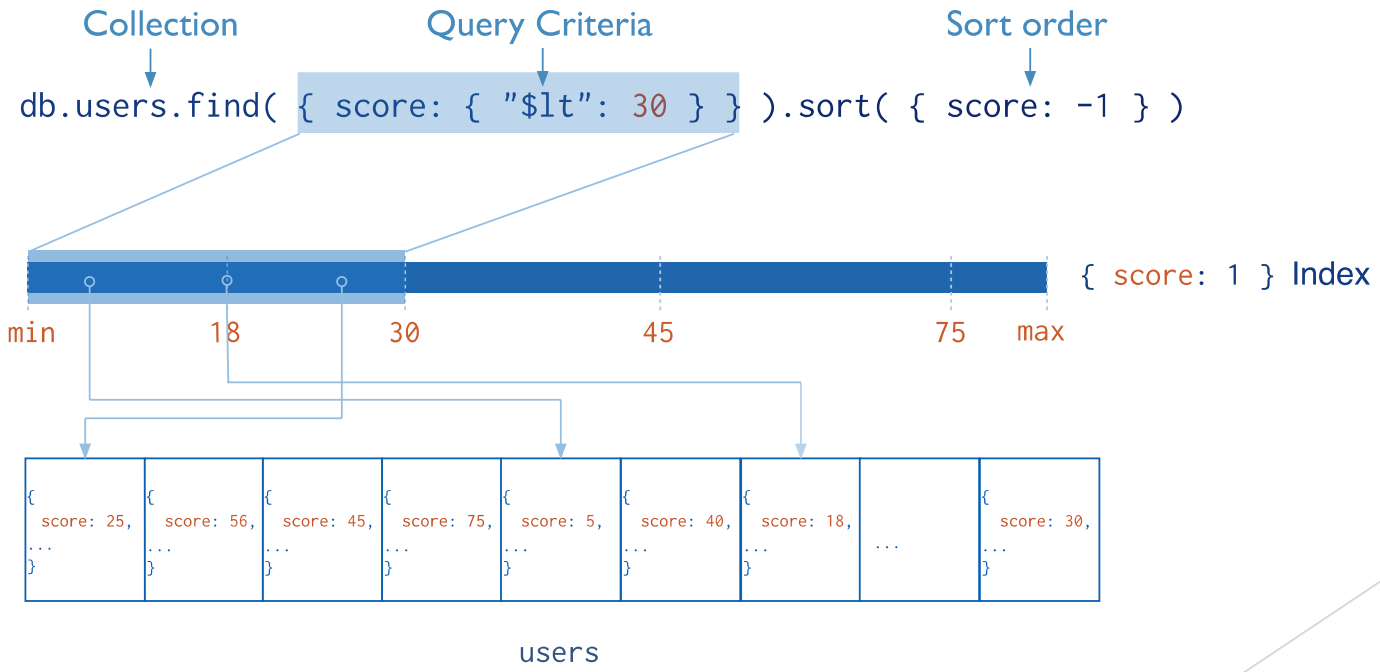
```
# Ordenar resultados
coleccion.find().sort('edad', 1)  # Ascendente
coleccion.find().sort('edad', -1)  # Descendente

# Limitar resultados
coleccion.find().limit(5)

# Contar documentos
total = coleccion.count_documents({})
```

Índices en MongoDB

- Facilita la búsqueda y mejora el rendimiento en consultas
- Se pueden crear índices en uno o varios campos



Índices

- Crear Índice en un Campo
- Crear Índice Compuesto
- Mostrar o eliminar Índices

```
# Crear índice
coleccion.create_index('nombre’)

# Crear índice compuesto
coleccion.create_index([('nombre', 1), ('edad', -1)])

# Mostrar índices
coleccion.index_information()

# Eliminar índices
coleccion.drop_index('nombre_1')
```

Agregaciones

- Las agregaciones filtran, agrupan y transforman documentos
- Agrupar Datos
  - Podemos crear una agrupación utilizando aggregate y pasándole un pipeline.

```
# Definimos el pipeline
pipeline = [
    {'$group': {'_id': '$edad', 'total': {'$sum': 1}}},
    {'$sort': {'total': -1}}
]
# Agrupamos la colección por el pipeline definido
resultado = coleccion.aggregate(pipeline)

for r in resultado:
    print(r)
```