Clase 046 — NoSQL: MongoDB con pymongo

Parte: 0 — Prerrequisitos · Fuente: MongoDB docs · pymongo docs · MongoDB: The Definitive Guide (Bradshaw et al.) cap. 1. · Duración estimada: 75 min.

Clase 046 — NoSQL: MongoDB con pymongo

> *Parte: 0 — Prerrequisitos · Fuente: MongoDB docs · pymongo docs · MongoDB: The Definitive Guide (Bradshaw et al.) cap. 1. · Duración estimada: 75 min.*

Objetivo

Que el alumno entienda el modelo NoSQL documento (collections de JSON-like), cuándo conviene sobre SQL, y use pymongo para CRUD básico + queries con operadores típicos. Sin pretender competir con un curso entero de MongoDB.

Resultados de aprendizaje

Al finalizar la clase, el alumno podrá:

1. Diferenciar modelo relacional (tablas + filas) vs documento (collections + docs JSON).
2. Reconocer cuándo NoSQL aporta (schema flexible, datos jerárquicos, escala horizontal).
3. Conectar con pymongo, hacer insert/find/update/delete.
4. Filtrar con operadores: $gt, $lt, $in, $regex, $and, $or.
5. Hacer agregaciones con el pipeline ($match, $group, $sort).

Temas

#	Tema	Por qué importa
1	SQL vs NoSQL — cuándo cada uno	No "NoSQL es mejor" — distinto.
2	Modelo documento: collections + docs JSON	Schema flexible.
3	pymongo: connect, insert_one, find, update	CRUD básico.
4	Operadores de query: $gt/$lt/$in/$regex	Equivalentes a WHERE.
5	Aggregation pipeline	$match/$group/$sort — análogo a SQL.
6	Cuándo NO usar Mongo	Cuando relacional es claramente mejor.

Definiciones y características

NoSQL documento

: Familia de DBs que almacena documentos JSON-like (BSON en Mongo) en lugar de filas en tablas. Schema flexible — cada documento puede tener campos distintos.

Collection

: Equivalente a una tabla en SQL, pero sin schema fijo. Contiene documentos del mismo "tipo" lógico (productos, usuarios, eventos).

Documento (dict BSON)

: Unidad de almacenamiento. JSON con tipos extra (Date, ObjectId, Decimal128). Puede contener arrays y sub-documentos anidados.

Operador ($gt, $in, $elemMatch)

: Prefijo $ en las queries Mongo: {'precio': {'$gt': 100}} ≈ WHERE precio > 100. La query es un dict JSON, no string.

Aggregation pipeline

: Equivalente Mongo a CTEs encadenadas: lista de etapas ($match, $group, $sort, $project, $lookup) que procesan documentos secuencialmente.

$elemMatch

: Operador para filtrar por condiciones sobre elementos de un array dentro del documento. Útil con arrays de sub-docs (reviews dentro de producto).

Dataset / recursos

MongoDB local (Docker o Atlas free tier) — o usar mongomock para tests. Datos sintéticos: collection de productos.

Ejercicios

1. CRUD básico. Conecta a Mongo (o mongomock), inserta 5 productos, lee todos, actualiza uno, borra uno.

2. Find con operadores. Productos con precio > 100 y categoría en ['libros', 'musica'].

3. Update con $set y $inc. Incrementa stock de un producto en 10 unidades.

4. Aggregation pipeline. Promedio de precio por categoría con $group.

5. Documento jerárquico. Inserta un producto con array de reviews (sub-documentos). Consulta los que tienen alguna review con rating < 3 usando $elemMatch.

Homework verificable

Notebook con mongomock (no requiere Mongo real): (a) collection productos con 20 docs sintéticos; (b) 5 queries demostrando operadores; (c) aggregation pipeline con $match → $group → $sort; (d) reporte: 3 casos donde Mongo es mejor que SQL y 3 donde no.

Criterio de aceptación: Las queries funcionan; el reporte tiene casos justificados.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
pymongo.errors.ServerSelectionTimeoutError	No conecta al servidor. Fix: verifica que
Update sin $set reemplaza el documento ent	update_one(filter, {'precio': 100}) reempl
Aggregation con $group sin _id falla	$group requiere _id (la key de agrupación,
Query con {} devuelve todo (no None)	{} es "sin filtro" en Mongo. Si querías ma
Cuento con count_documents({}) y va lento	Sin filtro, recorre toda la collection. Fi

Preguntas frecuentes

¿SQL o NoSQL?

SQL para datos tabulares con relaciones, integridad referencial, reporting/BI. NoSQL documento para schema variable, datos jerárquicos naturales, escala write masiva. No es "mejor" — distinto.

¿find_one o find?

find_one devuelve dict (o None). find devuelve cursor iterable. Para 1 doc: find_one. Para muchos: list(coll.find(query)) o iterar el cursor.

¿pymongo vs Motor (async)?

pymongo síncrono, default. Motor asíncrono (asyncio) — para web apps con muchas concurrent connections.

¿Cómo tipo los documentos en Python?

Usa pydantic con BaseModel. Convierte dict ↔ tipo validado. Combinado con FastAPI, casi gratis (verás en MLOps).

¿Mongo para data science?

Como fuente sí (extraes datos con aggregation, los pasas a pandas). Para análisis ya no — pandas/DuckDB son mejores. Mongo brilla en operaciones (logs, eventos).

Referencias

- pymongo docs
- MongoDB query operators
- mongomock
- Bradshaw, MongoDB: The Definitive Guide 3e, cap. 1.

Siguiente clase

Clase 047 — APIs REST con requests

Apéndice: notebook (primer bloque)

Para este notebook usamos mongomock para no requerir Mongo real: `bash pip install mongomock pymongo

```
try:
    import mongomock
    client = mongomock.MongoClient()
    print('mongomock OK')
except ImportError:
    print('Instala: pip install mongomock')

db = client['lab044']
productos = db['productos']
```

Archivos complementarios

- notebook.ipynb