

Chapter 1: Replica Set Transactions

introduction

- snapshot isolation
- all-or-nothing execution
- = ACID Data Guarantess en multiples comandos y múltiples documentos a través de 1 o varias colecciones → transacciones
- posible añadir a quien lo requiera, no efecta a «performance» si no es necesario
 - multi-document transactions
 - transacciones funcionan a través de los replica sets
 - entre shared clusters a partir de MongoDB 4.2

Transaction Considerations

- WiredTiger cache puede retirar snapshots y abortar transacciones si se llega a los límites
- tecnología de snapshots para asegurar las transacciones
- MongoDB tira atrás cualquier transacción que dure más de 60 segundos, haciendo un roll back
- aunque no hay límite, no se deberían de modificar más de 1000 documentos en una transacción
 - si afecta a más documentos, el desarrollador debería separarlo en lotes
- hay que contemplar la recogida de las excepciones por el exceso de tiempo (error transitorio de la red, elección?)
- Operaciones DDL (crear indice, borrar coleccion) → bloqueo de transacciones en el «namespace»

Write Conflicts

- si no se obtiene el «lock» para realizar la operación de escritura, la operación se aborta
 - 2 transacciones sobre el mismo documento, la segunda es abortada para evitar deadlocks
- no afecta a las operaciones de lectura

Transactions Use Cases

- se estima que el 80%-90% de las aplicaciones no necesitan el uso de transacciones
 - pero no están acostumbrados a este nuevo tipo de paradigma o siguen con la inercia de las BBDD relacionales
 - o están preocupados por si en un futuro si que lo necesitan
- MongoDB se encarga de las transacciones a multi-documento automaticamente

Abort vs Commit

- agresiva política para evitar los deadlock
- el desarrollador ha de tener control sobre operaciones abortadas
- errores en commit
 - se reintentan 1 vez desde el server
 - después se coge la excepción y se trata (con repeticiones o abortándola)

```
def run_transaction_and_retry_commit(client):  
    with client.start_session() as s:
```

```

s.start_transaction()
collection_one.insert_one(doc1, session=s)
collection_two.insert_one(doc2, session=s)
while True:
    try:
        s.commit_transaction()
        break
    except (OperationFailure, ConnectionFailure) as exc:
        if exc.has_error_label("UnknownTransactionCommitResult"):
            print("Unknown commit result, retrying...")
            continue
        raise

while True:
    try:
        return run_transaction_and_retry_commit(client)
    except (OperationFailure, ConnectionFailure) as exc:
        if exc.has_error_label("TransientTransactionError"):
            print("Transient transaction error, retrying...")
            continue
        raise

```

From:

<https://matebcn.eu/> - miguel angel torres egea

Permanent link:

<https://matebcn.eu/info:cursos:mongodbuniversity:m040:cap1>

Last update: **20/01/2019 20:17**

