Giải thích bookstore01Db_session07

29 Tháng Mười 2025 3:29 CH

Lưu ý chung về tên database

Trên MongoDB, tên DB **phân biệt hoa/thường**

1) Tạo CSDL & Collections

```
use("bookstore01Db")
db.createCollection("authors")
db.createCollection("books")
```

Khái niệm & cú pháp

- `use("<dbName>")`: chọn hoặc tạo (lười) DB nếu chưa có. Không tạo file ngay, Mongo chỉ thực sự tạo khi có dữ liệu.
- `db.createCollection("<name>")`: tạo collection trống.

Gợi ý sửa: đổi thành `use("bookstore01db")` cho nhất quán.

2) Insert dữ liệu mẫu

```
use("bookstore01Db")
db.authors.insertMany([
  { _id: 1, name: "Nguyen Nhat Anh", country: "Vietnam" },
  { _id: 2, name: "Haruki Murakami", country: "Japan" },
  { _id: 3, name: "J.K. Rowling", country: "UK" }
])
use("bookstore01Db")
db.books.insertMany([
  { title: "Cho tôi xin một vé đi tuổi thơ", authorId: 1, price: 8.5, sold: 20000 },
  { title: "Mắt biếc", authorId: 1, price: 7.2, sold: 18000 },
  { title: "Kafka on the Shore", authorId: 2, price: 12, sold: 25000 },
  { title: "Norwegian Wood", authorId: 2, price: 10, sold: 30000 },
  { title: "Harry Potter 1", authorId: 3, price: 15, sold: 50000 },
  { title: "Harry Potter 2", authorId: 3, price: 17, sold: 48000 }
])

use("bookstore01Db")
db.books.find()
```

Khái niệm & cú pháp

- `insertMany([...])`: chèn nhiều document một lúc.
- Trường `_id` (trong `authors`) là **khóa chính**. Bạn tự đặt `_id` số nguyên; còn `books` dùng `_id` mặc định ObjectId.
- Quan hệ logic: `books.authorId` trỏ tới `authors._id`.

Kiểm tra nhanh

`db.books.find()`

- `find()` trả về cursor, mặc định log ra vài dòng. Dùng `.pretty()` để dễ nhìn (cũ), hoặc để nguyên cũng được.

Gợi ý sửa: thống nhất `use("bookstore01db")` trước khi insert.

3) Câu 1 — Phân quyền (Users & Roles)

```
/*Câu 1 – Phân quyền
Tạo hai người dùng:
readUser: chỉ có quyền đọc dữ liệu trong bookstoreDB*/
use("bookstore01Db")
db.createUser(
  {
    user: "readUser",
    pwd: "read123",
    roles: [{ role: "read", db: "bookstore01Db" }]
  }
)
/*writeUser: có quyền đọc và ghi dữ liệu trong bookstoreDB*/
use("bookstore01Db")
db.createUser(
  {
    user: "writeUser",
    pwd: "write123",
    roles: [{ role: "readWrite", db: "bookstore01db" }]
  }
)
```

Khái niệm

- MongoDB có mô hình **role-based access control (RBAC)**.
- `db.createUser({ user, pwd, roles })`: tạo user **trong DB hiện tại**. Role xác định quyền.

Quan trọng để có hiệu lực

- Phải chạy mongod với `--auth`, và thường tạo **admin user** trong DB `admin` trước, rồi đăng nhập admin để tạo user các DB khác.
- Ở đây bạn gán role trỏ tới DB `"bookstore01db"` nhưng `use()` đang là `"bookstore01Db"`. Nên đồng nhất.

Gợi ý an toàn

- Dùng: `use("bookstore01db")` ở cả **createUser** và `roles.db`.

4) Câu 2 — Transaction + Rollback

4.1 Chuẩn bị server

Bạn ghi:

`mongod --port 27018 --dbpath "D:\APTECH\13MongoDB\Class\Session07" --replSet rs0 --bind_ip localhost`

- **Transactions đa tài liệu cần Replica Set** (dù 1 node). Lệnh trên đã bật `--replSet`.
- Sau khi mở mongod, lần đầu cần `rs.initiate()`:
  `mongosh --port 27018 rs.initiate()`
- Khi truy vấn, kết nối đúng cổng: `mongosh --port 27018`.

4.2 Mã transaction của bạn

```
/*Câu 2 – Transaction + Rollback :
mongod --port 27018 --dbpath ""D:\APTECH\13MongoDB\Class\Session07"" --replSet rs0 --bind_ip localhost
Tạo một transaction gồm 2 thao tác:
Thêm 1 quyển sách mới vào books
Cập nhật tổng số sách của tác giả tương ứng trong authors
Nếu 1 thao tác thất bại → rollback toàn bộ transaction.*/
use("bookstore01Db")
//2.1.start session
session = db.getMongo().startSession();
//2.2.choice database
bsdb = session.getDatabase("bookstore01Db");
//2.3.start transaction
session.startTransaction();
try {
  //2.3.1.thêm 1 quyển sách mới vào books
  bsdb.books.insertOne({
    title: "Harry Potter 3",
    authorId: 3,
    price: 18,
    sold: 37000
  })
  //2.3.2.cập nhật tổng số sách của tác giả tương ứng trong authors
  const up=bsdb.authors.updateOne(
    {_id:3},
    {$inc:{bookCount:1}}
  );
  if(up.matchedCount!==1)
    throw new Error("Author not found -> rollback");
  //2.3.3.gửi thông báo
  bsdb.Notification.insertOne({
    message: "New book added",
    read: false
  })
  //2.3.4.commit transaction xác nhận thay đổi
  session.commitTransaction();
  print("Transaction commit successfully")
} catch (error) {
  //2.3.5.rollback nếu lỗi
  session.abortTransaction();
  print("Transaction rollback: " + error)
} finally {
  //2.3.6.kết thúc session
  session.endSession();
}
```

Khái niệm & cú pháp

- `startSession()`: mở một **session**.
- `session.getDatabase("<db>")`: lấy "view" của DB gắn với session — mọi thao tác qua `bsdb.*` **nằm trong transaction**.
- `startTransaction()`: bắt đầu transaction.
- `commitTransaction()`: xác nhận tất cả thay đổi.
- `abortTransaction()`: **rollback** mọi thay đổi trong transaction.

Lỗi/nhắc nhở nhỏ

- `Notification` collection chưa tạo trước đó. Nếu chỉ minh họa 2 thao tác (insert book + update author) thì có thể bỏ bước này để đỡ rủi ro lỗi ngoài yêu cầu. Nếu muốn có, hãy dùng tên thống nhất như `notifications`.

Phiên bản đã chỉnh nhẹ cho chắc chắn:

```
use("bookstore01db")
const session = db.getMongo().startSession();
const bsdb = session.getDatabase("bookstore01db");
session.startTransaction({ writeConcern: { w: "majority" } });

try {
  bsdb.books.insertOne({ title: "Harry Potter 3", authorId: 3, price: 18, sold: 37000 });

  const up = bsdb.authors.updateOne({ _id: 3 }, { $inc: { bookCount: 1 } });
  if (up.matchedCount !== 1) throw new Error("Author not found -> rollback");

  // tùy chọn: bsdb.notifications.insertOne({ message: "New book added", read: false });

  session.commitTransaction();
  print("✅ Transaction committed");
} catch (e) {
  session.abortTransaction();
  print("❌ Rolled back: " + e.message);
} finally {
  session.endSession();
}
```

5) Câu 3 — Aggregation với `$lookup` (Join sách + tác giả)

```
//Câu 3 – Aggregation 1 (Sử dụng $lookup): Hiển thị danh sách sách kèm tên tác giả tương ứng.
use("bookstore01Db")
db.books.aggregate([
  {
    $lookup: {
      from: "authors",
      localField: "authorId",
      foreignField: "_id",
      as: "author"
    }
  },
  { $unwind: "$author" },
  {
    $project: {
      _id: 0,
      title: 1,
      price: 1,
      sold: 1,
      authorName: "$author.name",
      authorCountry: "$author.country"
    }
  }
])
```

Khái niệm & cú pháp

- **Aggregation pipeline**: chuỗi stage xử lý dữ liệu.
- `$lookup`: "join" giữa `books` (pipeline hiện tại) và `authors`.
- `localField`: trường ở collection hiện tại.
- `foreignField`: trường ở collection "from".
- `as`: tên mảng kết quả join.
- `$unwind`: "trải phẳng" mảng thành từng document riêng.
- `$project`: chọn/đổi tên/trả về field; `1` là giữ nguyên, `0` là bỏ; `"$path"` truy cập giá trị field.

Chú ý: `use("bookstore01db")` nhất quán.

6) Câu 4 — Tổng doanh thu theo tác giả

```
//Câu 4 – Aggregation 2: Tính tổng doanh thu (price × sold) của từng tác giả.
use("bookstore01Db")
db.books.aggregate([
  //tính tổng doanh thu theo từng tác giả
  {
    $group: {
      _id: "$authorId",
      totalRevenue: {
        $sum: { $multiply: ["$price", "$sold"] }
      },
      totalBooks: { $sum: 1 }
    }
  },
  //join để lấy tên tác giả
  {
    $lookup: {
      from: "authors",
      localField: "_id",
      foreignField: "_id",
      as: "author"
    }
  },
  { $unwind: "$author" },
  {
    $project: {
      _id: 1,
      authorId: "$_id",
      authorName: "$author.name",
      totalBooks: 1,
      totalRevenue: 1
    }
  }
])
```

Khái niệm & cú pháp

- `$group`: gom nhóm theo `_id` (ở đây là `"$authorId"`). Các **accumulator**:
- `$sum`, `$avg`, `$min`, `$max`, …
- `$multiply`: phép nhân trong pipeline.
- `$lookup` → join lại để lấy `author.name`.
- `$project`: xuất các trường mong muốn.

Lưu ý nhỏ: Bạn đang xuất cả `_id` và `authorId` (trùng nghĩa). Có thể bỏ `_id` cho gọn:

```
{ $project: { _id: 0, authorId: "$_id", authorName: "$author.name", totalBooks: 1, totalRevenue: 1 } }
```

7) Câu 5 — Lọc sách bán > 25.000 + tên tác giả

```
//Câu 5 – Aggregation 3: Liệt kê các quyển sách có số lượng bán > 25.000 và kèm theo tên tác giả.
use("bookstore01Db")
db.books.aggregate([
  {
    $match: {
      sold: { $gt: 25000 }
    }
  },
  {
    $lookup: {
      from: "authors",
      localField: "authorId",
      foreignField: "_id",
      as: "author"
    }
  },
  { $unwind: "$author" },
  {
    $project: {
      _id: 0,
      title: 1,
      sold: 1,
      authorName: "$author.name"
    }
  }
])
```

Khái niệm & cú pháp

- `$match`: lọc trước khi join để pipeline **nhanh** hơn (ít tài liệu hơn đi qua `$lookup`).
- `$gt`: so sánh "greater than".

8) Câu 6 — Index & explain()

```
/*Câu 6 – Index
Tạo index đơn trên trường authorId.
Tạo compound index trên (price, sold).
Kiểm tra hiệu quả truy vấn bằng explain().*/
use("bookstore01Db")
// tạo Index đơn trên authorId
db.books.createIndex({ authorId: 1 })
// Compound index (price, sold)
db.books.createIndex({ price: 1, sold: -1 })
//Kiểm tra hiệu quả bằng explain("executionStats")
//Truy vấn dùng index authorId:
db.books.find({ authorId: 2 }).explain("executionStats")
//Truy vấn lợi thế compound index: lọc theo price và sort theo sold cùng chiều như index (prefix rule)
db.books.find({ price: { $gte: 10 } })
  .sort({ sold: -1 })
  .explain("executionStats")
```

Khái niệm & cú pháp

- `createIndex({ field: direction })`: tạo chỉ mục. `1` tăng dần, `-1` giảm dần.
- **Compound index** `{ price: 1, sold: -1 }`:
- **Prefix rule**: chỉ mục phục vụ tốt khi filter/sort bắt đầu từ **đúng thứ tự tiền tố**.

  Ví dụ: lọc bởi `price` rồi sort `sold` sẽ tận dụng tốt index.
- `explain("executionStats")`: hiển thị kế hoạch chạy và thống kê:
- `totalKeysExamined`, `totalDocsExamined`, `executionTimeMillis`,…
- Tìm `IXSCAN` (dùng index) vs `COLLSCAN` (quét cả collection).

9) Câu 7 — Giá trung bình từng tác giả > 10

Bản 1 (đúng theo ý bạn viết)

```
//Câu 7 – Aggregation nâng cao:
//Tính giá trung bình của sách từng tác giả, và chỉ hiển thị các tác giả có giá trung bình > 10.
use("bookstore01Db")
db.books.aggregate([
  //tính giá trung bình sách của từng tác giả
  {$group: {
    _id: "$authorId",
    avgPrice: {
      $avg:"$price"
    },
    countBooks:{$sum:1}
  }},
  //join để lấy thông tin tác giả
  {$lookup: {
    from: "authors",
    localField: "_id",
    foreignField: "_id",
    as: "author"
  }},
  {$unwind: "$author"},
  {$project: {
    _id:0,
    authorId:"$_id",
    authorName:"$author.name",
    avgPrice:{$round:["$avgPrice",2]},
    countBooks:1
  }},

  //sắp xếp lấy giá trung bình >10
  {$match: {
    avgPrice:{$gt:10},
  }},
  {$sort: {
    avgPrice:-1
  }}
])
```

Khái niệm & cú pháp

- `$avg`: tính trung bình.
- `$round: [<number>, <digits>]`: làm tròn.
- `"$path"`: tham chiếu field trong pipeline (`"$_id"`, `"$author.name"`).
- `$match` và `$sort` giống đã giải thích phía trên.

Bản tối ưu (lọc sớm trước khi `$lookup`)

```
//gợi ý: nên $match avgPrice>10 lên trước, để khỏi phải join các mục avgPrice<10
use("bookstore01db")
db.books.aggregate([
  { $group: {
      _id: "$authorId",
      avgPrice: { $avg: "$price" },
      countBooks: { $sum: 1 }
  }},
  { $match: { avgPrice: { $gt: 10 } } },  // lọc sớm
  { $lookup: {
      from: "authors",
      localField: "_id",
      foreignField: "_id",
      as: "author"
  }},
  { $unwind: "$author" },
  { $project: {
      _id: 0,
      authorId: "$_id",
      authorName: "$author.name",
      avgPrice: { $round: ["$avgPrice", 2] },
      countBooks: 1
  }},
  { $sort: { avgPrice: -1 } }
])
```

Vì sao tối ưu?

Bạn giảm số tài liệu được join (chỉ các tác giả đạt `avgPrice > 10`), tiết kiệm tài nguyên.

10) Tóm tắt nhanh các toán tử/stage đã dùng

- **`use("<db>")`**: chọn DB.
- **`insertMany`, `insertOne`**: thêm dữ liệu.
- **`find`**: truy vấn cơ bản (filter/project/sort/limit có thể nối sau).
- **`createUser`**: tạo user với **roles** (đọc/ghi…).
- **Session/Transaction**:
- `startSession()`, `getDatabase(db)`, `startTransaction()`, `commitTransaction()`, `abortTransaction()`.
- **Aggregation pipeline**:
- `$match`: lọc.
- `$group`: nhóm, dùng accumulator (`$sum`, `$avg`, `$max`, `$min`, …); `_id` là key nhóm.
- `$lookup`: join giữa 2 collection theo `localField` ↔ `foreignField`.
- `$unwind`: biến mảng thành nhiều document.
- `$project`: chọn/đặt lại field; có thể dùng biểu thức như `$round`, phép toán.
- `$sort`: sắp xếp.
- Biểu thức: `$multiply`, `$gt`, `$round`.
- **Index**:
- `createIndex({ field: order })` (đơn & compound).
- **Prefix rule** với compound index.
- `explain("executionStats")` để đo IXSCAN/COLLSCAN, số docs/index keys examined.