Giải thích bankDB_session06

27 Tháng Mười 2025 7:16 CH

🧩 1️⃣ Mục đích của Transaction trong MongoDB

Transaction (giao dịch) cho phép **thực hiện nhiều thao tác ghi (insert, update, delete)** như **một đơn vị logic duy nhất**.

→ Nếu **tất cả** thành công → **commit** (xác nhận).
→ Nếu **một thao tác lỗi** → **rollback** (hoàn tác toàn bộ, đưa DB về trạng thái ban đầu).

💡 Giống như trong ngân hàng: nếu trừ tiền người A mà cộng tiền người B thất bại, thì phải hủy toàn bộ để không bị "mất tiền giữa đường".

⚙️ 2️⃣ Transaction thành công — phân tích từng bước

Khởi tạo session

```js
session = db.getMongo().startSession();
```

👉 Mỗi transaction phải chạy trong một **session (phiên làm việc riêng)** để MongoDB có thể theo dõi các thay đổi.

Chọn database sử dụng cho session

```js
bankDB = session.getDatabase("bankDB")
```

👉 Khi dùng transaction, bạn **phải truy cập DB qua session**, chứ không dùng `db.<collection>` thông thường.

Bắt đầu transaction

```js
session.startTransaction()
```

👉 Lệnh này báo cho MongoDB rằng: "Từ giờ trở đi, các thao tác dưới đây sẽ nằm trong cùng một transaction."

Các thao tác trong transaction

```js
bankDB.User.updateOne(
  { name: "Tuan" },
  { $inc: { balance: -1000 } }
)
```

- `$inc` dùng để **tăng/giảm giá trị** (ở đây là trừ tiền người gửi 1000).

```
js

bankDB.User.updateOne(
  { name: "Patrik" },
  { $inc: { balance: 1000 } }
)
```

- Cộng tiền cho người nhận tương ứng.

```
js

bankDB.Notification.insertOne({
  user: "Patrik",
  message: "Ban vua nhan duoc 1000 VND tu Tuan",
  createAt: new Date(),
  read: false
})
```

- Gửi thông báo cho người nhận (dạng "giao dịch thành công").

Commit Transaction

```
js

session.commitTransaction()
print("Transaction commit successfully")
```

- Khi commit, MongoDB sẽ **ghi vĩnh viễn** mọi thay đổi vào cơ sở dữ liệu.
- Nếu không commit, các thay đổi chỉ nằm trong vùng nhớ tạm (in-memory buffer) và sẽ bị hủy khi session kết thúc.

Rollback khi lỗi

```
js

catch (error) {
  session.abortTransaction();
  print("Transaction rollback: " + error)
}
```

- Nếu có lỗi xảy ra trong khối `try`, MongoDB **hoàn tác tất cả thay đổi**.
- Không có ai bị trừ tiền, dữ liệu quay lại như cũ.

Kết thúc session

```
js

finally {
  session.endSession();
}
```

- Đóng session và giải phóng tài nguyên.

🚨 3️⃣ Transaction thất bại — kiểm tra lỗi logic

Phiên bản thứ hai minh họa tình huống **thất bại có điều kiện kiểm tra logic trước khi chuyển tiền**.

Kiểm tra người gửi (sender)

```
js

sender = bankDB.User.findOne({ name: "Tuan" });
if (!sender) throw new Error("Khong tim thay nguoi gui")
```

👉 Nếu không tồn tại user Tuan, dừng lại.

Kiểm tra số dư

```
js

if (sender.balance < 5000) throw new Error("So du khong du")
```

👉 Nếu tiền < 5000, không được chuyển.

Thực hiện trừ tiền người gửi

```
js

bankDB.User.updateOne(
  { name: "Tuan" },
  { $inc: { balance: -5000 } }
)
```

👉 Nếu qua được 2 bước kiểm tra, bắt đầu trừ tiền.

Kiểm tra người nhận (receiver)

```
js

receiver = bankDB.User.findOne({ name: "Matic" });
if (!receiver) throw new Error("Chuyen nham nguoi roi.Tien da duoc chuyen lai cho ban")
```

👉 Nếu không có người nhận hợp lệ, phát sinh lỗi để rollback ngay.

Gửi thông báo sai người (Patrik thay vì Matic)

```
js

bankDB.Notification.insertOne({
  user: "Patrik",
  message: "Ban vua nhan duoc 5000 VND tu Tuan",
  createAt: new Date(),
  read: false
})
```

👉 Đây là **lỗi logic** (vì đáng lẽ phải gửi cho Matic).

Khi commit, có thể gây dữ liệu sai lệch → nếu phát hiện sẽ rollback.

Commit hoặc Rollback

Nếu toàn bộ khối `try` chạy được → commit.

Nếu **bất kỳ dòng nào lỗi** (người gửi không tồn tại, không đủ tiền, người nhận sai) → **abortTransaction** và hoàn tác.

✅ 4️⃣ Kết quả minh họa

Trạng thái	Hành động	Kết quả
Thành công	Giao dịch commit	Tuan bị trừ 1000, Patrik được cộng 1000, thêm thông báo mới
Thất bại (thiếu tiền / sai người)	Transaction rollback	Không ai bị trừ tiền, dữ liệu y nguyên trước đó

💡 Tổng kết cơ chế hoạt động

Bước	Hành động	Ý nghĩa
1	`startSession()`	Mở một phiên transaction
2	`getDatabase()`	Chọn database cần thao tác
3	`startTransaction()`	Báo hiệu bắt đầu giao dịch
4	Thực hiện nhiều thao tác CRUD	Các thao tác tạm thời chưa ghi thật
5	`commitTransaction()`	Ghi vĩnh viễn tất cả thay đổi
6	`abortTransaction()`	Hủy toàn bộ thay đổi khi lỗi
7	`endSession()`	Đóng session

phiên bản có console log chi tiết (in ra từng bước tiến trình giao dịch) để test trực quan trong mongosh.

Dưới đây là **phiên bản đầy đủ, chi tiết có in log từng bước** để bạn dễ theo dõi luồng xử lý transaction trong MongoDB (chạy trong **mongosh**).

Mình chia ra hai phần:

1️⃣ **Transaction thành công**

2️⃣ **Transaction thất bại (rollback)**

🧩 1️⃣ **Transaction Thành Công**

```
// ==============================
// TRANSACTION THÀNH CÔNG
// ==============================
```

```
use("bankDB");

// Kiểm tra dữ liệu hiện tại
print("\n===== TRƯỚC KHI GIAO DỊCH =====");
db.User.find().forEach(printjson);
db.Notification.find().forEach(printjson);

// 1. Bắt đầu session
print("\n[1] Bắt đầu session...");
session = db.getMongo().startSession();

// 2. Chọn database
print("[2] Chọn database bankDB...");
bankDB = session.getDatabase("bankDB");

// 3. Bắt đầu transaction
print("[3] Bắt đầu transaction...");
session.startTransaction();

try {
    print("[4] Trừ tiền người gửi (Tuan -1000)...");
    bankDB.User.updateOne(
        { name: "Tuan" },
        { $inc: { balance: -1000 } }
    );

    print("[5] Cộng tiền người nhận (Patrik +1000)...");
    bankDB.User.updateOne(
        { name: "Patrik" },
        { $inc: { balance: 1000 } }
    );

    print("[6] Gửi thông báo cho người nhận...");
    bankDB.Notification.insertOne({
        user: "Patrik",
        message: "Bạn vừa nhận được 1000 VND từ Tuan",
        createAt: new Date(),
        read: false
    });

    // 7. Commit transaction
    print("[7] Commit transaction...");
    session.commitTransaction();
    print("\n✅ Transaction commit successfully!");
} catch (error) {
    // Nếu có lỗi → rollback
    print("\n❌ Có lỗi xảy ra! Tiến hành rollback...");
    session.abortTransaction();
    print("Chi tiết lỗi: " + error);
} finally {
```

```
    // 8. Kết thúc session
    session.endSession();
    print("\n[8] Kết thúc session");
}

// Xem lại dữ liệu sau khi giao dịch
print("\n===== SAU KHI GIAO DỊCH =====");
db.User.find().forEach(printjson);
db.Notification.find().forEach(printjson);
```

✴️ 2️⃣ Transaction Thất Bại (Rollback)

```
// ================================
// TRANSACTION THẤT BẠI (ROLLBACK)
// ================================

use("bankDB");

// Kiểm tra dữ liệu hiện tại
print("\n===== TRƯỚC KHI GIAO DỊCH =====");
db.User.find().forEach(printjson);
db.Notification.find().forEach(printjson);

// 1. Bắt đầu session
print("\n[1] Bắt đầu session...");
session = db.getMongo().startSession();

// 2. Chọn database
print("[2] Chọn database bankDB...");
bankDB = session.getDatabase("bankDB");

// 3. Bắt đầu transaction
print("[3] Bắt đầu transaction...");
session.startTransaction();

try {
    print("[4] Kiểm tra người gửi (Tuan)...");
    sender = bankDB.User.findOne({ name: "Tuan" });
    if (!sender) {
        throw new Error("Không tìm thấy người gửi!");
    }
    print("→ Người gửi tồn tại, số dư hiện tại: " + sender.balance);

    if (sender.balance < 5000) {
        throw new Error("Số dư không đủ để chuyển tiền!");
    }
```

```
    print("[5] Trừ tiền người gửi (Tuan -5000)...");
    bankDB.User.updateOne(
        { name: "Tuan" },
        { $inc: { balance: -5000 } }
    );

    print("[6] Kiểm tra người nhận (Matic)...");
    receiver = bankDB.User.findOne({ name: "Matic" });
    if (!receiver) {
        throw new Error("Người nhận không tồn tại! Chuyển nhầm tài khoản.");
    }

    print("[7] Cộng tiền người nhận (Matic +5000)...");
    bankDB.User.updateOne(
        { name: "Matic" },
        { $inc: { balance: 5000 } }
    );
    print("[8] Gửi thông báo cho người nhận...");
    bankDB.Notification.insertOne({
        user: "Patrik", // ❌ sai người nhận → ví dụ lỗi logic
        message: "Bạn vừa nhận được 5000 VND từ Tuan",
        createAt: new Date(),
        read: false
    });

    print("[9] Commit transaction...");
    session.commitTransaction();
    print("\n✅ Transaction commit successfully!");
} catch (error) {
    // Nếu có lỗi → rollback
    print("\n❌ Có lỗi xảy ra! Tiến hành rollback...");
    session.abortTransaction();
    print("Chi tiết lỗi: " + error);
} finally {
    // 10. Kết thúc session
    session.endSession();
    print("\n[10] Kết thúc session");
}

// Xem lại dữ liệu sau khi giao dịch (đảm bảo rollback)
print("\n===== SAU KHI GIAO DỊCH (ROLLBACK) =====");
db.User.find().forEach(printjson);
db.Notification.find().forEach(printjson);
```

📊 Giải thích khi test thực tế:

Bước	Hành động	Console log hiển thị
1	Khởi tạo session	`[1] Bắt đầu session...`
2	Bắt đầu transaction	`[3] Bắt đầu transaction...`
3	Thực hiện cập nhật	`[4] Trừ tiền...`, `[5] Cộng tiền...`
4	Commit / Rollback	✅ `Transaction commit successfully!` hoặc ❌ `Có lỗi xảy ra!`
5	Kiểm tra kết quả	Dùng `db.User.find()` và `db.Notification.find()` để thấy thay đổi có/không

💡 Gợi ý để thử:

1️⃣ Chạy phần **thành công trước** → bạn sẽ thấy dữ liệu thay đổi thực.

2️⃣ Chạy phần **thất bại** → intentionally tạo lỗi (ví dụ người nhận `"Matic"` không tồn tại) → MongoDB rollback → dữ liệu không thay đổi.