SQL - TRANSACTIONS

Nguyễn Đình Hóa

dinhhoa@gmail.com

094-280-7711

Tóm tắt các câu lệnh SQL

SELECT	Data retrieval
INSERT UPDATE DELETE	Data manipulation language (DML)
CREATE ALTER DROP TRUNCATE	Data definition language (DDL)
COMMIT ROLLBACK	Transaction control
GRANT REVOKE	Data control language (DCL)

Giao dịch CSDL

- Một giao dịch bao gồm một chuỗi các câu lệnh SQL (được thực hiện bởi một người dùng) liên kết với nhau tạo thành một thể thống nhất về mặt logic.
- Một giao dịch được bắt đầu bằng câu lệnh SQL nào đó.
- Một giao dịch được kết thúc khi tất cả các câu lệnh đều được thực hiện thành công, hoặc bằng thao tác ROLLBACK của người dùng.

Ý nghĩa của giao dịch dữ liệu

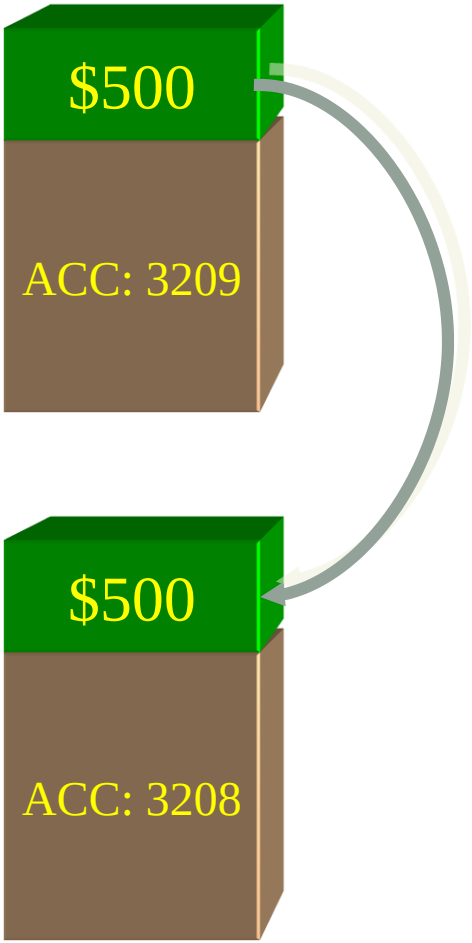
- Giao dịch được dùng để kết hợp các câu lệnh SQL với nhau tạo thành một khối thống nhất mang tính nguyên tố.
- Các câu lệnh trong SQL được thực hiện theo nguyên lý “tất cả hoặc không có gì”.
- Cú pháp của một giao dịch CSDL trong SQL:

```
BEGIN TRANSACTION
        <SQL statements>
COMMIT;
```

Ví dụ

- Trong một hệ CSDL quản lý ngân hàng, khi một khách hàng chuyển tiền từ saving account sang checking account, giao dịch trên CSDL bao gồm các thao tác sau:
  - Trừ tiền trong saving account
  - Cộng tiền trong checking account
  - Lưu thông tin giao dịch.

Ví dụ



A Banking Transaction

```
UPDATE savings_accounts
     SET balance = balance - 500
     WHERE account = 3209;
```

Decrement Savings Account

```
UPDATE checking_accounts
     SET balance = balance + 500
     WHERE account = 3208;
```

Increment Checking Account

```
INSERT INTO journal VALUES
     (journal_seq.NEXTVAL, '1B'
     3209, 3208, 500);
```

Record in Transaction Journal

```
COMMIT WORK;
```

End Transaction

Transaction Ends

Các yêu cầu trong giao dịch

- Hệ quản trị CSDL phải đảm bảo rằng tất cả các câu lệnh SQL trong giao dịch nói trên phải được thực hiện hoàn toàn nhằm đảm bảo số dư tài khoản không bị thay đổi.
- Trong trường hợp một câu lệnh nào đó không được thực hiện (vì một lý do nào đó), tất cả các câu lệnh còn lại không được chấp nhận; trạng thái này gọi là **rolling back**. Nếu có bất kỳ lỗi nào xảy ra trong quá trình cập nhật thông tin thì toàn bộ quá trình cập nhật đó bị huỷ bỏ.

Tính toàn vẹn của giao dịch

- Đây là một trong 4 đặc tính quan trọng của giao dịch dữ liệu (ACID).
- Có 3 kiểu vi phạm tính toàn vẹn của giao dịch (liên quan đến các mức biệt lập khác nhau trong CSDL):
  - Dirty reads
  - Non-Repeatable reads
  - Phantom rows

Các đặc tính của giao dịch dữ liệu

- Các đặc tính này quyết định chất lượng của một hệ quản trị CSDL, bao gồm:
  - Tính nguyên tố (Atomicity)
  - Tính nhất quán (Consistency)
  - Tính biệt lập (Isolation)
  - Tính bền vững (Durability)

Tính nguyên tố

- **Atomicity**: mỗi giao dịch dữ liệu là một thể thống nhất, nghĩa là tất cả các câu lệnh trong đó được thực hiện hoặc không có câu lệnh nào được thực hiện.
  - Nếu chỉ một câu lệnh không được thực hiện thì giao dịch đó sẽ bị huỷ bỏ.
  - Nếu chỉ một phần giao dịch được thực hiện thì tính nguyên tố của giao dịch sẽ bị vi phạm

Tính nhất quán

- **Consistency**: CSDL trước và sau khi diễn ra giao dịch đều phải trong trạng thái nhất quán.
  - Tất cả các bản ghi cũng như các giá trị trên các thuộc tính đều phải tuân theo các ràng buộc đã có.
  - Ví dụ: nếu bản ghi ở bảng gốc được lưu trong CSDL, trong khi bản ghi ở bảng tham chiếu không được lưu thì tính nhất quán không được bảo toàn

Tính biệt lập

- **Isolation**: Mỗi giao dịch CSDL phải biệt lập với các giao dịch khác về mặt câu lệnh cũng như thông tin vào và ra.
  - Mỗi giao dịch làm việc trên bộ dữ liệu đầu vào của nó, bất kể các dữ liệu đó có bị thay đổi trong quá trình giao dịch đang được thực hiện hay không.
  - Ví dụ: Giả sử Huấn đang cập nhật 100 bản ghi, trong khi đó, Hoa xoá một trong số bản ghi mà Huấn đang thao tác trên đó. Giao dịch cập nhật mà Huấn đang thực hiện sẽ không bị ảnh hưởng bởi giao dịch xoá dữ liệu của Hoa.
  - Đặc tính này sẽ thể hiện rõ nét trong các hệ CSDL có nhiều người dùng.

Tính bền vững

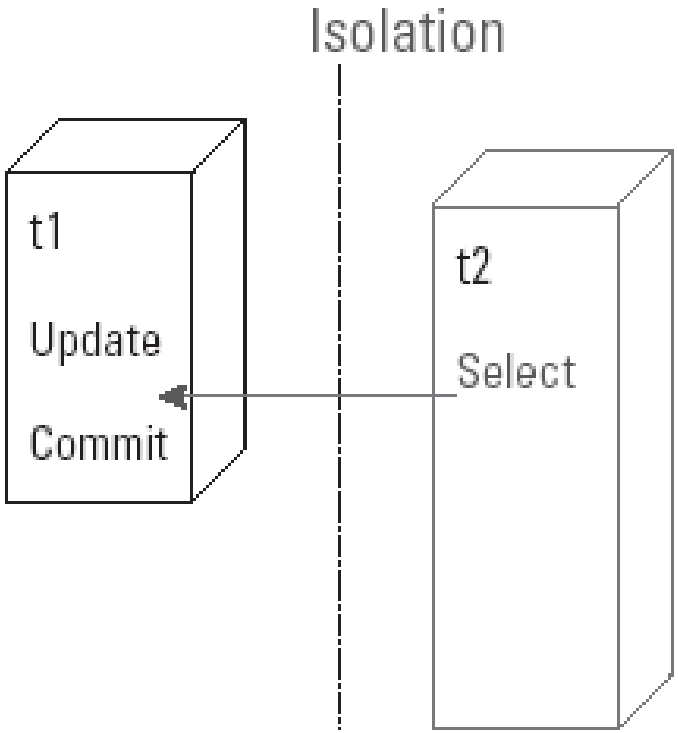
- **Durability**: Một khi giao dịch đã được thực hiện xong, những thay đổi hoặc tác động của nó trên CSDL sẽ được lưu vĩnh viễn.
  - Kể cả khi thiết bị lưu trữ dữ liệu bị hỏng, những kết quả của giao dịch vẫn sẽ được lưu trữ và phục hồi..

Các lỗi thường gặp với giao dịch

Các lỗi sau có thể ảnh hưởng đến tính nhất quán của giao dịch:

- Dirty read (rất nguy hiểm)
- Non-Repeatable reads (nguy hiểm vừa)
- Phantom rows (ít nguy hiểm)

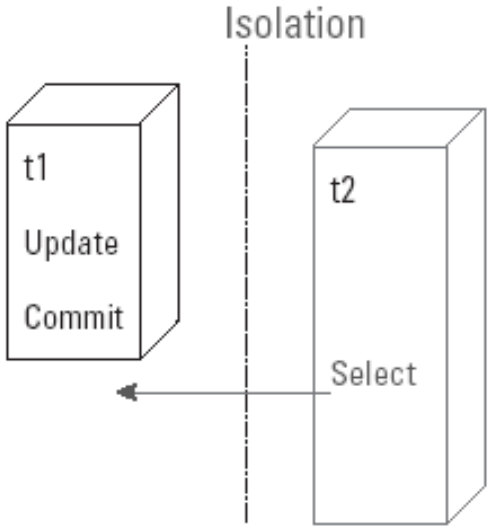
Dirty Read



A dirty read occurs when transaction two can read uncommitted changes made by transaction one.

- Lỗi nghiêm trọng nhất trong giao dịch là khi một giao dịch có thể được "nhìn thấy" bởi giao dịch khác trước khi nó kết thúc
- Khi một giao dịch có thể nhìn thấy quá trình thực hiện của một giao dịch khác đang diễn ra thì được gọi là ***dirty read***.

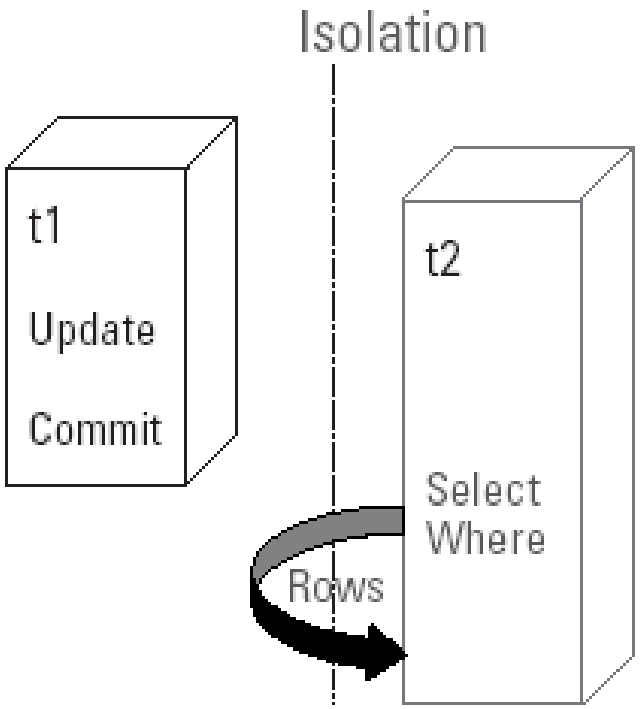
Non-repeatable Read



When transaction one's committed changes are seen by transaction two, that's a *non-repeatable read* transaction fault.

- Lỗi này gần giống với dirty read. Non-repeatable read xảy ra khi một giao dịch có thể thấy được thao tác cập nhật dữ liệu từ một giao dịch khác.
- *Biệt lập hoàn toàn* nghĩa là giao dịch này không làm ảnh hưởng đến giao dịch khác, nghĩa là mọi thay đổi dữ liệu của giao dịch này không làm ảnh hưởng đến các giao dịch bên cạnh
- Thao tác đọc dữ liệu trong một giao dịch luôn phải trả về cùng một kết quả
- Nếu quá trình đọc dữ liệu tạo ra nhiều hơn một kết quả, ta gọi đó là lỗi *non-repeatable read*

Phantom Rows



When the rows returned by a select are altered by another transaction, the phenomenon is called a phantom row.

- Giống với lỗi non-repeatable read, lỗi Phantom Rows xảy ra khi quá trình cập nhật dữ liệu từ một giao dịch khác làm ảnh hưởng tới kết quả của giao dịch, đồng thời làm cho câu lệnh SELECT trả về các bản ghi khác nhau.

Các mức biệt lập giao dịch

- CSDL giải quyết các lỗi trong giao dịch dữ liệu bằng việc thiết lập các mức biệt lập khác nhau
- Các mức biệt lập trong giao dịch được xây dựng nhằm ngăn chặn một, hai, hoặc cả 3 lỗi nêu trên, tuỳ thuộc vào yêu cầu của hệ CSDL.

Các mức biệt lập giao dịch

Table 11-1: ANSI SQL-92 Isolation Levels

Isolation Level	*Dirty Read* Seeing another transaction's non-committed changes	*Non-Repeatable Read* Seeing another transaction's committed changes	*Phantom Row* Seeing rows selected by `where` clause change as a result of another transaction
Read Uncommitted ***(least restrictive)***	Possible	Possible	Possible
Read Committed ***(SQL Server default; moderately restrictive)***	Prevented	Possible	Possible
Repeatable Read	Prevented	Prevented	Possible
Serializable (most restrictive)	Prevented	Prevented	Prevented

Các mức biệt lập giao dịch

Level 1 – Read Uncommitted

The least restrictive isolation level is *read uncommitted*, which doesn't prevent any of the transactional faults. It's like having no fence at all because it provides no isolation between transactions. Setting the isolation level to read uncommitted is the same as setting SQL Server's locks to no locks. This mode is best for reporting and data-reading applications. Because this mode has just enough locks to prevent data corruption, but not enough to handle row contention, it's not very useful for databases whose data is updated regularly.

Level 2 – Read Committed

Read committed prevents the worst transactional fault, but doesn't bog the system down with excessive lock contention. For this reason, it's the SQL Server default isolation level and an ideal choice for most OTLP projects.

Các mức biệt lập giao dịch

Level 3 — Repeatable Read

By preventing dirty reads and non-repeatable reads, the *repeatable read* isolation level provides an increase in transaction isolation without the extreme lock contention of serializable isolation.

Level 4 — Serializable

This most restrictive isolation level prevents all transactional faults and passes the serialized-transaction test mentioned in the definition of isolation. This mode is useful for databases for which absolute transactional integrity is more important than performance. Banking, accounting, and high-contention sales databases, such as the stock market, typically use serialized isolation.

Using the serializable isolation level is the same as setting locks to hold locks, which holds even share locks for the length of the transaction. While this setting provides absolute transaction isolation, it can cause serious lock contention and performance delays.

Các mức biệt lập giao dịch

- Lệnh cài đặt:

  SET TRANSACTION ISOLATION LEVEL <level>

  where <level> is replaced with any of the following keywords:
  - READ COMMITTED
  - READ UNCOMMITTED
  - REPEATABLE READ
  - SERIALIZABLE
  - SNAPSHOT

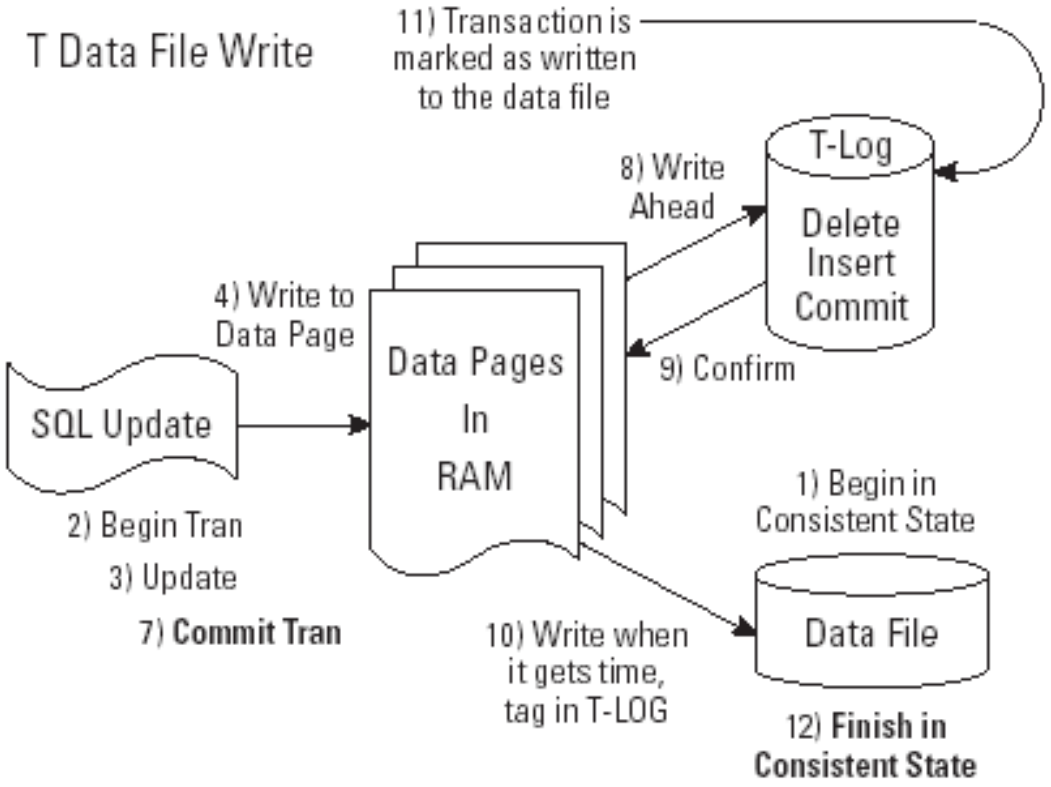
COMMIT và ROLLBACK

- **Committing** a transaction makes permanent the changes resulting from all SQL statements in the transaction. The changes made by the SQL statements of a transaction become visible to other user sessions' transactions that start only after the transaction is committed.
- **Rolling back** a transaction retracts any of the changes resulting from the SQL statements in the transaction. After a transaction is rolled back, the affected data is left unchanged as if the SQL statements in the transaction were never executed.

Ví dụ

```
BEGIN TRANSACTION
  INSERT GoldInventory (InventoryID, Location, Quantity)
    VALUES (101, 'Vault A', -2000)
  IF @@error <> 0
    BEGIN
      ROLLBACK TRANSACTION
      RAISERROR('There was an error', 16, 1)
      RETURN
    END
  INSERT GoldInventory (InventoryID, Location, Quantity)
    VALUES (101, 'Vault 12', 2000)
  IF @@error <> 0
    BEGIN
      ROLLBACK TRANSACTION
      RAISERROR('There was an error', 16, 1)
      RETURN
    END
COMMIT TRANSACTION
```

Ví dụ



As one of the last steps, the data modification is written to the data file.

- When the sequence of tasks is complete, the COMMIT TRANSACTION closes the transaction

Savepoints

- For long transactions that contain many SQL statements, intermediate markers, or **savepoints**, can be declared. Savepoints can be used to divide a transaction into smaller parts.

- By using savepoints, you can arbitrarily mark your work at any point within a long transaction. This gives you the option of later rolling back all work performed from the current point in the transaction to a declared savepoint within the transaction.

- For example, you can use savepoints throughout a long complex series of updates, so if you make an error, you do not need to resubmit every statement.