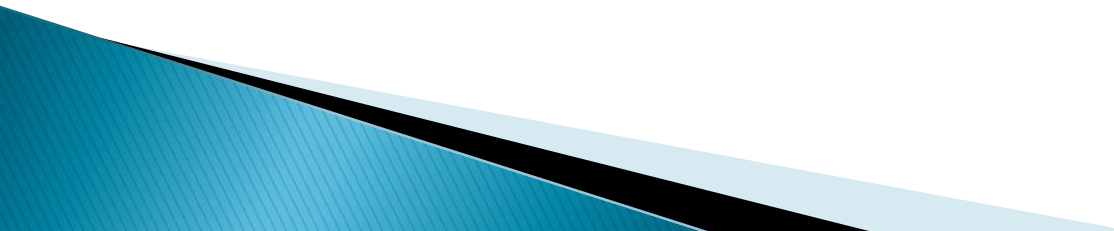


TRANSACTION & LOCK

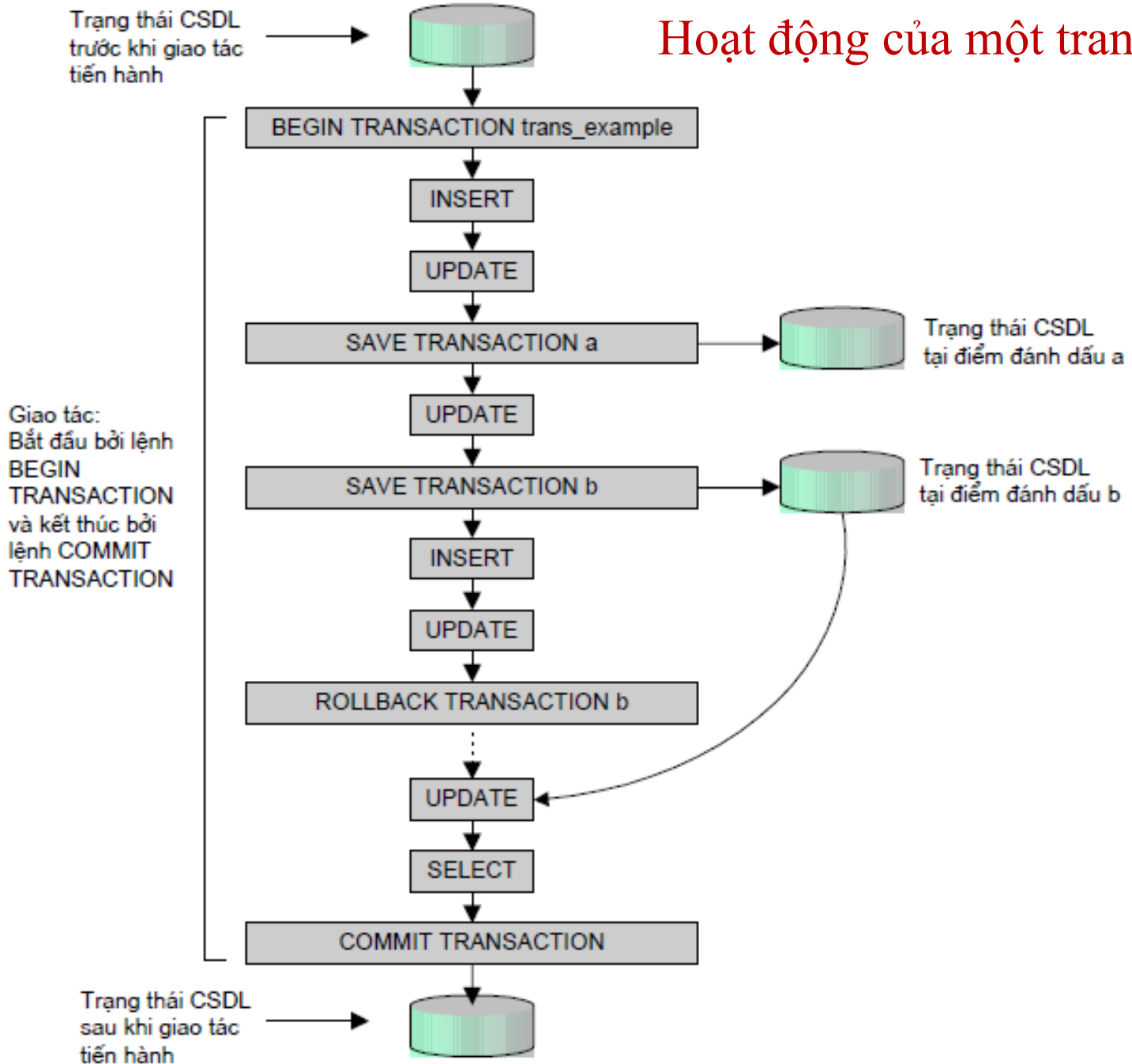
Trong SQL server

TRANSACTION

Transaction trong SQL

- ▶ **BEGIN TRANSACTION:** Bắt đầu một giao tác
 - ▶ **SAVE TRANSACTION:** Đánh dấu một vị trí trong giao tác (gọi là điểm đánh dấu).
 - ▶ **ROLLBACK TRANSACTION:** Quay lui trở lại đầu giao tác hoặc một điểm đánh dấu trước đó trong giao tác.
 - ▶ **COMMIT TRANSACTION:** Đánh dấu điểm kết thúc một giao tác. Khi câu lệnh này thực thi cũng có nghĩa là giao tác đã thực hiện thành công.
- 

Hoạt động của một transaction



Transaction trong SQL - Cú Pháp

BEGIN TRANSACTION

SQL Statements

COMMIT | ROLLBACK TRANSACTION

Chúng ta có thể sử dụng TRY...CATCH cùng với TRANSACTION

BEGIN TRY

BEGIN TRAN

-- Code for your transaction

COMMIT TRAN

END TRY

BEGIN CATCH

-- output an error message

ROLLBACK TRAN

END CATCH

Các biến và hàm hệ thống liên quan đến transaction

- ▶ @@trancount: Số lượng transaction.
- ▶ @@error <0: Xảy ra lỗi
- ▶ XACT_STATE(): Cho biết trạng thái transaction hiện thời

Transaction trong SQL - Ví dụ

- ▶ Giao tác dưới đây kết thúc do lệnh ROLLBACK TRANSACTION
→ Mọi thay đổi về mặt dữ liệu mà giao tác đã thực hiện (UPDATE) đều không có tác dụng.

```
use sample
BEGIN TRANSACTION giaotac1
UPDATE employee SET emp_fname='John' WHERE emp_no=3186
UPDATE works_on SET job='Clerk' WHERE job IS NULL
ROLLBACK TRANSACTION giaotac1
```

Transaction trong SQL - Ví dụ

- ▶ Giao tác dưới đây kết thúc bởi lệnh COMMIT và thực hiện thành công việc cập nhật dữ liệu trên các bảng employee và works_on.

```
use sample  
BEGIN TRANSACTION giaotac1  
UPDATE employee SET emp_fname='John' WHERE emp_no=1234  
UPDATE works_on SET job='Clerk' WHERE job IS NULL  
COMMIT TRANSACTION giaotac1
```


Transaction trong SQL - Ví dụ

```
Begin try
    begin tran
        Update project
        set budget =100 where project_no in ('p3','p7')
        update works_on
        set enter_date =enter_date+100
        where project_no in ('p3','p7')
    Commit tran
end try
begin catch
    rollback tran
    raiserror ('Transaction Error',16,1)
end catch
```

Ví dụ

```
USE sample;  
BEGIN TRANSACTION /* The beginning of the transaction */  
    UPDATE employee  
    SET emp_no = 39831  
    WHERE emp_no = 10102  
    IF (@@error <> 0)  
        ROLLBACK /* Rollback of the transaction */  
    UPDATE works_on  
    SET emp_no = 39831  
    WHERE emp_no = 10102  
    IF (@@error <> 0)  
        ROLLBACK  
COMMIT /*The end of the transaction */
```

Ví dụ sử dụng SAVE TRANSACTION

```
use SAMPLE;  
BEGIN TRANSACTION;  
    INSERT INTO department (dept_no, dept_name)  
        VALUES ('d4', 'Sales');  
SAVE TRANSACTION a;  
INSERT INTO department (dept_no, dept_name)  
    VALUES ('d5', 'Research');  
SAVE TRANSACTION b;  
INSERT INTO department (dept_no, dept_name)  
    VALUES ('d6', 'Management');  
ROLLBACK TRANSACTION b;  
INSERT INTO department (dept_no, dept_name)  
    VALUES ('d7', 'Support');  
ROLLBACK TRANSACTION a;  
COMMIT TRANSACTION;
```

Ví dụ:

1. Mở cửa sổ query 1, định nghĩa transaction sau

use SAMPLE

BEGIN TRAN

SELECT 'The current number of open transactions is:', @@trancount

SELECT 'The current last name of customer number 10201 is:', emp_lname

FROM employee

WHERE emp_no = 10201

UPDATE employee

SET emp_lname = 'Johnson'

WHERE emp_no = 10201

SELECT 'After UPDATE statement, the last name of customer 10201 is:', emp_lname

FROM employee

WHERE emp_no = 10201

2. Quan sát kết quả

	(No column name)	(No column name)
1	The current number of open transactions is:	1

	(No column name)	emp_lname
1	The current last name of customer number 10201 is:	nguyen

	(No column name)	emp_lname
1	After the UPDATE statement,the last name of customer 10201 is:	Johnson

3. Mở cửa sổ query thứ 2, thực hiện lệnh

```
SELECT 'the last name of customer 10201 is: ', emp_lname  
FROM employee  
WHERE emp_no = 10201
```



Không thực hiện được

4. Trở về cửa sổ query 1, thực hiện lệnh

ROLLBACK TRAN

```
SELECT 'After roll back, the last name of customer 1022 is: ', emp_lname  
FROM employee
```

```
WHERE emp_no = 10201
```

```
SELECT 'The current number of open transactions is: ', @@trancount
```



Results



Messages

	(No column name)	emp_lname
1	After the roll back statement, the last name of customer 1022 is:	nguyen
	(No column name)	(No column name)
1	The current number of open transactions is:	0

Các loại Transaction

- ▶ Implicit Transaction (Không tường minh)
 - Tự động bắt đầu một transaction cho mỗi câu lệnh SQL
 - Kết thúc bằng
 - COMMIT TRAN / COMMIT
 - ROLLBACK TRAN
- ▶ Explicit Transaction (Tường minh)
 - Bắt đầu bằng BEGIN TRANSACTION / BEGIN TRAN
 - Kết thúc bằng
 - COMMIT TRAN / COMMIT
 - ROLLBACK TRAN
- ▶ Auto commit transaction - Giao tác tự động COMMIT

Bật chế độ implicit transaction

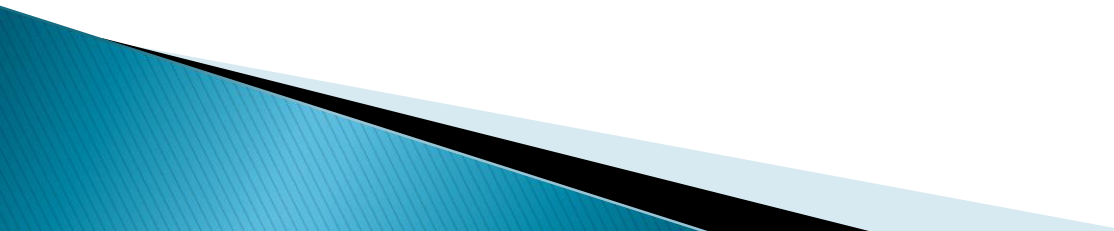
SET IMPLICIT_TRANSACTIONS ON

- ▶ Bắt đầu một transaction bất cứ khi nào thực hiện các lệnh sau:

ALTER TABLE	FETCH	REVOKE
CREATE	GRANT	SELECT
DELETE	INSERT	TRUNCATE TABLE
DROP	OPEN	UPDATE

- Không tự động commit cho tới khi gặp các lệnh sau:
 - COMMIT TRAN
 - ROLLBACK TRAN.

Autocommit Transaction

- ▶ Autocommit Transaction: Mặc định của SQL Server.
 - ▶ Một lệnh (statement) được committed nếu nó thực hiện thành công hay sẽ trả ngược lại ban đầu (roll back) nếu nó gặp lỗi.
 - ▶ Lệnh BEGIN TRANSACTION sẽ vượt quyền autocommit mặc định.
 - ▶ SQL Server trở về lại mode autocommit khi transaction tường mình đã được commit hay roll back hay khi mode implicit Transaction bị tắt.
- 

TRANSACTION LOG

- ▶ Dùng để lưu vết tất cả các giao dịch
- ▶ Phục hồi dữ liệu
- ▶ Một transaction log gồm:
 - Một record đánh dấu bắt đầu 1 transaction
 - Thông tin về transaction
 - Thao tác (cập nhật, xóa, chèn)
 - Tên các object ảnh hưởng bởi transaction
 - Giá trị trước và sau của các field được cập nhật.
 - Con trỏ trỏ đến dòng trước và sau trong cùng 1 transaction
 - Một record đánh dấu kết thúc transaction

TRANSACTION LOG (tt)

TABLE 9.1 A TRANSACTION LOG

TRL ID	TRX NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	****Start Transaction				
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_BALANCE	525.75	615.73
365	101	363	Null	COMMIT	**** End of Transaction				



TRL_ID = Transaction log record ID

PTR = Pointer to a transaction log record ID

TRX_NUM = Transaction number

(Note: The transaction number is automatically assigned by the DBMS.)

LOCKS & ISOLATION LEVEL

Các phương thức khóa cơ bản

- ▶ Shared Locks
- ▶ Exclusive Locks

Shared Locks (S)

- ▶ Shared Lock \Leftrightarrow Read Lock
- ▶ Khi đọc 1 đơn vị dữ liệu, SQL Server tự động thiết lập Shared Lock trên đơn vị dữ liệu đó
- ▶ Shared Lock có thể được thiết lập trên 1 bảng, 1 trang hay trên 1 dòng dữ liệu.
- ▶ Nhiều giao tác có thể đồng thời giữ Shared Lock trên cùng 1 đơn vị dữ liệu.
- ▶ Shared Lock thường được giải phóng ngay sau khi sử dụng xong dữ liệu được đọc, trừ khi có thiết lập giữ shared lock cho đến hết giao tác.

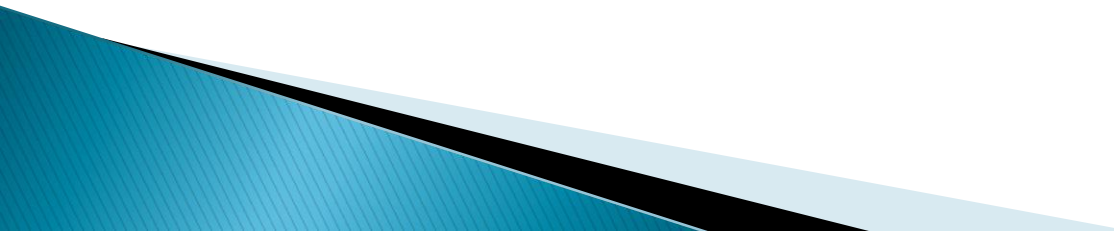
Exclusive Locks (X)

- ▶ Exclusive Lock \Leftrightarrow Write Lock
- ▶ Khi thực hiện thao tác ghi (insert, update, delete) trên 1 đơn vị dữ liệu, SQL Server tự động thiết lập Exclusive Lock trên đơn vị dữ liệu đó.
- ▶ Exclusive Lock luôn được giữ đến hết giao tác.
- ▶ Tại 1 thời điểm, chỉ có tối đa 1 giao tác được quyền giữ Exclusive Lock trên 1 đơn vị dữ liệu.
- ▶ Không thể thiết lập Exclusive Lock trên đơn vị dữ liệu đang có Shared Lock.

ISOLATION LEVEL - Mức cô lập

- ▶ SQL Sever có một số lựa chọn để chỉ định cách nó khóa các đơn vị dữ liệu đó là thiết lập các mức độ cô lập dữ liệu
- ▶ Mức độ cô lập của giao tác quy định thời gian giữ lock: lock có cần được giữ cho đến hết giao tác để ngăn những sự thay đổi trên CSDL do các giao tác khác tạo ra hay không

Các mức cô lập

- ▶ Read Uncommitted
 - ▶ Read Committed
 - ▶ Repeatable Read
 - ▶ Serializable
- 

Câu lệnh thiết lập mức độ cô lập

SET TRANSACTION ISOLATION LEVEL

< READ COMMITTED |
 READ UNCOMMITTED |
 REPEATABLE READ |
 SERIALIZABLE >

Read Uncommitted

► Đặc điểm:

- Không thiết lập **Shared Lock** trên những đơn vị dữ liệu cần đọc. Do đó không phải chờ khi đọc dữ liệu (kể cả khi dữ liệu đang bị lock bởi giao tác khác)
- Vẫn tạo Exclusive Lock trên đơn vị dữ liệu cần ghi, Exclusive Lock được giữ cho đến hết giao tác

► Ưu điểm:

- Tốc độ xử lý rất nhanh
- Không cản trở những giao tác khác thực hiện việc cập nhật dữ liệu

► Khuyết điểm:

- Có khả năng xảy ra mọi vấn đề khi xử lý đồng thời :
 - Dirty Reads
 - Unrepeatable Reads
 - Phantoms
 - Lost Updates

Read Committed

► Đặc điểm:

- Đây là mức độ cô lập mặc định của SQL Server
- Tạo Shared Lock trên đơn vị dữ liệu được đọc, **Shared Lock được giải phóng ngay sau khi đọc xong dữ liệu**
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác
- Ưu điểm:
 - Giải quyết vấn đề Dirty Reads
 - Shared Lock được giải phóng ngay, không cần phải giữ cho đến hết giao tác nên không cản trở nhiều đến thao tác cập nhật của các giao tác khác.
- Nhược điểm:
 - Chưa giải quyết được vấn đề
 - Unrepeatable Reads,
 - Phantoms,
 - Lost Updates
 - Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi (xlock)

Repeatable Read

► Đặc điểm:

- Tạo Shared Lock trên đơn vị dữ liệu được đọc và **giữ shared lock này đến hết giao tác** => Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này .
- Repeatable Read = Read Committed + Giải quyết Unrepeatable Reads
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

► Ưu điểm:

- Giải quyết vấn đề Dirty Reads và Unrepeatable Reads

► Nhược điểm:

- Chưa giải quyết được vấn đề Phantoms, do vẫn cho phép insert những dòng dữ liệu thỏa điều kiện thiết lập shared lock
- Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi (xlock)
- Shared lock được giữ đến hết giao tác ==> cản trở việc cập nhật dữ liệu của các giao tác khác

Serializable

► Đặc điểm:

- Tạo Shared Lock trên đơn vị dữ liệu được đọc và giữ shared lock này đến hết giao tác => Các giao tác khác phải chờ đến khi giao tác này kết thúc nếu muốn cập nhật, thay đổi giá trị trên đơn vị dữ liệu này .
- **Không cho phép Insert** những dòng dữ liệu thỏa mãn điều kiện thiết lập Shared Lock
- Serializable = Repeatable Read + Giải quyết Phantoms
- Tạo Exclusive Lock trên đơn vị dữ liệu được ghi, Exclusive Lock được giữ cho đến hết giao tác.

► Ưu điểm:

- Giải quyết thêm được vấn đề Phantoms

► Nhược điểm:

- Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi (xlock)
- Cản trở nhiều đến việc cập nhật dữ liệu của các giao tác khác