

# Transactions

TRANSACTIONS AND ERROR HANDLING IN SQL SERVER



**Miriam Antona**  
Software Engineer

# Dataset: bank transactions

customers

customer_id	first_name	last_name	email	phone
-----	-----	-----	-----	-----
1	Dylan	Smith	dylansmith@mail.com	555888999
2	John	Antona	johnantona@mail.com	555111222
3	Astrid	Harper	astridharper@mail.com	555000999
4	Angus	Brown	angusbrown@mail.com	555222012
5	David	Elcano	davideelcano@mail.com	555602314

# Dataset: bank transactions

accounts

account_id	account_number	customer_id	current_balance
1	55555555551234567890	1	25000,00
2	55555555559876543210	1	200,00
3	55555555557070700707	2	1000,00
4	55555555558080808080	2	90000,00
5	55555555559090909090	3	35000,00

# Dataset: bank transactions

transactions

transaction_id	account_id	amount	transaction_date
-----	-----	-----	-----
1	1	-100,00	2019-03-18 19:12:36.81
2	2	100,00	2019-01-18 19:12:36.91
3	1	-9000,00	2019-02-18 20:20:36.41
4	3	9000,00	2019-02-18 20:20:36.51
5	4	-50,00	2019-02-20 08:02:06.20

# What is a transaction?

- **Transaction:** one or more statements, all or none of the statements are executed

# What is a transaction?

**Transfer \$100 account A -> account B**

1. Subtract \$100 from account A
2. Add \$100 to account B

*Operation 2 FAILS -> Can't subtract \$100 from account A!*

# Transaction statements - BEGIN a transaction

```
BEGIN { TRAN | TRANSACTION }  
    [ { transaction_name | @tran_name_variable }  
      [ WITH MARK [ 'description' ] ]  
    ]  
[ ; ]
```

# Transaction statements - COMMIT a transaction

```
COMMIT [ { TRAN | TRANSACTION } [ transaction_name | tran_name_variable] ]  
      [ WITH ( DELAYED_DURABILITY = { OFF | ON } ) ][ ; ]
```



# Transaction statements - ROLLBACK a transaction

```
ROLLBACK { TRAN | TRANSACTION }  
    [ transaction_name | @tran_name_variable |  
      savepoint_name | @savepoint_variable ]    [ ; ]
```

# Transaction - example

- Account 1 = \$24,400
- Account 5 = \$35,300

```
BEGIN TRAN;  
    UPDATE accounts SET current_balance = current_balance - 100 WHERE account_id = 1;  
    INSERT INTO transactions VALUES (1, -100, GETDATE());  
  
    UPDATE accounts SET current_balance = current_balance + 100 WHERE account_id = 5;  
    INSERT INTO transactions VALUES (5, 100, GETDATE());  
COMMIT TRAN;
```

# Transaction - example

- Account 1 = \$24,400
- Account 5 = \$35,300

account_id	account_number	customer_id	current_balance
1	55555555551234567890	1	24300,00
5	55555555559090909090	3	35400,00

transaction_id	account_id	amount	transaction_date
10	5	100,00	2019-06-07 18:26:27.46
19	1	-100,00	2019-06-07 18:28:05.49

# Transaction - example

- Account 1 = \$24,400
- Account 5 = \$35,300

```
BEGIN TRAN;  
    UPDATE accounts SET current_balance = current_balance - 100 WHERE account_id = 1;  
    INSERT INTO transactions VALUES (1, -100, GETDATE());  
  
    UPDATE accounts SET current_balance = current_balance + 100 WHERE account_id = 5;  
    INSERT INTO transactions VALUES (5, 100, GETDATE());  
ROLLBACK TRAN;
```

# Transaction - example

- Account 1 = \$24,400
- Account 5 = \$35,300

account_id	account_number	customer_id	current_balance
1	55555555551234567890	1	24400,00
5	55555555559090909090	3	35300,00

# Transaction - example with TRY...CATCH

- Account 1 = \$24,400
- Account 5 = \$35,300

```
BEGIN TRY
    BEGIN TRAN;
        UPDATE accounts SET current_balance = current_balance - 100 WHERE account_id = 1;
        INSERT INTO transactions VALUES (1, -100, GETDATE());

        UPDATE accounts SET current_balance = current_balance + 100 WHERE account_id = 5;
        INSERT INTO transactions VALUES (5, 100, GETDATE());
    COMMIT TRAN;
END TRY
BEGIN CATCH
    ROLLBACK TRAN;
END CATCH
```

# Transaction - example with TRY...CATCH

- Account 1 = \$24,400
- Account 5 = \$35,300

account_id	account_number	customer_id	current_balance
1	55555555551234567890	1	24300,00
5	55555555559090909090	3	35400,00

transaction_id	account_id	amount	transaction_date
10	5	100,00	2019-06-07 18:26:27.46
19	1	-100,00	2019-06-07 18:28:05.49

# Transaction - example with TRY..CATCH

- Account 1 = \$24,400
- Account 5 = \$35,300

```
BEGIN TRY
  BEGIN TRAN;
    UPDATE accounts SET current_balance = current_balance - 100 WHERE account_id = 1;
    INSERT INTO transactions VALUES (1, -100, GETDATE());

    UPDATE accounts SET current_balance = current_balance + 100 WHERE account_id = 5;
    INSERT INTO transactions VALUES (500, 100, GETDATE()); -- ERROR!
  COMMIT TRAN;
END TRY
BEGIN CATCH
  ROLLBACK TRAN;
END CATCH
```



# Transaction - example with TRY...CATCH

- Account 1 = \$24,400
- Account 5 = \$35,300

account_id	account_number	customer_id	current_balance
1	55555555551234567890	1	24400,00
5	55555555559090909090	3	35300,00

# Transaction - without specifying a transaction

- Account 1 = \$24,400
- Account 5 = \$35,300

```
UPDATE accounts SET current_balance = current_balance - 100 WHERE account_id = 1;  
INSERT INTO transactions VALUES (1, -100, GETDATE());
```

```
UPDATE accounts SET current_balance = current_balance + 100 WHERE account_id = 5;  
INSERT INTO transactions VALUES (500, 100, GETDATE()); -- ERROR!
```

# Transaction - without specifying a transaction

- Account 1 = \$24,400
- Account 5 = \$35,300

account_id	account_number	customer_id	current_balance
1	55555555551234567890	1	24300,00
5	55555555559090909090	3	35400,00

transaction_id	account_id	amount	transaction_date
10	5	100,00	2019-06-07 18:26:27.46

# Let's practice!

TRANSACTIONS AND ERROR HANDLING IN SQL SERVER

# @@TRANCOUNT and savepoints

TRANSACTIONS AND ERROR HANDLING IN SQL SERVER



**Miriam Antona**  
Software Engineer

# @@TRANCOUNT

Number of BEGIN TRAN statements that are active in your current connection.

## Returns:

- greater than 0 -> open transaction
- 0 -> no open transaction

## Modified by:

- BEGIN TRAN -> @@TRANCOUNT + 1
- COMMIT TRAN -> @@TRANCOUNT - 1
- ROLLBACK TRAN -> @@TRANCOUNT = 0 (except with savepoint\_name)

# Nested transactions

```
SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
BEGIN TRAN;
    SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
    DELETE transactions;
    BEGIN TRAN;
        SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
        DELETE accounts;
        -- If @@TRANCOUNT > 1 it doesn't commit!
    COMMIT TRAN;
    SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
ROLLBACK TRAN;
SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
```

```
| @@TRANCOUNT value |
|-----|
| 0                  |
```

```
| @@TRANCOUNT value |
|-----|
| 1                  |
```

```
| @@TRANCOUNT value |
|-----|
| 2                  |
```

```
| @@TRANCOUNT value |
|-----|
| 1                  |
```

```
| @@TRANCOUNT value |
|-----|
| 0                  |
```

# Nested transactions

```
SELECT * FROM transactions
```

transaction_id	account_id	amount	transaction_date
1	1	-100,00	2019-03-18 19:12:36.81
2	2	100,00	2019-01-18 19:12:36.91
...	...	...	...

```
SELECT * FROM accounts
```

account_id	account_number	customer_id	current_balance
1	5555555551234567890	1	25000,00
2	5555555559876543210	1	200,00
...	...	...	...



# Nested transactions

```
SELECT @@TRANCOUNT AS '@@TRANCOUNT value';  
BEGIN TRAN;  
    SELECT @@TRANCOUNT AS '@@TRANCOUNT value';  
    DELETE transactions;  
    BEGIN TRAN;  
        SELECT @@TRANCOUNT AS '@@TRANCOUNT value';  
        DELETE accounts;  
    COMMIT TRAN;  
    SELECT @@TRANCOUNT AS '@@TRANCOUNT value';  
COMMIT TRAN;  
SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
```

# Nested transactions

```
SELECT * FROM transactions
```

```
| transaction_id | account_id | amount | transaction_date |
|-----|-----|-----|-----|
```

```
SELECT * FROM accounts
```

```
| account_id | account_number | customer_id | current_balance |
|-----|-----|-----|-----|
```

# @@TRANCOUNT in a TRY...CATCH construct

```
BEGIN TRY
    BEGIN TRAN;
        UPDATE accounts SET current_balance = current_balance - 100 WHERE account_id = 1
        INSERT INTO transactions VALUES (1, -100, GETDATE());

        UPDATE accounts SET current_balance = current_balance + 100 WHERE account_id = 5
        INSERT INTO transactions VALUES (5, 100, GETDATE());
    IF (@@TRANCOUNT > 0)
        COMMIT TRAN;
END TRY
BEGIN CATCH
    IF (@@TRANCOUNT > 0)
        ROLLBACK TRAN;
END CATCH
```

# Savepoints

- Markers within a transaction
- Allow to rollback to the savepoints

```
SAVE { TRAN | TRANSACTION } { savepoint_name | @savepoint_variable }  
[ ; ]
```

# Savepoints

```
BEGIN TRAN;
  SAVE TRAN savepoint1;
  INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');

  SAVE TRAN savepoint2;
  INSERT INTO customers VALUES ('Zack', 'Roberts', 'zackroberts@mail.com', '555919191');

  ROLLBACK TRAN savepoint2;
  ROLLBACK TRAN savepoint1;

  SAVE TRAN savepoint3;
  INSERT INTO customers VALUES ('Jeremy', 'Johnsson', 'jeremyjohnsson@mail.com', '555929292');
COMMIT TRAN;
```

customer_id	first_name	last_name	email	phone
13	Jeremy	Johnsson	jeremyjohnsson@mail.com	555929292

# Savepoints

```
BEGIN TRAN
...
ROLLBACK TRAN savepoint2;
SELECT @@TRANCOUNT AS '@@TRANCOUNT value';

ROLLBACK TRAN savepoint1;
SELECT @@TRANCOUNT AS '@@TRANCOUNT value';
...
COMMIT TRAN;
```

```
| @@TRANCOUNT value |
|-----|
| 1                |
```

```
| @@TRANCOUNT value |
|-----|
| 1                |
```

# Let's practice!

TRANSACTIONS AND ERROR HANDLING IN SQL SERVER

# XACT\_ABORT & XACT\_STATE

TRANSACTIONS AND ERROR HANDLING IN SQL SERVER



**Miriam Antona**  
Software Engineer



# XACT\_ABORT

Specifies whether the current transaction will be automatically rolled back when an error occurs.

```
SET XACT_ABORT { ON | OFF }
```

```
SET XACT_ABORT OFF
```

- Default setting
- If there is an error: There can be open transactions

```
SET XACT_ABORT ON
```

- If there is an error: Rollbacks the transaction and aborts the execution

# XACT\_ABORT - examples

```
SET XACT_ABORT OFF; --Default setting
```

```
BEGIN TRAN;
```

```
    INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');
```

```
    INSERT INTO customers VALUES ('Dylan', 'Smith', 'dylansmith@mail.com', '555888999'); -- ERROR!
```

```
COMMIT TRAN;
```

```
(1 row affected)
```

```
Msg. 2627, Level 14, State 1, Line 5
```

```
Violation of UNIQUE KEY 'unique_email'...
```

customer_id	first_name	last_name	email	phone
14	Mark	Davis	markdavis@mail.com	555909090

# XACT\_ABORT - examples

```
SET XACT_ABORT ON;

BEGIN TRAN;
    INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');
    INSERT INTO customers VALUES ('Dylan', 'Smith', 'dylansmith@mail.com', '555888999'); -- ERROR!
COMMIT TRAN;
```

```
Msg. 2627, Level 14, State 1, Line 4
Violation of UNIQUE KEY 'unique_email'...
```

```
SELECT * FROM customers WHERE first_name = 'Mark';
```

customer_id	first_name	last_name	email	phone
-----	-----	-----	-----	-----

# XACT\_ABORT WITH RAISERROR

```
SET XACT_ABORT ON;  
BEGIN TRAN;  
    INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');  
    RAISERROR('Raising an error!', 16, 1);  
    INSERT INTO customers VALUES ('Zack', 'Roberts', 'zackroberts@mail.com', '555919191');  
COMMIT TRAN;
```

```
Msg. 50000, Level 16, State 1, Line 5  
Raising an error!
```

```
SELECT * FROM customers WHERE first_name IN ('Mark', 'Zack');
```

customer_id	first_name	last_name	email	phone
14	Mark	Davis	markdavis@mail.com	555909090
15	Zack	Roberts	zackroberts@mail.com	555919191

# XACT\_ABORT with THROW

```
SET XACT_ABORT ON;  
BEGIN TRAN;  
    INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');  
    THROW 55000, 'Raising an error!', 1;  
    INSERT INTO customers VALUES ('Zack', 'Roberts', 'zackroberts@mail.com', '555919191');  
COMMIT TRAN;
```

```
(1 rows affected)  
Msg. 50000, Level 16, State 1, Line 5  
Raising an error!
```

```
SELECT * FROM customers WHERE first_name IN ('Mark', 'Zack');
```

```
| customer_id | first_name | last_name | email | phone |  
|-----|-----|-----|-----|-----|
```

# XACT\_STATE

XACT\_STATE()

- **0** -> no open transaction
- **1** -> open and committable transaction
- **-1** -> open and uncommittable transaction (doomed transaction)
  - can't commit
  - can't rollback to a savepoint
  - can rollback the full transaction
  - can't make any changes/can read data

# XACT\_STATE - open and committable

```
SET XACT_ABORT OFF;
BEGIN TRY
    BEGIN TRAN;
        INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');
        INSERT INTO customers VALUES ('Dylan', 'Smith', 'dylansmith@mail.com', '555888999'); -- ERROR!
    COMMIT TRAN;
END TRY
BEGIN CATCH
    IF XACT_STATE() = -1
        ROLLBACK TRAN;
    IF XACT_STATE() = 1
        COMMIT TRAN;
    SELECT ERROR_MESSAGE() AS error_message;
END CATCH
```

```
| error_message |
|-----|
| Violation of UNIQUE KEY 'unique_email'...
```

# XACT\_STATE - open and committable

customer_id	first_name	last_name	email	phone
14	Mark	Davis	markdavis@mail.com	555909090



# XACT\_STATE - open and uncommittable (doomed)

```
SET XACT_ABORT ON;
BEGIN TRY
    BEGIN TRAN;
        INSERT INTO customers VALUES ('Mark', 'Davis', 'markdavis@mail.com', '555909090');
        INSERT INTO customers VALUES ('Dylan', 'Smith', 'dylansmith@mail.com', '555888999'); -- ERROR!
    COMMIT TRAN;
END TRY
BEGIN CATCH
    IF XACT_STATE() = -1
        ROLLBACK TRAN;
    IF XACT_STATE() = 1
        COMMIT TRAN;
    SELECT ERROR_MESSAGE() AS Error_message;
END CATCH
```

# XACT\_STATE - open and uncommittable (doomed)

```
SELECT * FROM customers WHERE first_name = 'Mark';
```

customer_id	first_name	last_name	email	phone
-----	-----	-----	-----	-----

# Let's practice!

TRANSACTIONS AND ERROR HANDLING IN SQL SERVER