

Triggers and Transactions

Database Programmability



SoftUni Team
Technical Trainers



SoftUni



Software University

<https://about.softuni.bg/>

sli.do

#csharp-db

Table of Contents

1. Transactions
2. ACID Model
3. Triggers
4. Database Security



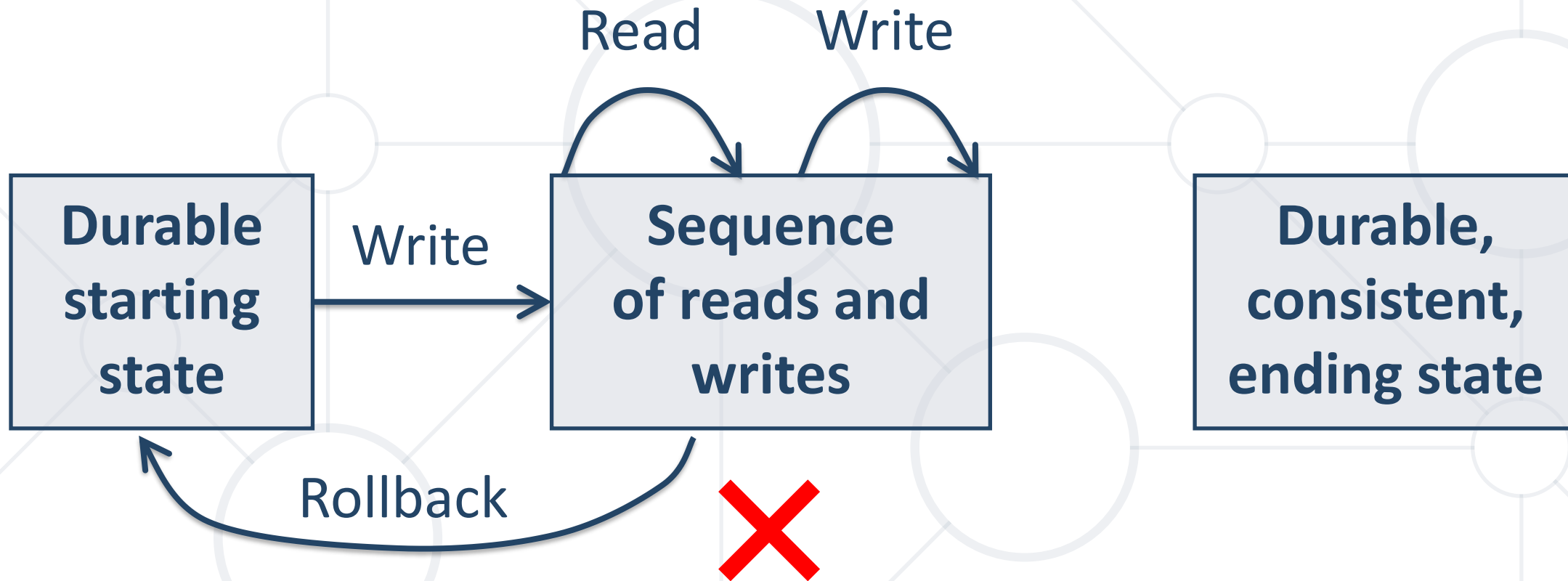


Transactions

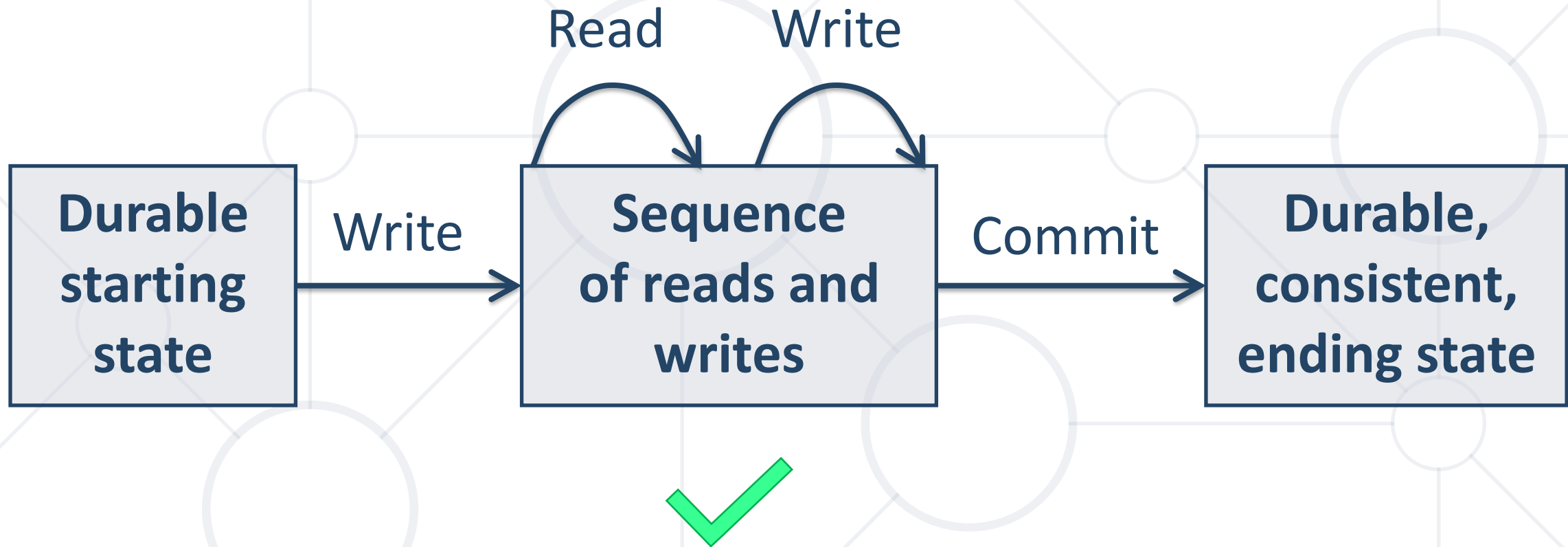
Definition, Usage, ACID Model

- A **Transaction** is a **sequence of actions (database operations) executed as a whole**:
 - Either **all** of them **complete successfully** or **none** of them **do**
- Examples:
 - A bank transfer from one account into another (**withdrawal + deposit**)
 - If either the **withdrawal** or the **deposit fails** the **whole operation is cancelled**

Transactions: Lifecycle (Rollback)



Transactions: Lifecycle (Commit)

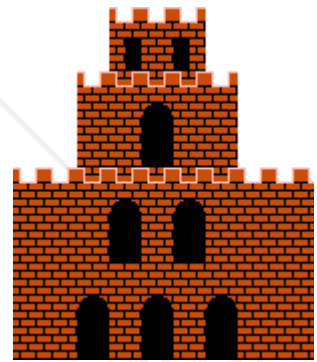


- **Transactions** guarantee the **consistency** and the **integrity** of the **database**
 - All **changes** in a transaction **are temporary**
 - Changes are **persisted** when a **COMMIT** is **executed**
 - At any time, **all changes** can be **canceled** by **ROLLBACK**
- All **changes** are **persisted at once**
 - As long as **COMMIT** is called

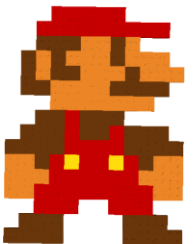
Transactions: What Can Go Wrong?

- Some actions **fail to complete**
 - The application **software** or database **server crashes**
 - The user **cancels the action** while it's **in progress**
- **Interference** from another **transaction**
 - What happens if several transfers run for the same account at the same time?

Checkpoints in Games



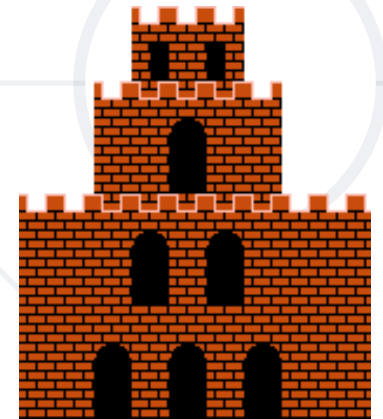
Castle 1-2



Mario

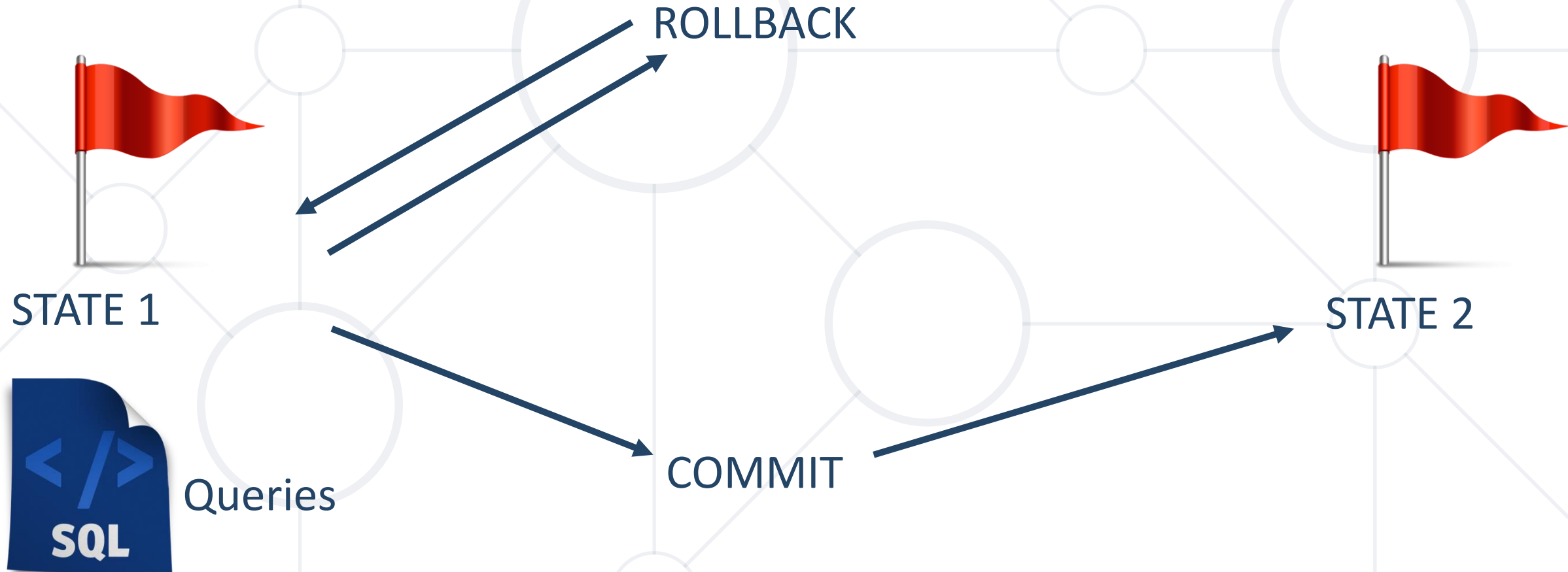
DIE

SURVIVE



Castle 1-3

What Are Transactions?



Transactions Syntax

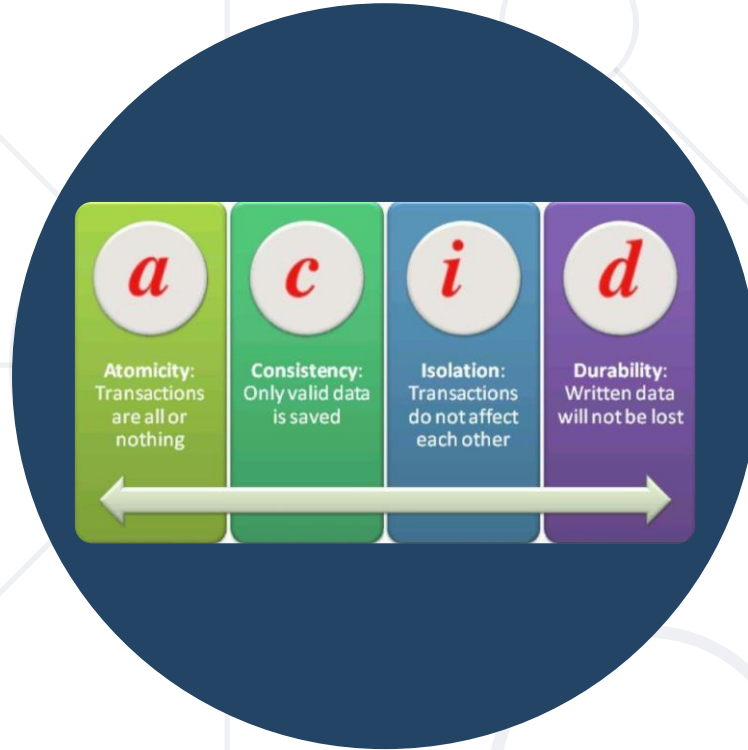
```
CREATE PROC usp_Withdraw (@withdrawAmount DECIMAL(18,2), @accountId
INT)
AS
BEGIN TRANSACTION
UPDATE Accounts SET Balance = Balance - @withdrawAmount
WHERE Id = @accountId
IF @@ROWCOUNT <> 1 -- Didn't affect exactly one row
BEGIN
ROLLBACK
THROW 50001, 'Invalid account!', 1
RETURN
END
COMMIT
```

Start Transaction

Withdraw Money

Undo Changes

Save Changes



ACID Models

Solving Problems Before They Arise

- Modern DBMS servers have **built-in transaction support**
 - Implement "**ACID**" transactions
 - MS SQL Server, Oracle, MySQL, PostgreSQL, etc.
- **ACID** means:
 - **Atomicity**
 - **Consistency**
 - **Isolation**
 - **Durability**



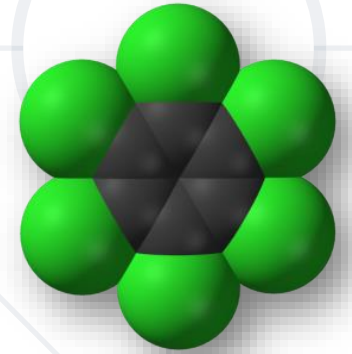
- **Atomicity** means that
 - **Transactions** execute as a **whole**
 - DBMS guarantees that **either all** of the operations are performed **or none** of them
- Example: Transferring funds between bank accounts
 - Either withdraw + deposit both succeed, or none of them do
 - In case of failure, the database stays unchanged



- **Consistency** means that
 - The database has a legal state in both the **transaction's beginning** and **its end**
 - Only **valid data** will be **written** to the DB
 - Transaction **cannot break the rules** of the database
 - Primary keys, foreign keys, check constraints, data types...
- Consistency example:
 - Transaction **cannot end with a duplicate primary key** in a table

- **Isolation** means that
 - **Multiple transactions** running at the same time **do not impact each other's execution**
 - Transactions **don't see** other transactions' **uncommitted changes**
 - Isolation level defines how deep transactions **isolate from one another**
- Isolation example:
 - If two or more people try to buy the last copy of a product, only one of them will succeed

- **Durability** means that:
 - If a transaction is **committed** it becomes **persistent**
 - **Cannot** be **lost** or **undone**
 - Ensured by the use of **database transaction logs**
- Durability example:
 - After funds are transferred and committed, the power supply at the DB server is lost
 - Transaction stays persistent (**no data is lost**)



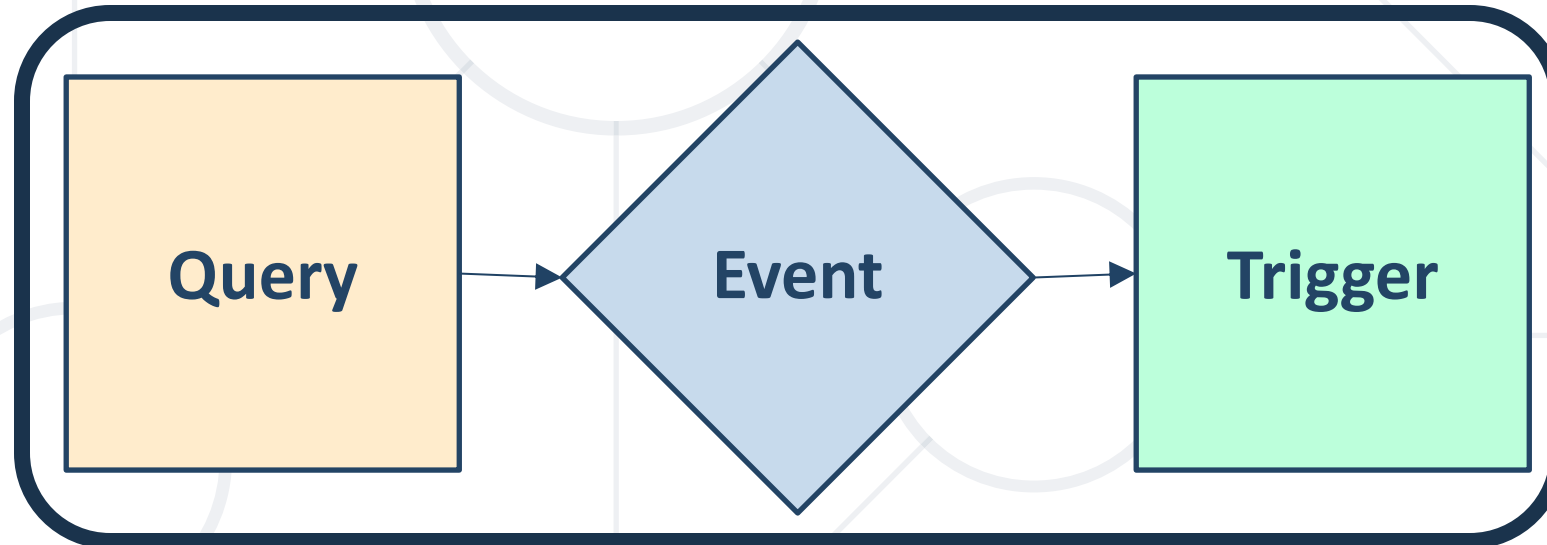


Triggers

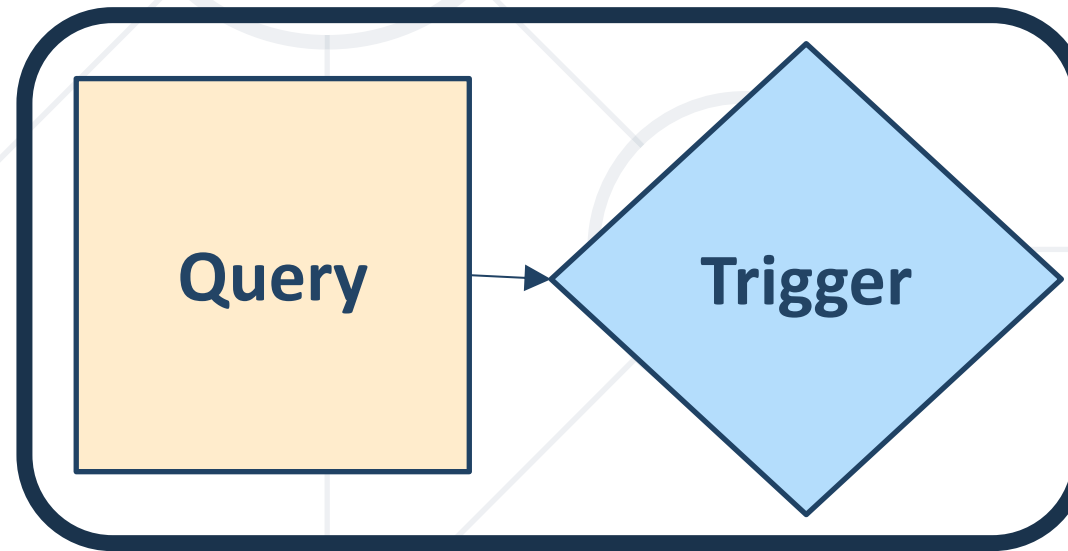
What Are Triggers?

- **Triggers** are very much like **stored procedures**
 - Called **in case of a specific event**
- We do **not call** triggers **explicitly**
 - **Triggers are attached to a table**
 - Triggers are fired **when a certain SQL statement is executed** against the contents of the table
 - Syntax:
 - **AFTER INSERT/UPDATE/DELETE**
 - **INSTEAD OF INSERT/UPDATE/DELETE**

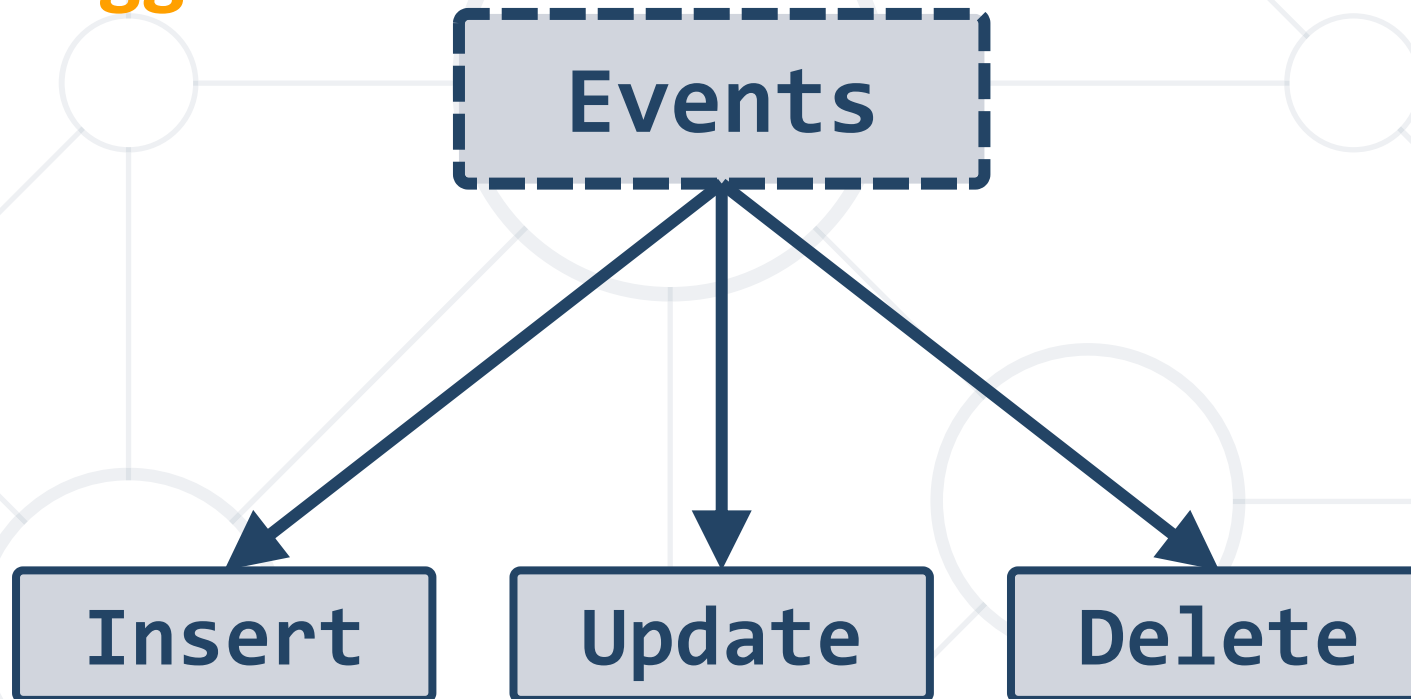
- **AFTER Trigger** is executed **right after an event is fired**



- **INSTEAD OF** Trigger completely replaces an event action from happening
 - You can apply totally different logic



- There are **three different events** that can be applied **within a trigger**

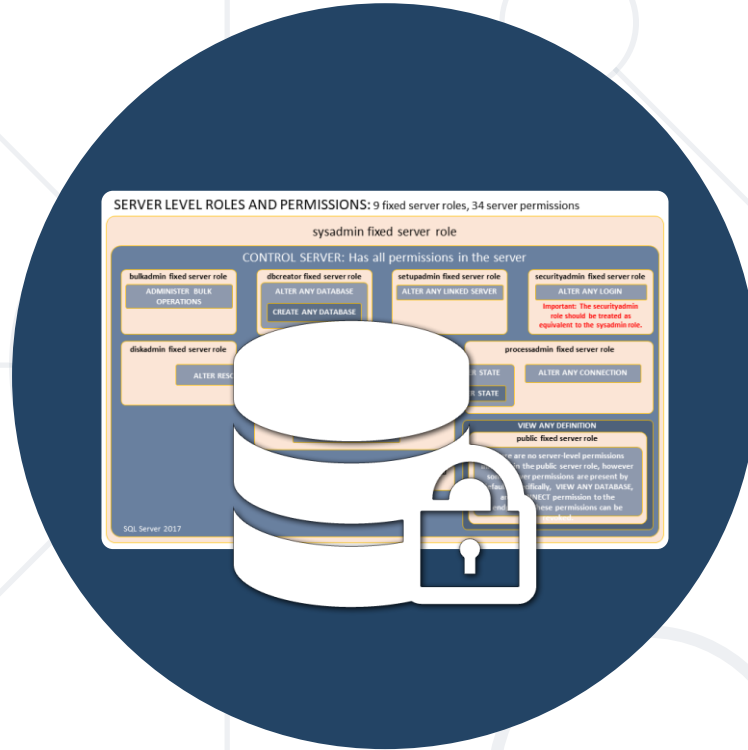


- Defined by the keyword **FOR**

```
CREATE TRIGGER tr_AddToLogsOnAccountUpdate
ON Accounts FOR UPDATE
AS
    INSERT INTO Logs(AccountId, OldAmount, NewAmount, UpdatedOn)
    SELECT i.Id, d.Balance, i.Balance, GETDATE()
    FROM inserted AS i
    JOIN deleted AS d ON i.Id = d.Id
    WHERE i.Balance != d.Balance
GO
```


- Defined by using **INSTEAD OF**

```
CREATE OR ALTER TRIGGER tr_SetIsDeletedOnDelete
ON AccountHolders
INSTEAD OF DELETE
AS
    UPDATE AccountHolders SET IsDeleted = 1
    WHERE Id IN (SELECT Id FROM deleted)
GO
```



Database Security

Fixed Server Roles, Fixed Database Roles

- SQL Server has **two layers of database security**
 - **Fixed Server Roles**
 - sysadmin, bulkadmin, dbcreator, securityadmin
 - **Fixed Database Roles**
 - db_owner, db_securityadmin, db_accessadmin
 - db_backupoperator, db_ddladmin
 - db_datareader/db_datawriter

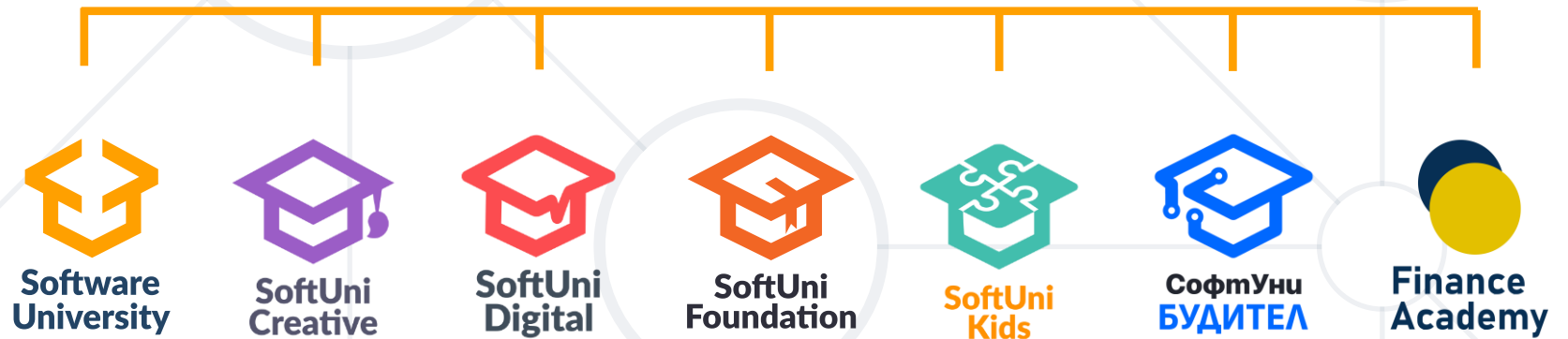
- SQL Server lets us create **custom roles**
 - **Collection of privileges** (permissions)
- Fine **control over permissions**
 - Can use **one role for multiple users** (groups)
- Makes **auditing operations easier**



- **Transactions** give our operations **stability**
 - Operation Integrity
 - Solving the concurrent operation problem
 - The ACID model is implemented in most RDBMS
- **Triggers** apply a given behavior when a condition is hit
 - Gives us temporary **INSERTED** and **DELETED** tables
- **Security** in SQL Server can be finely controlled
 - Using fixed **server roles** and fixed **database roles**
- **Custom roles** control permissions even more finely



Questions?



SoftUni Diamond Partners



THE CROWN IS YOURS



- Software University – High-Quality Education, Profession and Job for Software Developers

- softuni.bg, about.softuni.bg

- Software University Foundation

- softuni.foundation

- Software University @ Facebook

- facebook.com/SoftwareUniversity



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg>
- © Software University – <https://softuni.bg>

