COMMON DEVELOPER MISTAKES

MILOŠ RADIVOJEVIĆ, MICROSOFT DATA PLATFORM MVP

Common Developer Mistakes

- Order of outer JOINS
- Placement of predicates and evaluation order
- NULL and NOT IN
- BETWEEN and date predicates
- DISTINCT or TOP(1) as ad-hoc solutions for logical problems
- MERGE statement – unique key violation
- Omitting schema names and aliases
- Delete/Update in Large Tables (LOCK Escalation)
- Not archiving/deleting (working with all data instead of actual data sets)
- Creating reports from production server

Error Handling in SQL Server

```
INSERT INTO T1(id) VALUES(1);
INSERT INTO T1(id) VALUES(1);
```

```
Msg 2627, Level 14, State 1, Line 17
Violation of PRIMARY KEY constraint 'PK__T1__3213E83FE00A3B3D'. Cannot insert duplicate key in
object 'dbo.T1'. The duplicate key value is (1).
```

```
SELECT 1/0
```

```
Msg 8134, Level 16, State 1, Line 19
Divide by zero error encountered.
```

```
SELECT CAST('abc' AS INT)
```

```
Msg 245, Level 16, State 1, Line 21
Conversion failed when converting the varchar value 'abc' to data type int.
```

Error Handling in SQL Server

- Types of error and where and how they occur?
  - Syntax errors
  - Resolving errors
  - Runtime errors
    - Statement level
    - Batch level
    - Connection level
- Where to handle errors?
  - In the SQL Server database engine
  - In the client application

Error Severity

- 0-10 Info messages
- 11-16 Errors that can be corrected by user
- 17-19 Error that cannot be corrected by user
- 20+ Serious, internal errors

Transaction

- Unit of activity that should be considered atomic
- ACID
  - Atomic (all or nothing)
  - Consistent (data are consistent at the beginning and at the end of transaction)
  - Isolated (process is isolated from other processes)
  - Durability (it is guaranteed that changes will be permanently saved in case of a failure)
- Every SQL statement is considered as an implicit transaction

Transaction Myths

- Myths about transactions
  - A transaction will be rolled back if an error occurred within the transaction - FALSE
  - Every stored procedure is considered as an implicit transaction – FALSE

- Transaction doesn't guarantee that business logic within it is consistent!

Error Handling in SPs - Recommendation

```
CREATE OR ALTER PROCEDURE dbo.YourSP
AS
SET XACT_ABORT ON;
SET NOCOUNT ON;
BEGIN TRY
      --here your code
END TRY
BEGIN CATCH
      IF @@TRANCOUNT > 0 ROLLBACK TRAN;
      ;THROW
END CATCH
```

Example: MERGE Statement Exception

```
BEGIN TRY
  WHILE 1 = 1
  BEGIN
    MERGE INTO dbo.tabMerge AS t
    USING (SELECT CHECKSUM(SYSDATETIME()), N'abc', N'test', 'blah',1)
          AS s(id,c1,c2,c3,c4)
        ON s.id = t.id
    WHEN MATCHED THEN UPDATE
        SET t.c1 = s.c1,
            t.c2 = s.c2,
            t.c3 = s.c3,
            t.c4 = s.c4
    WHEN NOT MATCHED THEN INSERT
      VALUES(s.id, s.c1, s.c2, s.c3, s.c4);
  END;
END TRY
BEGIN CATCH
  ;THROW;
END CATCH;
```

```
Msg 2627, Level 14, State 1, Line 9
Violation of PRIMARY KEY constraint 'PK_tabMerge'. Cannot insert duplicate key in object 'dbo.tabMerge'. The duplicate key value is (1639635611).
```

Solution

- Understand what does the exception mean!

- It's a primary key violation!?
- OK, what's problem with it? Someone tried to insert a row which already exists…
- Yes, but…
- You wanted a row with values 7,6 an 5 there?
- Yes.
- Is this row there?
- Yes, but, my session did not insert it
- Is it important that particularly your session has to insert it?

Solution 1 – Ignore PK errors

```
BEGIN CATCH
    IF ERROR_NUMBER() <> 2627
    THROW;
END CATCH;
```

Solution 2 – Retry

```
DECLARE @retries TINYINT = 2;
WHILE (@retries > 0)
BEGIN
  BEGIN TRY
                  WHEN MATCHED AND t.SequenceNumber < s.SequenceNumber THEN UPDATE
---
    SET @retries = 0;
  END TRY

  BEGIN CATCH
IF ERROR_NUMBER() <> 2627
SET @retries -= 1;
    ELSE
      THROW;
END CATCH;
END
```

Solution 3 - Serialization

```
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
```

- In a few corner cases is required
- That kills parallelism and significantly reduces throughput
- Use it only if you are sure you know what are you doing and there is no other way to solve an issue
  - Usually, there is another way