**Определения (Лекция 18)**

**Структурированная обработка ошибок:**

BEGIN TRY

PRINT 16/2;

PRINT 'No error';

END TRY

BEGIN CATCH

PRINT 'Error';

END CATCH;

**Транзакции:**

Пример 0: Что будет, если написать так?

CREATE TABLE #T (keycol int primary key, col1 int)

INSERT INTO #T(keycol, col1) VALUES(4, 101);

INSERT INTO #T(keycol, col1) VALUES(5, 201);

INSERT INTO #T(keycol, col1) VALUES(6, 101);

INSERT INTO #T(keycol, col1) VALUES(5, 201);

SELECT \* FROM #T

DROP TABLE #T

Пример 0.1: А так?

CREATE TABLE #T (keycol int primary key, col1 int)

INSERT INTO #T(keycol, col1) VALUES(4, 101), (5, 201), (6, 101), (5, 201);

SELECT \* FROM #T

DROP TABLE #T

Пример 1: Пример явной отмены транзакции

CREATE TABLE #T (keycol int primary key, col1 int)

-- вставим значения и откатим транзакцию, хотя никакой ошибки не было. Просто мы не захотели вносить изменения

BEGIN TRAN;

INSERT INTO #T(keycol, col1) VALUES(4, 101);

INSERT INTO #T(keycol, col1) VALUES(5, 201);

SELECT \*

FROM #T

ROLLBACK TRAN;

SELECT \* FROM #T

DROP TABLE #T

Пример 2: Подтверждение и отмена транзакции

|  |  |
| --- | --- |
| CREATE TABLE #T (keycol int primary key, col1 int)  BEGIN TRAN;  INSERT INTO #T(keycol, col1) VALUES(4, 101);  INSERT INTO #T(keycol, col1) VALUES(5, 201);  COMMIT TRAN;  BEGIN TRAN;  INSERT INTO #T(keycol, col1) VALUES(6, 101);  INSERT INTO #T(keycol, col1) VALUES(5, 201);  ROLLBACK TRAN;  SELECT \* FROM #T  DROP TABLE #T | keycol col1  4 101  5 201  И ошибка про невозможность добавления 5. Но 6 не добавится.  Что изменится, если вместо ROLLBACK вставить COMMIT? |

**Пример транзакции с обработкой ошибок:**

CREATE TABLE #T (keycol int primary key, col1 int)

-- вставим значения

BEGIN TRAN;

INSERT INTO #T(keycol, col1) VALUES(4, 101);

INSERT INTO #T(keycol, col1) VALUES(5, 201);

COMMIT TRAN;

SELECT \* FROM #T

-- Здесь изменять значения id и val для тестирования вывода сообщений об ошибках.

DECLARE @id int, @val float

**SET @id = 7** -- неповторный ключ для тестирования транзакции

**--SET @id = 4** -- повторный ключ для того, чтобы вставка в транзакции сломалась

**--SET @val = 2** -- для тестирования ошибки транзакции

**SET @val = 0** -- чтобы в catch попадали по делению на ноль, если вставка данных прошла.

-- знакомый Вам из программирования блок TRY-CATCH.

BEGIN TRY

BEGIN TRAN;

SELECT 1 Где, **@@TRANCOUNT** Сколько -- проверка значения переменной

INSERT INTO #T(keycol, col1) VALUES(6, 201); -- новое значение ключа

INSERT INTO #T(keycol, col1) VALUES(@id, 201); -- тестовая вставка, в зависимости от @id транзакция либо откатится, либо нет.

SELECT 2 Где, @@TRANCOUNT Сколько

COMMIT TRAN; -- если всё ок, то транзакция пройдет, если ошибка при вставке, то попадаем в catch сразу из ошибки.

SELECT 3 Где, @@TRANCOUNT Сколько

PRINT 10/@val -- если вставка прошла успешно, то попадание в catch происходит, если val = 0

END TRY

BEGIN CATCH

SELECT 4 Где, @@TRANCOUNT Сколько

-- Подумайте, зачем нужно это условие на служебную переменную @@Trancount?

**IF @@TRANCOUNT > 0**

ROLLBACK TRAN;

DECLARE @ErrorMessage NVARCHAR(4000);

DECLARE @ErrorSeverity INT;

DECLARE @ErrorState INT;

-- с 2012 версии для проброски текста ошибки существует THROW.

SELECT

@ErrorMessage = ERROR\_MESSAGE(),

@ErrorSeverity = ERROR\_SEVERITY(),

@ErrorState = ERROR\_STATE();

-- посмотрите, что будет, если **закомментировать RAISEERROR**: транзакция откатится, но сообщения об ошибке не будет.

RAISERROR (@ErrorMessage, -- Message text.

@ErrorSeverity, -- Severity.

@ErrorState -- State.

);

END CATCH

SELECT 5 Где, @@TRANCOUNT Сколько

SELECT \* FROM #T

DROP TABLE #T