Базы данных

11 | Обработка ошибок и транзакции

План

- Ошибки и сообщения об ошибках
- Выбрасывание ошибок
- Обработка ошибок
- Введение в транзакции
- Реализация явных транзакций

Ошибки и сообщения об ошибках

Elements of Database Engine Errors	
Error number	Уникальный номер, определяющий конкретную ошибку
Error message	Текст, описывающий ошибку
Severity	Числовая индикация серьезности от 1-25
State	Код внутреннего состояния ядра базы данных
Procedure	Имя хранимой процедуры или триггера, в котором произошла ошибка
Line number	Какой оператор в пакете или процедуре генерировал ошибку

В SQL Server (не Azure SQL Database):

- Сообщения об ошибках в **sys.messages;**
- Можно добавлять пользовательские сообщения через **sp_addmessage.**

Выбрасывание ошибок. RAISERROR

- Команда RAISERROR
  - Выбрасывание ошибок, определенных в **sys.messages** (только SQL Server)
  - Выбрасывание явно указанного сообщения, кода серьезности и внутреннего состояния (SQL Server и Azure SQL Database)

```
RAISERROR (‘Произошла ошибка’, 16, 0);
```

```
RAISERROR (N’Это сообщение %s %d.', -- Текст сообщения.
           10, -- Код серьезности (Severity),
           1, -- Код состояния (State),
           N’номер', -- Первый аргумент.
           5); -- Второй аргумент.
-- Текст сообщения: Это сообщение номер 5.
```

Выбрасывание ошибок. THROW

- Команда THROW:
  - Замена для RAISERROR;
  - Выбрасывание явного номера ошибки, сообщения и состояния (серьезность = 16);
  - Повторное выбрасывание (пробрасывание) существующей ошибки.

THROW 50001, ‘Произошла ошибка‘, 0;

```
EXEC sys.sp_addmessage @msgnum = 60000, @severity = 16,
@msgtext = N'Это тестовое сообщение с числовым параметром (%d), строковым
параметром (%s) и другим строковым параметром (%s).',
@lang = 'us_english'        --, @replace = 'replace';
GO

DECLARE @msg NVARCHAR(2048) = FORMATMESSAGE(60000, 500, N'Первая строка',
N'вторая строка');
THROW 60000, @msg, 1;
```

DEMO

Выбрасывание ошибок

Обработка ошибок

- Используйте оператор TRY…CATCH;
- Обработка ошибок в CATCH:
  - Получить информацию об ошибке:
    - @@ERROR
    - ERROR_NUMBER()
    - ERROR_MESSAGE()
    - ERROR_SEVERITY()
    - ERROR_STATE()
    - ERROR_PROCEDURE()
    - ERROR_LINE()
  - Выполнить коррекцию или журналирование;
  - Пробросить исходную ошибку или выбросить новую пользовательскую ошибку.

```
DECLARE @Discount INT = 0;
BEGIN TRY
        UPDATE SalesLT.Product
        SET Price = Price / @Discount
END TRY
BEGIN CATCH
        PRINT ERROR_MESSAGE();
        THROW 50001, 'Произошла ошибка', 0;
END CATCH;
```

DEMO

Обработка ошибок

Введение в транзакции

- Транзакция – группа задач, представляющих единое целое.
- Все задачи транзакции должны вместе успешно выполниться или не выполниться; частичное выполнение не разрешено.

```
-- Две задачи, составляющие неделимую единицу:
INSERT INTO SalesLT.SalesOrderHeader ...
INSERT INTO SalesLT.SalesOrderDetail ...
```

- Отдельные операторы изменения данных автоматически трактуются как автономные транзакции.
- SQL Server использует механизм блокировок и журналирование изменений для поддержки транзакций.

Свойства транзакций (ACID)

- Atomicity – атомарность («все или ничего»). Если часть транзакции терпит неудачу, то и вся транзакция терпит неудачу и БД остается в исходном состоянии (как до начала транзакции).
- Consistency – связность, целостность. Транзакция переводит БД из одного согласованного (правильного) состояния в другое (успешная проверка всех ограничений).
- Isolation – изоляция. Транзакции выполняются независимо друг от друга (как если бы они запускались строго последовательно).
- Durability – долговечность. Как только транзакция подтверждена, то изменения не могут быть потеряны даже в случае отключения питания компьютера, сбоя или других ошибок.

Реализация явных транзакций

- Используйте **BEGIN TRANSACTION** для начала транзакции.
- Используйте **COMMIT TRANSACTION** для завершения транзакции.
- Используйте **ROLLBACK TRANSACTION** для отмены транзакции.
  - Или включите опцию XACT_ABORT для автоматической отмены транзакции при ошибках
- Используйте @@TRANCOUNT и XACT_STATE() для проверки состояния транзакции.

```
BEGIN TRY
  BEGIN TRANSACTION
    INSERT INTO SalesLT.SalesOrderHeader…
    INSERT INTO SalesLT.SalesOrderDetail…
  COMMIT TRANSACTION
END TRY
BEGIN CATCH
  IF @@TRANCOUNT > 0
  BEGIN
    ROLLBACK TRANSACTION
  END
  PRINT ERROR_MESSAGE();
  THROW 50001, 'Произошла ошибка', 0;
END CATCH;
```

Реализация транзакций. @@TRANCOUNT

Возвращает количество операторов BEGIN TRANSACTION выполненных в текущей сессии (без парных подтверждений или отмен транзакций).

```
PRINT @@TRANCOUNT
--  Оператор BEGIN TRAN увеличит счетчик транзакций на 1.
BEGIN TRAN
   PRINT @@TRANCOUNT
   BEGIN TRAN
      PRINT @@TRANCOUNT
-- Оператор COMMIT уменьшит счетчик транзакций на 1.
   COMMIT
   PRINT @@TRANCOUNT
COMMIT
PRINT @@TRANCOUNT
```

```
-- Результаты:
--0
--1
--2
--1
--0
```

DEMO

Реализация транзакций

Изучено

- Ошибки и сообщения об ошибках
- Выбрасывание ошибок
- Обработка ошибок
- Введение в транзакции
- Реализация явных транзакций