Тема лекції 13:

Програмування в SQL Server

- Збережені процедури
- Інтеграція з платформою .NET Framework
- Користувацькі функції
- Тригери

Основи збережених процедур

- **Збережена процедура** – попередньо скомпільований набір операторів мови SQL, який зберігається на сервері.
- Збережені процедури є основним засобом оформлення часто розв’язуваних задач, який забезпечує їх ефективне виконання, оскільки інструкції не потрібно повторно компілювати.
- Збережені процедури в SQL Server є аналогічні процедурам в інших мовах програмування

Переваги збережених процедур у порівнянні з кодом, який зберігається локально на клієнтських комп’ютерах

- Збережені процедури ***реєструються на сервері***

- Збережені процедури можуть мати ***атрибути безпеки*** і до них можна прикріпляти сертифікати. Користувачі можуть мати права на виконання збережених процедур замість прямих прав для роботи з об’єктами, на які посилаються ці процедури.

Переваги збережених процедур

- Збережені процедури підтримують ***модульне програмування***. Процедуру можна створити один раз і за необхідністю викликати її будь-яку кількість раз. Це робить зручнішим обслуговування програми і дозволяє уніфікувати доступ програм до бази даних.
- Збережені процедури дозволяють ***зменшити мережевий трафік***. Операцію, яка займає сотні рядків програмного коду Transact-SQL, можна виконати в одній інструкції, яка обробляє процедуру, а не передає цей код по мережі.

Типи збережених процедур

- користувацькі збережені процедури
  - збережені процедури Transact-SQL
  - збережені процедури CLR
- розширені збережені процедури
- системні збережені процедури

Користувацькі збережені процедури

- Це модулі або підпрограми, в яких міститься повторно використовуваний код.
- Збережена процедура:
  - приймає вхідні параметри,
  - виконує інструкції мови визначення даних (DDL) і мови обробки даних (DML),
  - повертає клієнту табличні або скалярні результати, а також вихідні параметри.
- В SQL Server збережена процедура може бути
  - T-SQL-процедурою
  - CLR-процедурою.

Користувацькі збережені процедури Transact-SQL

- ЦЕ збережена колекція інструкцій мови Transact-SQL, яка може приймати і повертати параметри. Наприклад:
  - збережена процедура може містити інструкції, які виконують вставку нового рядка в одну або декілька таблиць на основі значень, отриманих від клієнтської програми
  - повертати програмі дані, які отримані з БД.
- Приклад: у веб-додатку електронної комерції збережена процедура може повертати відомості про конкретні продукти, в залежності від критеріїв пошуку, заданих користувачем в інтерактивному режимі.

Користувацькі збережені CLR процедури

- являють собою посилання на метод середовища CLR (Common Language Runtime – загальномовного середовища виконання) платформи .NET Framework, який може приймати і повертати користувачу параметри.
- реалізовані у вигляді загальних статичних методів класу у збірці платформи .NET

Розширені збережені процедури

- Дозволяють створювати зовнішні підпрограми на мовах програмування
- Являють собою бібліотеки DLL, які можуть динамічно завантажуватись і виконуватись екземпляром SQL Server
- Виконання таких процедур відбувається безпосередньо в адресному просторі екземпляра SQL Server і програмується при використанні інтерфейсів API розширених збережених процедур SQL Server

Користувацькі збережені CLR процедури і розширені збережені процедури

- Інтеграція з середовищем CLR є **надійнішим способом**, ніж використання розширених збережених процедур.
- ***Зауваження:*** використання розширених збережених процедур необхідно уникати в нових розробках і необхідно змінювати існуючі програми, в яких вони застосовуються. Замість цього необхідно використовувати інтеграцію з середовищем CLR.

Системні збережені процедури

- За допомогою цих процедур відбувається багато адміністративних дій в SQL Server.
- Фізично такі процедури зберігаються в базі даних ресурсів і мають префікс sp_.
- Логічно такі процедури відображаються в будь-якій системній або користувацькій БД у схемі sys.
- В SQL Server до системних збережених процедур можуть застосовуватись інструкції GRANT, DENY і REVOKE.
- SQL Server підтримує системні збережені процедури, які забезпечують інтерфейс між SQL Server і зовнішніми програмами для виконання різних операцій з обслуговування системи. Такі розширені системні збережені процедури мають префікс xp_.

Створення збережених процедур

- Служить інструкція Transact-SQL CREATE PROCEDURE
- Для шифрування тексту збереженої процедури використовується параметр WITH ENCRYPTION. У цьому випадку:
  - результат шифрування не видно ні в одній системній таблиці чи системній віртуальній таблиці SQL Server.
  - користувачі, які не мають доступу до системних таблиць чи файлів БД, не зможуть отримати текст процедури.
  - привілегійовані користувачі, які мають прямий доступ до файлів БД, використовуючи операцію дешифрування, можуть отримати вихідний текст визначення збереженої процедури.

Правила проектування збережених процедур

- Інструкція CREATE PROCEDURE може включати будь-яку кількість інструкцій SQL будь-якого типу, ***окрім наступних***:
  - CREATE або ALTER FUNCTION
  - CREATE або ALTER TRIGGER
  - CREATE або ALTER PROCEDURE
  - CREATE або ALTER VIEW
  - CREATE AGGREGATE;
  - CREATE DEFAULT;
  - SET PARSEONLY;
  - SET SHOWPLAN_TEXT;
  - USE <database_name>
  - CREATE RULE;
  - CREATE SCHEMA;
  - SET SHOWPLAN_ALL;
  - SET SHOWPLAN_XML;
- Усі решта об’єктів можна створювати всередині збереженої процедури.
- До створеного в процедурі об’єкту можна звертатись, як і до будь-якого іншого об’єкту БД.

Правила проектування збережених процедур

- В збереженій процедурі можна ***звертатись до тимчасових таблиць***. Якщо в збереженій процедурі створюється локальна тимчасова таблиця, то вона може використовуватись лише в ній і при виході з процедури видалиться.
- При виконанні збереженої процедури, яка ***викликає іншу збережену процедуру***, остання може звертатись до всіх об’єктів, які створені першою, включаючи тимчасові таблиці.
- При виконанні ***віддаленої збереженої процедури***, яка виконує операції на віддаленому екземплярі SQL Server, не можна зробити відкат цих операцій. Віддалені збережені процедури не приймають участь в транзакціях.

Правила створення збережених процедур

- Інструкцію CREATE PROCEDURE не можна використовувати з іншими інструкціями SQL в одному пакеті (до команди GO).
- Збережену процедуру можна створити лише в поточній БД.
- Для створення збережених процедур необхідно мати права для CREATE PROCEDURE в базі даних і права ALTER у відповідній схемі. Для створення збережених процедур CLR необхідно або володіти збіркою, на яку посилається <method_specifier>, або мати в цій збірці права REFERENCES.

Правила створення збережених процедур

- Для підвищення швидкодії ***всі об'єкти***, які використовуються у межах однієї збереженої процедури, повинні ***належати її власнику***, яким є власник БД dbo. В іншому випадку витрачається багато часу на перевірку прав доступу. Найефективніше створити збережену процедуру від імені власника БД і призначати іншим користувачам права на її виконання всюди, де це можливо.
- Збережені процедури є об'єктами схеми, і їх імена повинні відповідати вимогам до імен ідентифікаторів. При повному форматі імені процедури (вказання її схеми) ядру СУБД не потрібно шукати процедуру в декількох схемах.

Правила створення збережених процедур

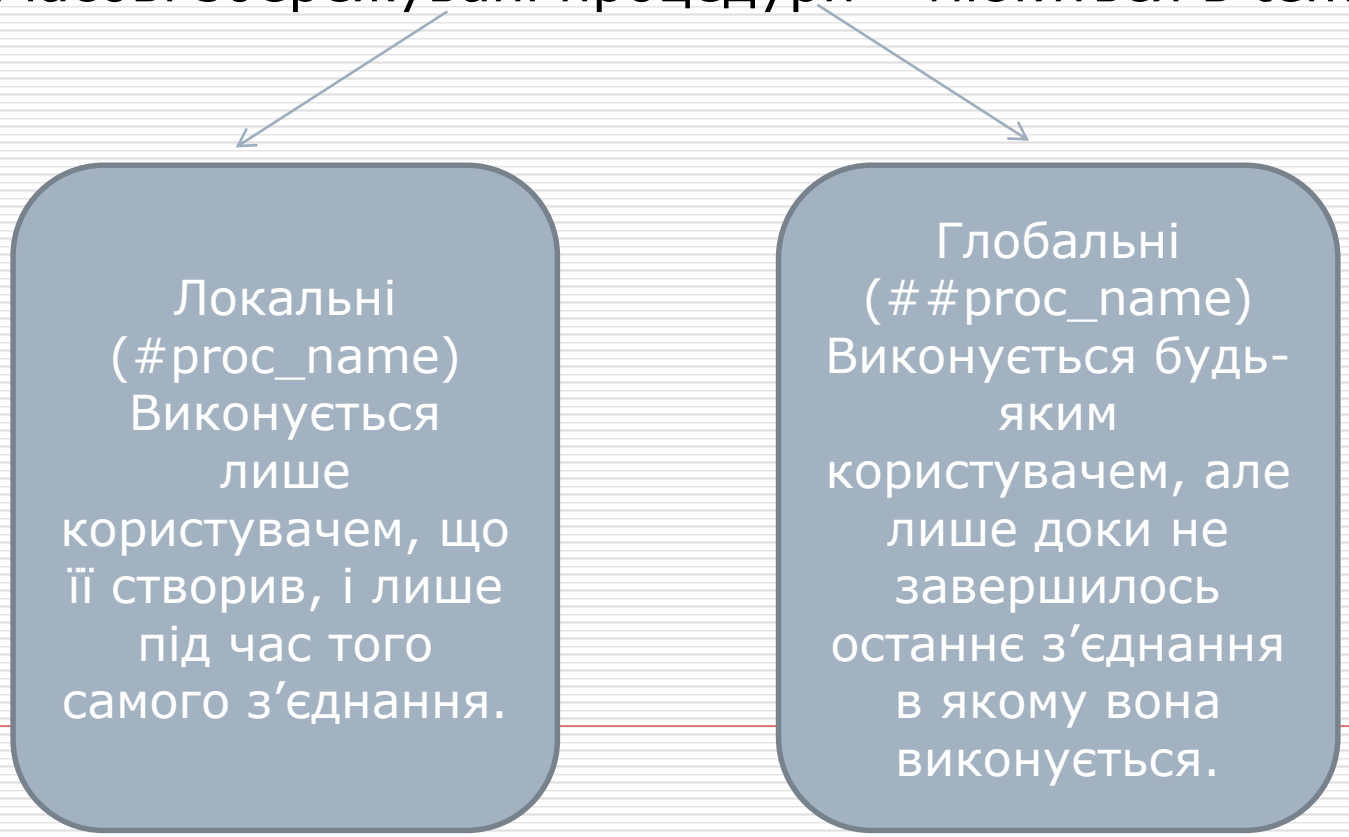
- Не рекомендується присвоювати збереженим процедурам імена з префіксом sp_ (SQL Server використовує префікс sp_ для позначення системних збережених процедур).
- Імена об’єктів при написанні збереженої процедури найкраще задавати у повному форматі, а саме

<obj_owner>.<obj_name>.

Тимчасові збережені процедури

«Базові» збережувані процедури – містяться в поточній БД.

Тимчасові збережувані процедури – містяться в tempdb.



Вибір між процедурами Transact-SQL і CLR процедурами

- якщо потрібен доступ до даних сервера, то необхідно користуватись мовою Transact-SQL;
- якщо потрібно максимально використовувати ресурси сервера і виконувати складні логічні задачі, то необхідно використовувати мови програмування, які підтримують платформу .NET.

Випадки використання CLR процедур

- розробка складної програмної логіки з обробкою виняткових ситуацій;
- використання чужих бібліотек класів для реалізації власних компонент;
- написання програм баз даних, які вимагають багато обчислень і виконання операцій розрахунково-аналітичного характеру;
- необхідність зберігання та обробки об’єктів всередині сервера баз даних

Можливість виконання коду CLR в SQL Server

- Можливість SQL Server виконувати код CLR за замовчуванням відключена.
- Можна створювати, змінювати і видаляти об’єкти бази даних, які посилаються на модулі керуючого коду, але ці посилання не будуть виконані в SQL Server, поки не включиться параметр clr enabled за допомогою процедури sp_configure.

```
USE sample;
EXEC sp_configure 'clr_enabled', 1
RECONFIGURE
```

Збережені процедури CLR

Для реалізації, компіляції та збереження процедур з використанням CLR необхідно виконати 4 кроки у вказаному порядку:

1. Напишіть збережену процедуру мовою C# та відкомпілюйте програму, використовуючи відповідний компілятор.
2. Використовуйте оператор CREATE ASSEMBLY для створення відповідного виконуваного файлу.
3. Збережіть процедуру як серверний об’єкт через оператор CREATE PROCEDURE.
4. Виконайте процедуру, використовуючи оператор EXECUTE.

Виконання збережених процедур в SQL Server

- Використовується команда Transact-SQL EXECUTE або EXEC.
- Якщо збережена процедура є першою інструкцією в пакеті (до команди GO), то команду EXEC можна не записувати.

Модифікація збережених процедур

- Якщо необхідно змінити інструкції або параметри збереженої процедури, можна видалити її і створити заново. При цьому всі права доступу будуть втрачені.
- При безпосередньому редагуванні можна змінити інструкції і параметри, а права доступу залишаться, а також залишаться залежні від неї процедури або тригери. Для цього використовується команда

  ALTER PROCEDURE.
- При модифікації збереженої процедури можна задати параметр шифрації або параметр перекомпіляції.
- Зміна імені або визначення збереженої процедури може призвести до того, що усі залежні від неї об’єкти при виконанні будуть повертати помилку, якщо вони не були оновлені у відповідності зі змінами, внесеними в процедуру.

Перекомпіляція збережених процедур

- Після зміни бази даних через додавання індексів чи зміни даних в індексованих стовпцях необхідно заново виконати оптимізацію вихідних планів запитів через перекомпіляцію.
- Така оптимізація проводиться ***автоматично*** під час першого виконання збереженої процедури після перезапуску SQL Server, а також при зміні базової таблиці, яка використовується збереженою процедурою.
- Якщо додається індекс, який надає процедурі переваги, то оптимізація не проводиться до наступного виконання збереженої процедури. У такій ситуації доцільно провести ***примусову*** перекомпіляцію при наступному виконанні збереженої процедури (*див. дод. ресурси*)

Видалення збережених процедур

- Коли збережена процедура не потрібна, її можна видалити командою

  DROP PROCEDURE

- Якщо на видалену процедуру посилається інша збережена процедура, то при її виклику Microsoft SQL Server відобразить повідомлення про помилку. Однак, якщо замість видаленої визначити іншу збережену процедуру з таким же ім’ям і параметрами, то процедури, які на неї посилаються будуть виконуватись успішно.

Відмінності збережених процедур і користувацьких функцій

- Збережені процедури відрізняються від функцій тим, що вони не повертають значення на місце своїх імен, і їх не можна безпосередньо використовувати у виразах.
- Функція являє собою підпрограму Transact-SQL або середовища CLR, яка повертає значення.
- Користувацька функція не може виконувати дії, які змінюють стан бази даних.
- Користувацькі і системні функції можуть викликатись із запиту.

Використання користувацьких функцій

- в інструкціях Transact-SQL, наприклад SELECT
- у програмах, які викликають функцію
- у визначенні іншої користувацької функції
- для параметризації віртуальної таблиці (view)
- для визначення стовпця таблиці
- для визначення обмеження CHECK на стовпець
- для заміни збереженої процедури

Створення користувацьких функцій

- Створюється користувацька функція командою

  CREATE FUNCTION

- Користувацькі функції (написані на Transact-SQL або на платформі .NET Framework) можуть бути:
  - скалярними
  - які повертають табличне значення

Користувацькі скалярні функції

- Повертають одне значення типу даних, який задається у фразі RETURN.
- Текст простої скалярної функції складається з єдиної інструкції (часто це інструкція SELECT).
- Якщо текст функції складається з декількох інструкції, то він поміщається у блок BEGIN...END, який повертає одне значення
- Такі функції можуть повертати будь-які типи даних, крім text, ntext, image, cursor, spatial, hierarchyID і timestamp.
- Скалярні функції, як і збережені процедури, виконуються інструкцією EXECUTE.

Користувацькі табличні функції

- Повертають тип даних table. Результатом простої такої функції є таблиця, як результатний набір інструкції SELECT.
- За допомогою табличних функцій можна створювати та вставляти рядки у табличні результати. У таких випадках використовується блок BEGIN...END, який і визначає текст функції, що складається з послідовності інструкцій Transact-SQL.

Особливості користувацьких табличних функцій

- фраза RETURN визначає ім’я локальної змінної для результатної таблиці, а також формат результатної таблиці;
- інструкції Transact-SQL в тілі функції створюють і вставляють рядки в результатну змінну, яка визначена фразою RETURN;
- при виконанні фрази RETURN рядки, які вставляються у змінну, повертаються у якості результатних табличних даних функції;
- фраза RETURN не може мати аргумента;

Особливості користувацьких табличних функцій

- ні одна з інструкцій Transact-SQL з тіла функції не може повертати результатний набір безпосередньо користувачу: єдині дані, які можуть повернутись користувачу – це таблиця table, яка повертається цією функцією
- функція викликається там, де є права на табличні вирази, наприклад, у фразі FROM.
- в інструкції SELECT користувацьку функцію, яка повертає значення типу table, можна викликати лише один раз.

Користувацькі табличні функції і віртуальні таблиці

- Користувацькі функції, які повертають табличне значення, можуть бути повноцінною альтернативою представленням (view – віртуальним таблицям).
- Якщо представлення (view) обмежені однією інструкцією SELECT, то користувацькі функції можуть містити декілька інструкцій, які забезпечують ефективнішу логіку, ніж та, яка можлива у представленнях.
- Табличні функції також можуть заміняти і збережені процедури, які повертають один результатний набір.

Альтернативи табличним функціям

```
USE sample;
GO
CREATE TYPE departmentType AS TABLE
(dept_no CHAR(4), dept_name CHAR(25),location CHAR(30));
GO
CREATE TABLE #dallasTable
(dept_no CHAR(4), dept_name CHAR(25),location CHAR(30));
GO
CREATE PROCEDURE insertProc
@Dallas departmentType READONLY
AS SET NOCOUNT ON
INSERT INTO #dallasTable (dept_no, dept_name, location)
SELECT * FROM @Dallas
GO
DECLARE @Dallas AS departmentType;
INSERT INTO @Dallas( dept_no, dept_name, location)
SELECT * FROM department
WHERE location = 'Dallas'
EXEC insertProc @Dallas;
```

Тригер як вид збереженої процедури

- Тригер - це відкомпільована SQL-процедура, виконання якої обумовлено певними подіями всередині реляційної бази даних
- Тригер – це спеціальний тип збереженої процедури, яка автоматично запускається сервером при спробі зміни даних у таблицях, з якими пов’язаний цей тригер

Тригери як інструмент для підтримки цілісності даних

- За допомогою обмежень цілісності, правил і значень за замовчуванням не завжди можна домогтися потрібного рівня функціональності.
- Часто потрібно реалізувати складні алгоритми перевірки даних, що гарантують їх достовірність і реальність.
- Іноді необхідно відстежувати зміни значень таблиці, щоб потрібним чином змінити пов'язані дані.
- Тригери можна розглядати як свого роду фільтри, які вступають в дію після виконання всіх операцій відповідно до правил, стандартних значень і т.д.

Оголошення тригера в стандарті SQL

CREATE TRIGGER <ім’я тригера>

BEFORE|AFTER <операції над табл>
[OF <список полів>]

ON <ім’я табл> [WHEN <умова>]

[BEGIN ATOMIC] <оператори SQL>
[END];

Оголошення тригера в стандарті SQL (пояснення)

- Якщо умова у виразі WHEN є істинною або цей вираз відсутній, то до (BEFORE) або після (AFTER) виконання операції INSERT, UPDATE чи DELETE над таблицею, зазначеною після слова ON, буде виконано вказані SQL-оператори. Коли такий SQL-оператор один, то ключові слова BEGIN та END не використовуються.
- Конструкція
  - BEFORE | AFTER <операції над табл> [OF <список полів>] ON <ім'я табл> називається ***реченням ініціювання***,
  - WHEN <умова> – ***умовою ініціювання***,
  - BEGIN <оператори SQL> END – ***дією тригера***.

Типи тригерів в MS SQL Server

- **Тригери DML**. Їх дія відбувається при виконанні для таблиці якої-небудь інструкції мови маніпулювання даними (DML).
- **Тригери DDL**. Спрацьовують у відповідь на ряд подій мови визначення даних (DDL). Ці події насамперед відповідають інструкціям Transact-SQL CREATE, ALTER, DROP, GRANT, DENY, REVOKE і деяким системним збереженим процедурам, які виконують схожі з DDL операції.
- **Тригери входу** можуть спрацьовувати у відповідь на подію LOGON, що виникає при налаштування користувальницьких сеансів.

Оголошення DML-тригера в MS SQL Server

```
CREATE TRIGGER [ schema_name.]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
    { FOR | AFTER | INSTEAD OF }
    { [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
     [ WITH APPEND ]
    [ NOT FOR REPLICATION ]
    AS { sql_statement  [ ; ] [ ,...n ] |
         EXTERNAL NAME <method specifier [ ; ] >
          }
```

Оголошення DML-тригера в MS SQL Server (пояснення)

- Конструкції { FOR | AFTER | INSTEAD OF }
  { [INSERT] [,] [UPDATE] [,] [DELETE] } визначають, на яку команду буде реагувати тригер. При його оголошенні необхідно вказати хоча б одну команду. Допускається створення тригера, який реагує на дві або на усі три команди.
- Аргумент WITH APPEND дозволяє створювати декілька тригерів кожного типу.
- При створенні тригера з аргументом NOT FOR REPLICATION забороняється його запуск під час виконання модифікації таблиць механізмами реплікації.
- Конструкція AS sql_оператор[...n] визначає набір SQL-операторів і команд, які будуть виконуватись при запуску тригера.
  - У SQL Server тригери можуть бути створені безпосередньо з інструкцій Transact-SQL або з методів збірок, створених в середовищі CLR платформи.NET Framework, і передані на екземпляр SQL Server.

Два параметри, які визначають поведінку DML-тригерів

1. **AFTER-тригер** виконується **після** успішного виконання команд, які його викликали. Якщо ж команди з якоїсь причини не можуть бути успішно завершені, тригер не виконується.
- Можна визначити кілька AFTER-тригерів для кожної операції (INSERT, UPDATE, DELETE).
- Якщо для таблиці передбачено виконання декількох AFTER-тригерів, то за допомогою системної збереженої процедури ***sp_settriggerorder*** можна вказати, який з них буде виконуватися першим, а який останнім.
- За замовчуванням в SQL Server всі тригери є AFTER-тригерами.

Два параметри, які визначають поведінку DML-тригерів

2. **INSTEAD OF** -тригер викликається **замість** виконання інструкцій DML.
- На відміну від AFTER-тригера INSTEAD OF-тригер може бути визначений як для таблиці, так і для представлення (view).
- Для кожної операції INSERT, UPDATE, DELETE можна визначити тільки один INSTEAD OF-тригер.

Обмеження при визначенні DML-тригерів

- Інструкція CREATE TRIGGER повинна бути першою в пакеті і може застосовуватись лише до однієї таблиці.
- Тригер створюється лише в поточній базі даних, але може містити посилання на об’єкти за межами поточної бази даних.
- Створює тригер тільки власник бази даних. Це обмеження дозволяє уникнути випадкового зміни структури таблиць, способів зв'язку з ними інших об'єктів.
- Тригери INSTEAD OF DELETE / UPDATE не можна визначити для таблиці, в якої є зовнішній ключ, визначений для каскадного виконання операції DELETE / UPDATE.

Оголошення DDL-тригера в MS SQL Server

```
CREATE TRIGGER trigger_name
ON { ALL SERVER | DATABASE }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { event_type |
    event_group } [ ,...n ]
AS { sql_statement  [ ; ] [ ,...n ] |
    EXTERNAL NAME < method specifier >
    [ ; ] }
```

Оголошення тригера входу в MS SQL Server

```
CREATE TRIGGER trigger_name
ON ALL SERVER
[ WITH <logon_trigger_option> [ ,...n ] ]
{ FOR | AFTER } LOGON
AS { sql_statement  [ ; ] [ ,...n ] |
    EXTERNAL NAME <method specifier>[;]
      }
```

Тригери і транзакції

Дії, які виконуються під час основної операції та в тригері, становлять єдину транзакцію:

- Перед виконанням операцій модифікації даних неявно ініціюється команда початку транзакції.
- Реалізується операція модифікації даних.
- Ініціюється та виконується тригер.
- Тригер скасовує транзакцію або за замовчуванням його дія завершується.

Обмеження на використання тригерів

- Основна перевага тригерів – стандартні функції зберігаються всередині бази даних і злагоджено активізуються при кожному її оновленні. Це може істотно спростити додатки роботи з БД;
- у деяких випадках тригери не визначаються для віртуальних таблиць;
- після видалення таблиці усі зв’язані з нею тригери також видаляються;
- тригери визначаються лише для створених таблиць.

Недоліки тригерів

- Використання тригерів пов'язане з додатковими витратами ресурсів на операції введення / виведення.
- У тому випадку, коли таких же результатів (з набагато меншими витратами ресурсів) можна домогтися за допомогою збережених процедур або прикладних програм, застосування тригерів недоцільно.
- Складність: при переміщенні деяких функцій в базу даних ускладнюються завдання її проектування, реалізації та адміністрування.
- Неправильно написані тригери можуть призвести до "мертвих" блокувань. Тригери здатні тривалий час блокувати безліч ресурсів, тому варто звернути особливу увагу на зведення до мінімуму конфліктів доступу.

Недоліки тригерів

- Прихована функціональність: перенесення частини функцій в базу даних і збереження їх у вигляді одного або декількох тригерів іноді призводить до приховування від користувача деяких функціональних можливостей. Хоча це певною мірою спрощує його роботу, але, на жаль, може стати причиною незапланованих, потенційно небажаних і шкідливих побічних ефектів, оскільки в цьому випадку користувач не в змозі контролювати всі процеси, що відбуваються в базі даних
- Вплив на продуктивність: перед виконанням кожної команди по зміні стану бази даних СУБД повинна перевірити тригерну умову, щоб з'ясувати необхідність запуску тригера для цієї команди. Виконання подібних обчислень позначається на загальній продуктивності СУБД, а в моменти пікового навантаження її зниження може стати особливо помітним. Очевидно, що при зростанні кількості тригерів збільшуються і ресурсні витрати, пов'язані з такими операціями.

Дякую за увагу

Опрацювати: Д.Петковіч «Microsoft SQL Server 2012. Руководство для начинающих» Глава 8,14