Wydział Elektryczny

POLITECHNIKA WARSZAWSKA

1DI1532:A/Z1 - Zapisywanie oraz odtwarzanie kluczy obcych z realizowanego ich back’upu

Sprawozdanie

Maciej Czarkowski, nr indeksu 292810, 27.10.2020

Opis problemu

Pierwszym zadaniem realizowanym w ramach laboratorium z Administrowania Bazami Danych jest umożliwienie tworzenie kopii kluczy obcych założonych na bazie danych, tak, aby mogły być one usunięte na czas np. wgrywania nowej bazy danych bądź *dogrywania* danych, a następnie ponowne założone na bazie. Zależności budowane przez klucze obce niekiedy powodują problemy i konflikty podczas wspomnianych działań, dlatego też możliwość ich tymczasowego usunięcia, a następnie odtworzenia, jest niezwykle komfortową opcją dla osób zarządzających bazami danych, które nie muszą się przejmować *constraintami*.

Tworzenie backupu kluczy, czy liczby wierszy, wraz z zapisem konkretnych stanów jest również dobrym narzędziem statystycznym i diagnostycznym, które pozwala monitorować stan naszej bazy, ilość rekordów w niej, co niekiedy może zwrócić uwagę na niepokojące procesy, np. zbyt szybkie *puchnięcie* bazy.

Opis funkcjonalny

Zrealizowanie zadania zakłada udostępnienie funkcjonalności polegającej na możliwości zapisu kluczy do tabeli w bazie `DB_STAT`, wraz z indeksem zapisu oraz datą jego utworzenia. Umożliwi to później identyfikowanie stanu danych kluczy w zależności od daty.

Opis realizacji zadania

Zadanie zostało zrealizowane z wykorzystaniem procedur składowanych w bazie danych `DB_STAT`. Stworzone przeze mnie, oraz wykorzystane procedury, to `dbo.DB_TC_STORE` -- procedura pozwalająca na zapis kluczy obcych z wybranej bazy do `DB_STAT.dbo.DB_FK` oraz zapis liczby rekordów w każdej z tabel w bazie do `DB_STAT.dbo.DB_RCOUNT` (procedura tworzy wpis w `DB_STAT.dbo.DB_STAT` - na podstawie `id_stat` jesteśmy wtedy w stanie określić dla którego zrzutu danych do tabeli określona jest liczba wierszy w tabelach oraz spis kluczy obych w wybranej bazie), `dbo.DB_TC_DELETE_KEYS` -- procedurę pozwalającą na usunięcie kluczy obcych z wybranej bazy oraz `dbo.DB_TC_ADD_KEYS`, czyli procedurę, która pozwala odtworzyć klucze w bazie na podstawie wybranego `stat_id` (lub ostatniego utworzonego zrzutu kluczy dla danej bazy, jeśli argument ten nie został podany).

Przy realizacji procedur skorzystałem z kursorów, które pozwoliły mi na przechodzenie po kolejnych rekordach tabel. Istotne były również parametry procedur, które pozwoliły na zapis danych z konkretnej bazy.

```
    /* kursor po wszystkich tabelach uzytkownika */
    DECLARE CC INSENSITIVE CURSOR FOR
            SELECT o.[constraint], o.[table]
                FROM #TC o

    OPEN CC
    FETCH NEXT FROM CC INTO @const, @tab
    WHILE (@@FETCH_STATUS = 0)
    BEGIN
        SET @sql = 'ALTER TABLE ' + LTRIM(@db) + '.dbo.' + LTRIM(@tab) + ' DROP CONSTRAINT ' + LTRIM(@const)
        --SELECT @sql
        EXEC sp_sqlexec @sql
        /* Przechodzimy do następnego wiersza */
        FETCH NEXT FROM CC INTO @const, @tab
    END
    CLOSE CC
    DEALLOCATE CC
GO
```

Rys. 1 Przykładowa pętla do usuwania kolejnych kluczy

W powyższym przykładzie skorzystałem z kursora oraz zmiennych w procedurze, a także z tabel tymczasowych. Pozwoliły one skonstruować odpowiednie zapytania usuwające kolejne klucze z bazy. Analogicznie, z pętli skorzystałem również przy dodawaniu kluczy.

Dokumentacja działania procesu

Na poniższych zrzutach ekranu udokumentowałem działanie stworzonych procedur i całego procesu usunięcia, a następnie odtworzenia kluczy.

```
    /* przykładowe zapytanie dla kluczy obcych na wybranej bazie */
    /*
    USE PWX_DB_Z1

    SELECT
        f.name constraint_name
    ,OBJECT_NAME(f.parent_object_id) referencing_table_name
    ,COL_NAME(fc.parent_object_id, fc.parent_column_id) referencing_column_name
    ,OBJECT_NAME (f.referenced_object_id) referenced_table_name
    ,COL_NAME(fc.referenced_object_id, fc.referenced_column_id) referenced_column_name
        FROM sys.foreign_keys AS f
        JOIN sys.foreign_key_columns AS fc
        ON f.[object_id] = fc.constraint_object_id
        ORDER BY f.name
    */
END
GO
```

Results | Messages

constraint_name	referencing_table_name	referencing_column_name	referenced_table_name	referenced_column_name
fk_etaty__firmy	etaty	id_firmy	firmy	nazwa_skr
fk_etaty__osoby	etaty	id_osoby	osoby	id_osoby
fk_firmy__miasta	firmy	id_miasta	miasta	id_miasta
FK_FIRMY_CECHY__WARTOSCI_CECH	FIRMY_CECHY	id_wartosci	WARTOSCI_CECH	id_wartosci
fk_miasta__woj	miasta	kod_woj	woj	kod_woj
fk_osoby__miasta	osoby	id_miasta	miasta	id_miasta
FK_WARTOSCI_CECHY__CECHY	WARTOSCI_CECH	id_CECHY	CECHY	id_CECHY

Rys. 2 Wyświetlenie kluczy przed procesem

```
EXEC DB_STAT.dbo.DB_TC_STORE @commt = 'dump_test_superowy', @db = N'pwx_db_Z1'

USE DB_STAT
GO

EXEC DB_STAT.dbo.DB_TC_DELETE_KEYS @db = 'pwx_db_Z1'
```

Messages

```
(1 row affected)

(8 rows affected)

(1 row affected)
```

Rys. 3 Uruchomienie procedury zapisu kluczy i liczby rekordów

```
SELECT * FROM DB_STAT
/*stat_id     db_nam              comment             when                    usr_nam
----------- ------------------- ------------------- ----------------------- ---------------
1           pwx_db              test                2020-10-15 11:18:22.167 dbo
2           pwx_db              test                2020-10-15 11:50:32.983 dbo
```

Results Messages

stat_id	db_nam	comment	when	usr_nam	host
4	pwx_db_Z1	dump2	2020-10-25 15:07:18.897	dbo	DESKTOP-4SJHTLT
5	pwx_db_Z1	dump_test	2020-10-25 15:16:04.060	dbo	DESKTOP-4SJHTLT
6	pwx_db_Z1	dump_test2	2020-10-25 15:59:25.543	dbo	DESKTOP-4SJHTLT
7	pwx_db_Z1	dump_test3	2020-10-25 16:17:40.557	dbo	DESKTOP-4SJHTLT
8	pwx_db_Z1	dump_test4	2020-10-25 16:20:09.470	dbo	DESKTOP-4SJHTLT
9	pwx_db_Z1	dump_test5	2020-10-25 16:20:31.987	dbo	DESKTOP-4SJHTLT
10	pwx_db	dump_test5	2020-10-25 22:08:08.987	dbo	DESKTOP-4SJHTLT
11	pwx_db_Z1	dump_test10	2020-10-27 09:31:46.707	dbo	DESKTOP-4SJHTLT
12	pwx_db_Z1	dump_test_superowy	2020-10-27 18:24:41.510	dbo	DESKTOP-4SJHTLT

Rys.4 Zrzut danych widoczny jest w tabeli z właściwym komentarzem (`stat_id`=12)

```
SELECT * FROM DB_STAT.dbo.DB_RCOUNT where stat_id = 12

/*stat_id     table
----------- ------------------------------------------------------------
3           A4
```

Results Messages

stat_id	table	RCOUNT	RDT
12	CECHY	2	2020-10-27 18:24:43.957
12	etaty	10	2020-10-27 18:24:43.967
12	firmy	3	2020-10-27 18:24:43.970
12	FIRMY_CECHY	6	2020-10-27 18:24:43.977
12	miasta	5	2020-10-27 18:24:43.980
12	osoby	6	2020-10-27 18:24:43.993
12	WARTOSCI_CECH	5	2020-10-27 18:24:44.000
12	woj	3	2020-10-27 18:24:44.003

Rys. 5 Widzimy, że liczby rekordów zapisały się poprawnie

```
395
396 SELECT * FROM DB_STAT.dbo.DB_FK where stat_id = 12
397
398 /* mozna zrobić kursor po bazach i w petli wołąc procedurę i mieć zrzut dla wszystkich baz */
399 SELECT d.name FROM sys.databases d WHERE d.database_id > 4 -- ponizej 5 są systemowe
400
401 /* Można stworzyć procedurę do przechowywania liczby wierszy w KAZDEJ !!! bazie */
```

Results Messages

stat_id	constraint_name	referencing_table_name	referencing_column_name	referenced_table_name	referenced_column_name
12	fk_miasta__woj	miasta	kod_woj	woj	kod_woj
12	fk_osoby__miasta	osoby	id_miasta	miasta	id_miasta
12	fk_firmy__miasta	firmy	id_miasta	miasta	id_miasta
12	fk_etaty__osoby	etaty	id_osoby	osoby	id_osoby
12	fk_etaty__firmy	etaty	id_firmy	firmy	nazwa_skr
12	FK_WARTOSCI_CECHY__CECHY	WARTOSCI_CECH	id_CECHY	CECHY	id_CECHY
12	FK_FIRMY_CECHY__WARTOSCI_CECH	FIRMY_CECHY	id_wartosci	WARTOSCI_CECH	id_wartosci

Rys. 6 Klucze również zostały zapisane poprawnie

```
354 EXEC DB_STAT.dbo.DB_TC_DELETE_KEYS @db = 'pwx_db_Z1'
355
356 EXEC DB_STAT.dbo.DB_TC_ADD_KEYS @db = 'pwx_db_Z1'
357
358 SELECT * FROM DB_STAT
359 /*stat_id     db_nam              comment             when
```

Results Messages

```
(7 rows affected)

(7 rows affected)
```

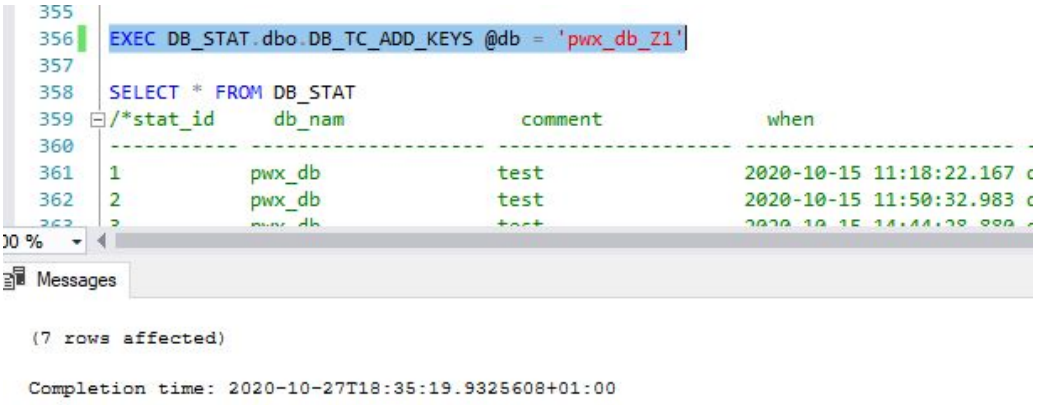
Rys. 7 Usuwam klucze procedurą `DB_TC_DELETE_KEYS`

```
125     /*
126     USE PWX_DB_Z1
127
128     SELECT
129         f.name constraint_name
130     ,OBJECT_NAME(f.parent_object_id) referencing_table_name
131     ,COL_NAME(fc.parent_object_id, fc.parent_column_id) referencing_column_name
132     ,OBJECT_NAME (f.referenced_object_id) referenced_table_name
133     ,COL_NAME(fc.referenced_object_id, fc.referenced_column_id) referenced_column_name
134         FROM sys.foreign_keys AS f
135         JOIN sys.foreign_key_columns AS fc
136         ON f.[object_id] = fc.constraint_object_id
137         ORDER BY f.name
138     */
139 END
140  GO
```

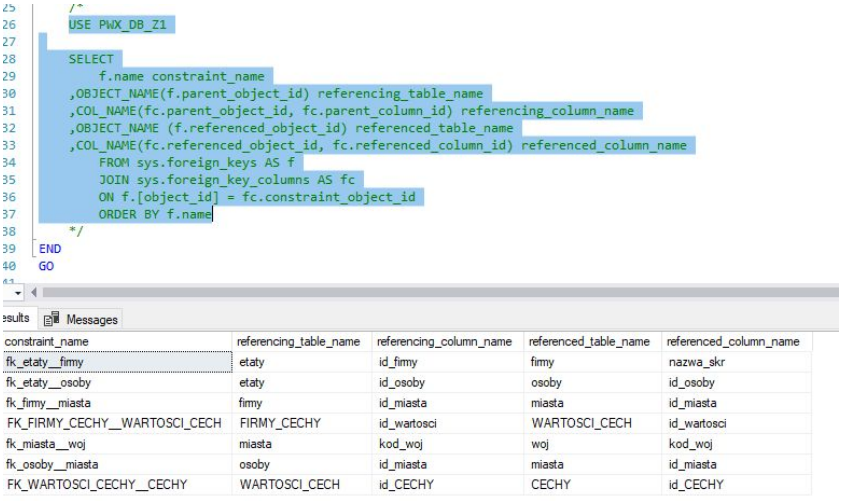
Results Messages

constraint_name	referencing_table_name	referencing_column_name	referenced_table_name	referenced_column_name

Rys. 8 Widzimy, że po usunięciu w bazie nie widać żadnych kluczy



Rys. 9 Dodaję kluczę (nie podałem argumentu id -- klucze będą odtworzone zgodnie z ostatnim zrzutem dla danej bazy)



Rys. 10 Klucze zostały poprawnie odtworzone -- znów są widoczne

Opis użytkowania

W pierwszej kolejności należy zapisać klucze z danej bazy -- korzystamy z procedury `dbo.DB_TC_STORE` (przykładowe wywołanie rys. 2). Wywołujemy tę procedurę z dwoma argumentami -- komentarzem (opcjonalnie) i nazwą bazy. Następnie, należy usunąć klucze przy pomocy procedury `dbo.DB_TC_DELETE_KEYS`, gdzie argumentem jest baza, dla której klucze mają zostać usunięte (przykładowe wywołanie rys. 7).

Po skończonych działaniach związanych np. z importem nowych danych, możemy przywrócić klucze wykorzystując wybrany `stat_id` z tabeli `DB_STAT`, czyli z którego zrzutu danych klucze chcemy wykorzystać (przykładowe wywołanie rys. 9). Opcjonalnie, możemy jako argument podać `@id`, określi nam ono wspomniane

`stat_id`, które będzie wykorzystane. Jako argument `@db` podajemy również bazę danych, w której klucze przywracamy.

Dane dotyczące zapisywanych w `DB_STAT` kluczy oraz liczby rekordów, odnajdziemy wraz z ich `stat_id` w tabelach odpowiednio `DB_FK` oraz `DB_RCOUNT` w bazie `DB_STAT`.

Wnioski

Opisane kroki pozwalają na dogodne zapisywanie, usuwanie oraz dodawanie kluczy na podstawie wybranego ich zrzutu. Poza przydatnością w kwestiach zarządzania bazą np. w momencie importu nowych danych, są przydatne w celach czysto statystycznych. Możemy uruchomić oprogramowanie takie jak SQLAgent, a w nim zadanie uruchamiające cyklicznie opisane procedury. Pozwala to na kontrolę liczby rekordów w bazie oraz *constraintów,* co pozwala zwrócić uwagę na podejrzane zachowania bazy danych czy niekontrolowane jej *rośnięcie*.