



**Wydział
Elektryczny**

POLITECHNIKA WARSZAWSKA

1DI1532:A/Z4 - Tworzenie indeksów dla kluczy obcych

Sprawozdanie

Opis problemu

Czwartym zadaniem realizowanym w ramach laboratorium z Administrowania Bazami Danych jest umożliwienie tworzenia indeksów na kluczach obcych w tabeli, tak aby przyspieszyć wiele wyszukiwań, które w znaczącej większości opierają się na wyszukiwaniu po kluczach obcych. Dodanie takich indeksów pozwala na zmniejszenie czasów wyszukiwań elementów w tabelach, co może mieć szczególne znaczenie przy bardzo dużych bazach danych, co czyni to zadanie bardzo przydatnym jeśli chodzi o wydajność pracy administratora bazy danych.

Opis funkcjonalny

Zrealizowanie zadania zakłada udostępnienie funkcjonalności polegającej na możliwości tworzenia indeksów na kluczach obcych w tabeli z uwagą aby nie dublować indeksów przy kilkukrotnym wywołaniu. Całość powinna być realizowana w formie procedury. Procedura powinna przyjmować nazwę bazy danych jako argument specyfikujący, w której bazie chcemy uzupełnić klucze obce.

Opis realizacji zadania

Do realizacji zadania utworzyłem bazę `z4_baza`. Utworzyłem ją z wykorzystaniem skryptu, który otrzymaliśmy na początku zajęć (bazy z tabelami `firmy`, `etaty`, `faktury`, `miasta` etc.). W bazie tej występuje kilka kluczów obcych, które idealnie nadają się do testów stworzonej procedury `dbo.create_missing_indexes`. **Kod tej procedury udostępniony jest na końcu sprawozdania.**

Początkowo, sprawdzam dostępne utworzone już w bazie indeksy, przeglądając rekordy z tabeli `sys.indexes`.

	object_id	name	index_id	type	type_desc	is_unique	data_space_id	ignore_dup_key	is_primary_key	is_unique_constraint	fill_factor	is_padder
168	15151...	NULL	0	0	HEAP	0	1	0	0	0	0	0
169	15311...	_idx_trusted_assemblies	1	1	CLUSTERED	1	1	0	0	0	0	0
170	15471...	idx_server_resource_stats	1	1	CLUSTERED	1	1	0	0	0	0	0
171	15631...	IX_external_library_setup_failures	1	1	CLUSTERED	1	1	0	0	0	0	0
172	15916...	pk_woj	1	1	CLUSTERED	1	1	0	1	0	0	0
173	16236...	pk_miasta	1	1	CLUSTERED	1	1	0	1	0	0	0
174	16716...	pk_osoby	1	1	CLUSTERED	1	1	0	1	0	0	0
175	17196...	pk_firmy	1	1	CLUSTERED	1	1	0	1	0	0	0
176	17676...	pk_etaty	1	1	CLUSTERED	1	1	0	1	0	0	0
177	18031...	NULL	0	0	HEAP	0	1	0	0	0	0	0
178	18316...	PK_CECHY	1	1	CLUSTERED	1	1	0	1	0	0	0
179	18796...	PK_WARTOSCI_CECH	1	1	CLUSTERED	1	1	0	1	0	0	0
180	19276...	PK_FIRMY_CECHY	1	1	CLUSTERED	1	1	0	1	0	0	0
181	21071...	NULL	0	0	HEAP	0	1	0	0	0	0	0

Rys.1 Dostępne domyślnie indeksy w `sys.indexes`

Jak widzimy, dostępne są indeksy dla kluczy *primary*, co jest domyślnym zachowaniem dla większości silników baz danych i przydatnym elementem przy wyszukiwaniu bezpośrednio po kluczu głównym.

Zanim dodamy indeksy na klucze obce, warto podejrzeć sobie je wszystkie, wykorzystując zapytanie `SELECT` odnoszące się do tabel `sys.foreign_keys` i `sys.foreign_key_columns`. Efekt jest następujący:

```

26      + N'INSERT INTO #FK
27      SELECT f.name,
28      f.[object_id],
29      OBJECT_NAME(f.parent_object_id) referencing_table_name,
30      COL_NAME(fc.parent_object_id, fc.parent_column_id) referencing_column_name
31      FROM sys.foreign_keys f
32      JOIN sys.foreign_key_columns AS fc
33      ON f.[object_id] = fc.constraint_object_id'
34
35      EXEC sp_sqlexec @sql
36
37      CREATE TABLE #IndexedFK ([object_id] int)

```

100 %

Results Messages

	name	object_id	referencing_table_name	referencing_column_name
1	fk_miasta_woj	661577395	miasta	kod_woj
2	fk_osoby_miasta	709577566	osoby	id_miasta
3	fk_firmy_miasta	757577737	firmy	id_miasta
4	fk_etaty_osoby	805577908	etaty	id_osoby
5	fk_etaty_firmy	821577965	etaty	id_firmy
6	FK_WARTOSCI_CECHY_CECHY	917578307	WARTOSCI_CECH	id_CECHY
7	FK_FIRMY_CECHY_WARTOSCI_CECH	965578478	FIRMY_CECHY	id_wartosci

Rys.2 Wszystkie klucze obce dostępne w mojej bazie

W celu przetestowania, czy późniejsze wykonanie stworzonej przeze mnie procedury nie zdubluje indeksów i utworzy je tylko na tych kluczach obcych, dla których nie istnieją klucze obce oraz sprawdzenia czy indeksy dodają się poprawnie, jeden z nich dodaję ręcznie (z wykorzystaniem `CREATE INDEX`) po czym sprawdzam czy pojawił się w `sys.indexes`.

```

72
73      CREATE INDEX ifk_miasta_woj ON miasta (kod_woj)
74      SELECT * FROM sys.indexes

```

100 %

Results Messages

	object_id	name	index_id	type	type_desc	is_unique	data_space
148	30957...	NULL	0	0	HEAP	0	1
149	32557...	NULL	0	0	HEAP	0	1
150	56557...	wpr_bucket_clustered_idx	1	1	CLUSTERED	1	1
151	59757...	pk_woj	1	1	CLUSTERED	1	1
152	62957...	pk_miasta	1	1	CLUSTERED	1	1
153	62957...	ifk_miasta_woj	2	2	NONCLUSTERED	0	1
154	67757...	pk_osoby	1	1	CLUSTERED	1	1
155	72557...	pk_firmy	1	1	CLUSTERED	1	1
156	77357...	pk_etaty	1	1	CLUSTERED	1	1
157	83757...	PK_CECHY	1	1	CLUSTERED	1	1
158	88557...	PK_WARTOSCI_CECH	1	1	CLUSTERED	1	1
159	93357...	PK_FIRMY_CECHY	1	1	CLUSTERED	1	1
160	10030...	group clustered index	1	1	CLUSTERED	1	1

Rys.3 Testowe utworzenie indeksu `ifk_miasta_woj` i sprawdzenie jego obecności w `sys.indexes`

Jak widzimy, nowy indeks znalazł się w tabeli, dodanie indeksu odbyło się pomyślnie. Możemy dodać teraz indeksy dla pozostałych kluczy, zwracając uwagę czy się nie zdublują dla klucza obcego `fk_miasta_woj`.

Procedurę wywołujemy specyfikując jako parametr nazwę bazy, dla której chcemy tworzyć indeksy na kluczach obcych.

```
EXEC dbo.create_missing_indexes 'z4_baza'
```

Po wykonaniu możemy znów sprawdzić w `sys.indexes` czy indeksy się nie zdublowały oraz czy się poprawnie utworzyły.

149	32557...	NULL	0	0	HEAP	0	1	0	0
150	56557...	wpr_bucket_clustered_idx	1	1	CLUSTERED	1	1	0	0
151	59757...	pk_woj	1	1	CLUSTERED	1	1	0	1
152	62957...	pk_miasta	1	1	CLUSTERED	1	1	0	1
153	62957...	ifk_miasta_woj	2	2	NONCLUSTERED	0	1	0	0
154	67757...	pk_osoby	1	1	CLUSTERED	1	1	0	1
155	67757...	ifk_osoby_miasta	2	2	NONCLUSTERED	0	1	0	0
156	72557...	pk_firmy	1	1	CLUSTERED	1	1	0	1
157	72557...	ifk_firmy_miasta	2	2	NONCLUSTERED	0	1	0	0
158	77357...	pk_etaty	1	1	CLUSTERED	1	1	0	1
159	77357...	ifk_etaty_firmy	2	2	NONCLUSTERED	0	1	0	0
160	77357...	ifk_etaty_osoby	3	2	NONCLUSTERED	0	1	0	0
161	83757...	PK_CECHY	1	1	CLUSTERED	1	1	0	1
162	88557...	PK_WARTOSCI_CECHY	1	1	CLUSTERED	1	1	0	1
163	88557...	ifk_WARTOSCI_CECHY_CECHY	2	2	NONCLUSTERED	0	1	0	0
164	93357...	PK_FIRMY_CECHY	1	1	CLUSTERED	1	1	0	1
165	19930...	queue_clustered_index	1	1	CLUSTERED	1	1	0	0

Rys.4 Sprawdzenie indeksów po wykonaniu procedury

Jak widzimy powyżej, indeksy o nazwach "i" + nazwa_klucza_obcego utworzyły się poprawnie, co oznacza, że **osiągnęliśmy zamierzony efekt, zadanie wykonane!** Testowo puściłem procedurę raz jeszcze, nie zdublowała ona poprzednich indeksów, czyli wszystko jest w porządku.

Aby raz jeszcze przetestować wykonane zadanie wykonałem następujące polecenie, które dało pożądany efekt.

```
SELECT * FROM osoby o WITH(Index(ifk_osoby_miasta))
JOIN miasta m ON (o.id_miasta=m.id_miasta)
```

76	SELECT * FROM osoby o WITH(Index(ifk_osoby_miasta))
77	JOIN miasta m ON (o.id_miasta=m.id_miasta)

	id_osoby	id_miasta	imię	nazwisko	imię_i_nazwisko	id_miasta	kod_woj	nazwa
1	1	1	Maciej	Stodolski	M. Stodolski	1	MAZ	WESOLA
2	2	2	Jacek	Korytkowski	J. Korytkowski	2	MAZ	WARSZAWA
3	5	2	Juz	Niepracujący	J. Niepracujący	2	MAZ	WARSZAWA
4	3	3	Mis	Nieznany	M. Nieznany	3	POM	GDANSK
5	4	4	Król	Neptun	K. Neptun	4	POM	SOPOT

Rys.5 Test wykonanej procedury -- udany

Kod procedury

```

/***** Z4 Maciej Czarkowski 292810 *****/
USE z4_baza
GO
/*****/
-- Dodanie indeksów dla nieistniejących kluczy obcych
IF NOT EXISTS
(
    SELECT 1
    from sysobjects o (NOLOCK)
    WHERE    (o.[name] = 'create_missing_indexes')
    AND      (OBJECTPROPERTY(o.[ID], 'IsProcedure')=1)
)
BEGIN
    DECLARE @stmt nvarchar(100)
    SET @stmt = 'CREATE PROCEDURE dbo.create_missing_indexes AS '
    EXEC sp_sqlexec @stmt
END
GO

ALTER PROCEDURE dbo.create_missing_indexes (@db nvarchar(100))
AS
BEGIN
    DECLARE @sql nvarchar(1000), @table nvarchar(256), @column nvarchar(256), @fk
    nvarchar(256)
    CREATE TABLE #FK ([fk_name] varchar(30), [object_id] int, [table_name] varchar(30),
    [column_name] varchar(30))
    SET @sql = N'USE [' + @db + N']; '
    + N'INSERT INTO #FK
        SELECT f.name,
        f.[object_id],
        OBJECT_NAME(f.parent_object_id) referencing_table_name,
        COL_NAME(fc.parent_object_id, fc.parent_column_id)
referencing_column_name
        FROM sys.foreign_keys f
        JOIN sys.foreign_key_columns AS fc
        ON f.[object_id] = fc.constraint_object_id'

    EXEC sp_sqlexec @sql

    CREATE TABLE #IndexedFK ([object_id] int)

    INSERT INTO #IndexedFK
    SELECT fc.[constraint_object_id]
    FROM sys.foreign_key_columns fc
    JOIN sys.index_columns ic ON fc.[parent_object_id] = ic.[object_id]
    JOIN #FK fk ON fc.constraint_object_id = fk.[object_id]
    WHERE fc.parent_column_id = ic.column_id

    CREATE TABLE #IndexesToAdd([fk_name] varchar(30), [object_id] int, [table_name]
    varchar(30), [column_name] varchar(30))

    -- Wybieram tylko te klucze obce, dla których nie ma jeszcze indeksów
    INSERT INTO #IndexesToAdd
    SELECT * FROM #FK f WHERE f.[object_id] NOT IN (SELECT object_id FROM #IndexedFK)

```

```
DECLARE CC INSENSITIVE CURSOR FOR
    SELECT o.[fk_name], o.[table_name], o.[column_name]
    FROM #IndexesToAdd o
    ORDER BY 1

OPEN CC
FETCH NEXT FROM CC INTO @fk, @table, @column

WHILE (@@FETCH_STATUS = 0)
BEGIN
    SET @sql = 'USE [' + @db + N']; '
    + N'CREATE INDEX i' + @fk + N' ON ' + @table + N' (' + @column + N')'
    EXEC sp_sqlexec @sql

    FETCH NEXT FROM CC INTO @fk, @table, @column
END
CLOSE CC
DEALLOCATE CC

END
GO

EXEC dbo.create_missing_indexes 'z4_baza'
SELECT * FROM sys.foreign_key_columns
SELECT * FROM sys.indexes

SELECT * FROM osoby o WITH(Index(ifk_osoby__miasta))
JOIN miasta m ON (o.id_miasta=m.id_miasta)
```