



**Wydział
Elektryczny**

POLITECHNIKA WARSZAWSKA

**1DI1532:A/Z3 - Triggery, odtwarzanie bazy z
utworzonego wcześniej backupu,
porównywanie baz danych**

Sprawozdanie

Opis problemu

Trzecie, realizowane w ramach laboratorium z Administrowania Bazami Danych, polegało na wykorzystaniu zdobytych wcześniej umiejętności, takich jak tworzenie backupu bazy, do odtworzenia bazy, realizując na niej jednocześnie kilka triggerów, czyli procedur wykonywanych automatycznie przy określonych zdarzeniach. Zadaniem było stworzenie triggerów, które zapamiętują wystawioną fakturę, jak i również uniemożliwiają zmianę numeru faktury czy id_klienta. Należało również uniemożliwić zmianę NIPu u klienta. Na potrzeby zadania stworzono nowe bazy -- administracyjną z logiem oraz bazę z fakturami, klientami i pozycjami. Opisana poniżej realizacja tych zadań pozwala na wszystkie opisane wyżej działania oraz zapewnia pożądaną funkcjonalność. W końcowej części dokumentu umieszczony jest całkowity kod wykorzystany w realizacji zadania, do którego będą pojawiały się odniesienia w niektórych fragmentach dokumentu.

Opis funkcjonalny

Opisany poniżej kod SQL pozwala na realizację założeń z opisu problemu. Utworzenie na bazie wykonanych przeze mnie triggerów, procedur, podążanie z niniejszą instrukcją, pozwala na uniemożliwienie edycji danych, które nie powinny być zmieniane, ale również na rejestrowanie nowych rekordów pojawiających się w bazie. Dodatkowe procedury pozwalają na porównanie aktualnej bazy z logiem, dzięki czemu można zweryfikować czy baza jest aktualna lub czy brakuje nam określonych danych.

Opis realizacji zadania i dokumentacja działania

Tworzenie baz i tabel

Zadanie rozpocząłem od stworzenia baz (pwx_db_Z3 oraz admin_db) i tabel (faktura, klient, pozycje na bazie pwx_db_Z3 oraz LOG_FA na bazie admin_db) potrzebnych do wykonania zadania:

```

1  -- Maciej Czarkowski 292810
2  -- Administrowanie bazami danych Z3
3  /***** Tworzę nową bazę danych pwx_db_Z3 oraz tabele faktura, klient, pozycja na potrzeby zadania *****/
4  IF NOT EXISTS (SELECT d.name
5                  FROM sys.databases d
6                  WHERE (d.database_id > 4)
7                        AND (d.[name] = N'pwx_db_Z3'))
8  )
9  BEGIN
10     CREATE DATABASE pwx_db_Z3
11 END
12 GO
13 /***** Usuwam tabel *****/
14 USE pwx_db_Z3

```

00 % Messages
Commands completed successfully.
Completion time: 2020-11-30T08:26:50.9047686+01:00

Rys.1 Tworzenie nowej bazy danych pwx_db_Z3 na potrzeby zadania

```

31 )
32 BEGIN
33 CREATE TABLE dbo.klient
34 (
35     [id_klienta] int NOT NULL IDENTITY constraint pk_klienta primary key
36     , [NIP] nvarchar(20) NOT NULL
37     , [nazwa] nvarchar(100) NOT NULL
38     , [adres] nvarchar(100) NOT NULL
39 )
40 GO
41
42 -- tabela faktura
43 USE pwx_db_Z3
44 GO
45
46 IF NOT EXISTS
47 (
48     SELECT 1
49     FROM sysobjects o (NOLOCK)
50     WHERE (o.[name] = N'faktura')
51     AND (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
52 )
53 BEGIN
54 CREATE TABLE dbo.faktura
55 (
56     [id_faktury] int NOT NULL IDENTITY constraint pk_faktury primary key
57     , [id_klienta] int NOT NULL constraint pk_fk_klienta foreign key references klient(id_klienta)
58 )
59 GO

```

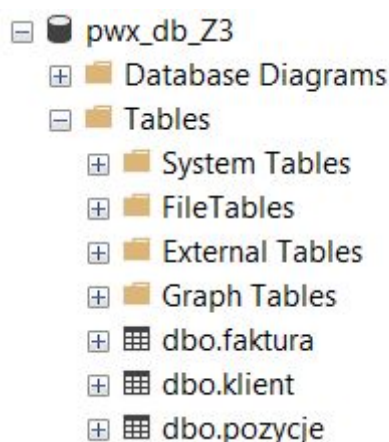
IO %

Messages

Commands completed successfully.

Completion time: 2020-11-30T08:28:06.2035503+01:00

Rys.2 Tworzę nowe tabele w utworzonej bazie



Rys.3 Tabele widoczne w bazie

```

./
.8 -----
.9 /***** Baza administracyjna *****/
!0 -- Tworzę bazę administracyjną
!1 IF NOT EXISTS (SELECT d.name
!2                 FROM sys.databases d
!3                 WHERE (d.database_id > 4)
!4                 AND (d.[name] = N'admin_db'))
!5 )
!6 BEGIN
!7     CREATE DATABASE admin_db
!8 END
!9 GO
!10
!11 USE admin_db
!12 GO

```

Rys.4 Tworzę nową bazę administracyjną admin_db...

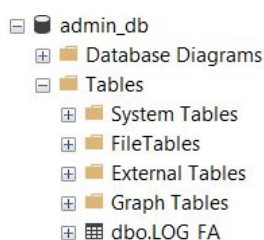
```
136 -- Tworzę tabelę LOG_FA
137 IF NOT EXISTS
138 (
139     SELECT 1
140     from sysobjects o (NOLOCK)
141     WHERE (o.[name] = N'LOG_FA')
142     AND (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
143 )
144 BEGIN
145     CREATE TABLE dbo.[LOG_FA]
146     (
147         [numer_faktury] int not null
148         , [nip_klienta] int not null
149         , [data] datetime not null
150         , [anulowana] bit not null
151     )
152 END
153 GO
154 /***** Triggery
```

Messages

Commands completed successfully.

Completion time: 2020-11-30T08:30:50.3735682+01:00

Rys.5 ...a w niej tabelę LOG_FA



Rys.6 Widoczny efekt poprzednich działań

Tworzenie i testy triggerów

Kolejnym krokiem jest utworzenie triggerów, które pozwolą na m.in. zapisywanie logów w powyższej bazie administracyjnej oraz blokując niepożądane zmiany, które mogłyby unieaktualniać dane przechowywane w bazie.

```
29 CREATE TRIGGER [dbo].[block_client_changes]
30 ON [dbo].[klient]
31 FOR UPDATE
32 AS
33 BEGIN
34     IF EXISTS
35     (
36         SELECT * FROM INSERTED ins
37         JOIN DELETED del on ins.id_klienta = del.id_klienta
38         WHERE ins.NIP != del.NIP
39     )
40     BEGIN
41         ROLLBACK TRANSACTION
42         RAISERROR('BŁĄD! NIE MOŻESZ EDYTOWAĆ NIPU', 11, 1);
43     END
44 END
45 GO
```

Messages

Commands completed successfully.

Completion time: 2020-11-30T17:04:33.7127123+01:00

Rys.7 Trigger na update tabeli klient

```

49 CREATE TRIGGER [dbo].[block_invoice_changes]
50 ON [dbo].[faktura]
51 FOR UPDATE
52 AS
53 BEGIN
54     IF EXISTS
55     (
56         SELECT * FROM inserted ins
57         JOIN deleted del on ins.id_faktury = del.id_faktury
58         WHERE ( ins.numer != del.numer OR ins.id_klienta != del.id_klienta )
59     )
60     BEGIN
61         ROLLBACK TRANSACTION
62         RAISERROR('BŁĄD! NIE MOŻESZ EDYTOWAĆ ID_KLIENTA ORAZ NUMERU FAKTURY', 11, 1);
63     END
64 END
65 GO
66 -- Maciej Czarkowski 292810

```

Messages

Commands completed successfully.

Completion time: 2020-11-30T17:05:33.9524090+01:00

Rys.8 Trigger na update tabeli faktura

```

8 CREATE TRIGGER [dbo].[insert_faktura_trigger]
9 ON faktura
10 AFTER INSERT
11 AS
12 BEGIN
13     INSERT INTO admin_db.dbo.LOG_FA
14     SELECT ins.numer as numer_faktury, k.NIP as nip_klienta, ins.[data], ins.anulowana from
15     INSERTED ins JOIN klient k on ins.id_klienta = k.id_klienta
16 END
17

```

Messages

Commands completed successfully.

Completion time: 2020-11-30T11:00:57.3217180+01:00

Rys.9 Trigger na insert do tabeli faktura

W celu przetestowania triggera na insert wstawiam dane do tabel (wstawiam wiele rekordów na raz -- trigger wykłupuje również taką sytuację, dzięki zastosowanej pętli i obsługuje wszystkie wstawiane rekordy) i sprawdzam czy odpowiednie wpisy pojawiają się w LOG_FA

```

85 USE pwx_db_Z3
86 GO
87
88 DECLARE @id_mc int, @id_td int, @id_ms int, @id_mm int, @id_kd int
89
90 INSERT INTO klient VALUES ('7876537', 'Maciej Kasztanabe', 'Tbilisi')
91 SET @id_mc = SCOPE_IDENTITY()
92
93 INSERT INTO klient VALUES ('1247895', 'Tomasz Duszanski', 'Kutaisi')
94 SET @id_td = SCOPE_IDENTITY()
95
96 INSERT INTO klient VALUES ('0214657', 'Monika Sanzoghlu', 'Moskwa')
97 SET @id_ms = SCOPE_IDENTITY()
98
99 INSERT INTO klient VALUES ('0265444', 'Maria Maren', 'Jakuck')
100 SET @id_mm = SCOPE_IDENTITY()
101
102 INSERT INTO klient VALUES ('9874698', 'Karen Daniel', 'Houston')
103 SET @id_kd = SCOPE_IDENTITY()
104
105
106 INSERT INTO faktura VALUES
107 (@id_mc, convert(datetime, '20070101', 112), 1027, 0),
108 (@id_mc, convert(datetime, '20150101', 112), 1003, 0),
109 (@id_ms, convert(datetime, '20180101', 112), 7622, 0),

```

Results

	id_faktury	id_klienta	data	numer	anulowana
1	10	5	2021-02-25 00:00:00.000	2318	0
2	9	4	2020-01-14 00:00:00.000	9877	0
3	8	3	2020-04-01 00:00:00.000	5432	0
4	7	3	2019-11-21 00:00:00.000	1234	0
5	6	3	2020-07-08 00:00:00.000	1232	0
6	5	2	2020-03-01 00:00:00.000	7863	0
7	4	2	2011-01-01 00:00:00.000	1222	0

Rys.10 Uzupełniam bazę pwx_db_Z3 danymi

```

365  /*****
366  -- Zapytania pomocnicze do testów
367  SELECT * FROM admin_db.dbo.LOG_FA
368  SELECT * FROM pwx_db_Z3.dbo.faktura

```

100 %

Results Messages

	numer_faktury	nip_klienta	data	anulowana
1	7622	7876537	2018-01-01 00:00:00.000	0
2	1003	7876537	2015-01-01 00:00:00.000	0
3	1027	7876537	2007-01-01 00:00:00.000	0
4	7863	1247895	2020-03-01 00:00:00.000	0
5	1222	1247895	2011-01-01 00:00:00.000	0
6	5432	214657	2020-04-01 00:00:00.000	0
7	1234	214657	2019-11-21 00:00:00.000	0

Rys.11 Widoczny jest efekt działania triggera na insert -- dodane faktury pojawiają się w tabeli LOG_FA w bazie administracyjnej admin_db

Kolejnym krokiem były testy triggerów update'ujących dane, które nie powinny ulegać zmianom. Poniżej prezentuję testy, które potwierdzają działanie triggerów -- dane, które nie powinny być zmienione, pomimo próby ich zmiany, zostają takie same.

Test triggera na update klienta:

```

4  SELECT * FROM admin_db.dbo.LOG_FA
5  SELECT * FROM faktura
6  SELECT * FROM klient
7

```

100 %

Results Messages

	id_klienta	NIP	nazwa	adres
1	1	7876537	Maciej Kasztanabe	Tbilisi
2	2	1247895	Tomasz Duszanski	Kutaisi
3	3	9214657	Monika Sarzoghlu	Moskwa
4	4	1265444	Maria Maren	Jakuck
5	5	9874698	Karen Daniel	Houston

Rys.12 Tabela klienci przed próbą update'u

```

12
13  UPDATE [dbo].klient SET NIP=1111111 WHERE id_klienta=1
14  GO

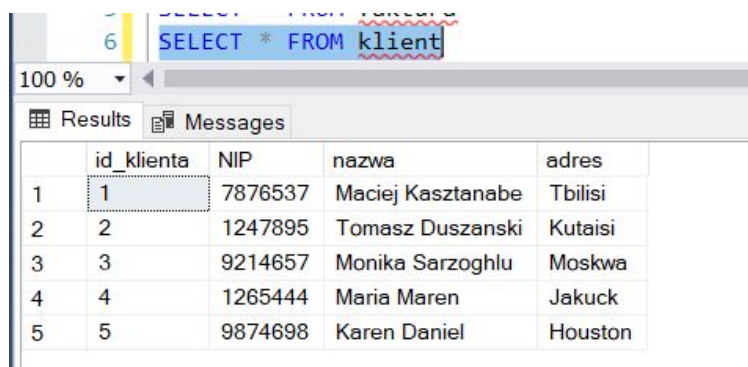
```

100 %

Messages

Msg 50000, Level 11, State 1, Procedure block_client_changes, Line 13 [Batch Start Line 12]
 BŁĄD! NIE MOŻESZ EDYTOWAĆ NIPU
 Msg 3609, Level 16, State 1, Line 13
 The transaction ended in the trigger. The batch has been aborted.
 Completion time: 2020-11-30T16:45:35.4755839+01:00

Rys.13 Uruchamiam update, który powinien zmienić NIP klienta gdyby nie było triggera, a efektem jest komunikat o błędzie, który informuje o tym, że nie można zmieniać numeru NIP



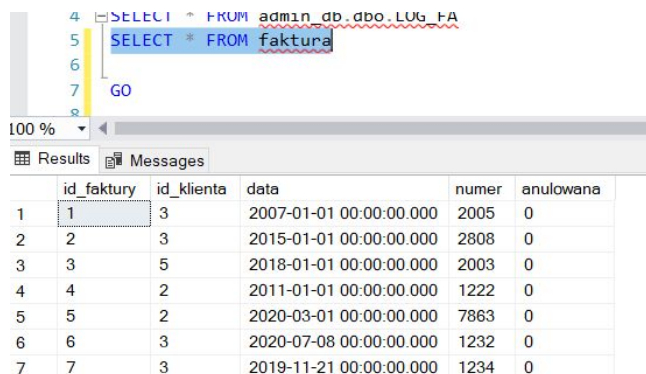
```
SELECT * FROM klient
```

	id_klienta	NIP	nazwa	adres
1	1	7876537	Maciej Kasztanabe	Tbilisi
2	2	1247895	Tomasz Duszanski	Kutaisi
3	3	9214657	Monika Sarzoghlu	Moskwa
4	4	1265444	Maria Maren	Jakuck
5	5	9874698	Karen Daniel	Houston

Rys.14 Efekt po próbie update'u

Efekt jest zgodny z oczekiwanym -- pomimo próby zmiany NIPu dla klienta Maciej Kasztanabe, pozostaje on niezmienny, zgodnie z założeniem działania triggera, który miał wychwytywać i blokować takie sytuacje.

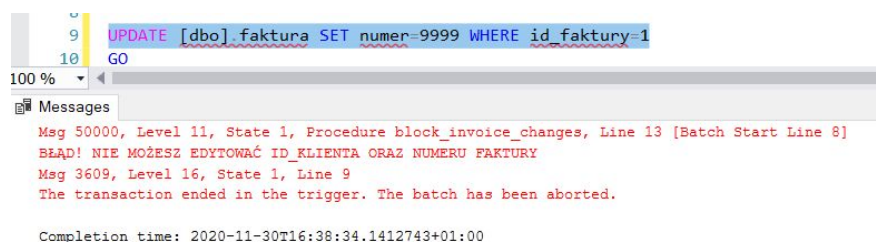
Kolejnym krokiem są testy triggera na update na tabeli faktura -- w tym celu spróbuję dokonać zmian w tej tabeli.



```
SELECT * FROM faktura
```

	id_faktury	id_klienta	data	numer	anulowana
1	1	3	2007-01-01 00:00:00.000	2005	0
2	2	3	2015-01-01 00:00:00.000	2808	0
3	3	5	2018-01-01 00:00:00.000	2003	0
4	4	2	2011-01-01 00:00:00.000	1222	0
5	5	2	2020-03-01 00:00:00.000	7863	0
6	6	3	2020-07-08 00:00:00.000	1232	0
7	7	3	2019-11-21 00:00:00.000	1234	0

Rys.15 Tabela faktura przed próbą update'u

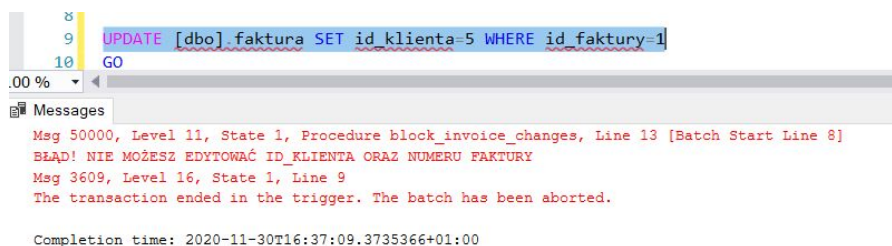


```
UPDATE [dbo].[faktura] SET numer=9999 WHERE id_faktury=1
GO
```

Msg 50000, Level 11, State 1, Procedure block_invoice_changes, Line 13 [Batch Start Line 8]
 BŁĄD! NIE MOŻESZ EDYTOWAĆ ID_KLIENTA ORAZ NUMERU FAKTURY
 Msg 3609, Level 16, State 1, Line 9
 The transaction ended in the trigger. The batch has been aborted.

Completion time: 2020-11-30T16:38:34.1412743+01:00

Rys.16 Próba update'u numeru zakończona niepowodzeniem i informacją o błędzie

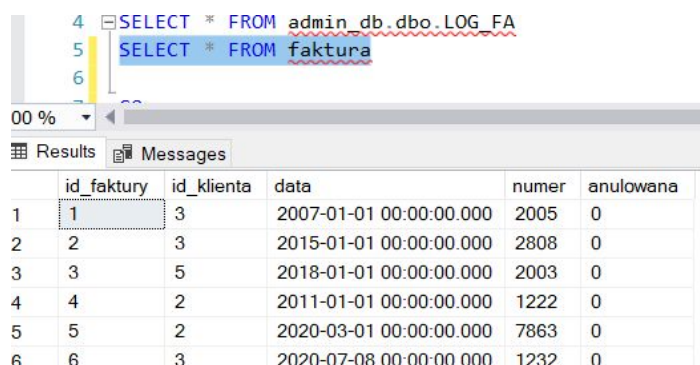


```
UPDATE [dbo].[faktura] SET id_klienta=5 WHERE id_faktury=1
GO
```

Msg 50000, Level 11, State 1, Procedure block_invoice_changes, Line 13 [Batch Start Line 8]
 BŁĄD! NIE MOŻESZ EDYTOWAĆ ID_KLIENTA ORAZ NUMERU FAKTURY
 Msg 3609, Level 16, State 1, Line 9
 The transaction ended in the trigger. The batch has been aborted.

Completion time: 2020-11-30T16:37:09.3735366+01:00

Rys.17 Próba edycji id_klienta -- ponownie zakończona niepowodzeniem i informacją o błędzie



```

4 SELECT * FROM admin_db.dbo.LOG_FA
5 SELECT * FROM faktura
6

```

	id_faktury	id_klienta	data	numer	anulowana
1	1	3	2007-01-01 00:00:00.000	2005	0
2	2	3	2015-01-01 00:00:00.000	2808	0
3	3	5	2018-01-01 00:00:00.000	2003	0
4	4	2	2011-01-01 00:00:00.000	1222	0
5	5	2	2020-03-01 00:00:00.000	7863	0
6	6	3	2020-07-08 00:00:00.000	1232	0

Rys.18 Widzimy, że tabela została niezmienniona -- trigger zablokował niepożądane zmiany

Dzięki wykorzystaniu powyższych triggerów umożliwiało zapisywanie wystawianych faktur w logu oraz blokowanie niepożądanych zmian na polach, które nie powinny być zmieniane, jest to oczekiwany efekt, który zakładaliśmy na początku.

Tworzenie i odtwarzanie backupu bazy

Backup tworzę z wykorzystaniem poniższej procedury. Zapisuje go do pliku oznaczonego nazwą bazy oraz stemplem czasowym, który trafia do folderu `C:\temp`

```

276 ALTER PROCEDURE dbo.Z3_database_backup
277 AS
278 BEGIN
279     DECLARE @path nvarchar(1000), @fname nvarchar(256), @sql nvarchar(200), @db nvarchar(100)
280     SET @db = 'pwx_db_Z3'
281     SET @path = N'C:\temp\'
282     SET @db = LTRIM(RTRIM(@db))
283     SET @fname = REPLACE(REPLACE(CONVERT(nchar(19), GETDATE(), 126), N':', N'_'), '-', '_')
284     SET @fname = @path + RTRIM(@db) + @fname + N'.bak'
285     SET @sql = 'backup database ' + @db + ' to DISK= ''' + @fname + ''''
286     EXEC sp_sqlserv @sql
287 END
288 GO
289
290 EXEC dbo.Z3_database_backup
291
292

```

00 %

Messages

Processed 384 pages for database 'pwx_db_Z3', file 'pwx_db_Z3' on file 1.
 Processed 1 pages for database 'pwx_db_Z3', file 'pwx_db_Z3_log' on file 1.
 BACKUP DATABASE successfully processed 385 pages in 0.130 seconds (23.088 MB/sec).
 Completion time: 2020-11-30T09:09:13.1338669+01:00

Rys.19 Tworzenie backupu bazy `pwx_db_Z3`

Nazwa	Data modyfikacji	typ	rozmiar
pwx_db_Z32020_11_30T09_09_12.bak	30.11.2020 09:09	Plik BAK	3 161 KB
product.tpl	27.11.2020 13:27	Plik TPL	14 KB

Rys.20 Backup widoczny w określonym w procedurze folderze `C:\temp`


```

292 |-----Odtworzenie bazy -----
293 |/*****
294 |
295 |RESTORE DATABASE db_restore FROM DISK = N'C:\temp\pwx_db_Z32020_11_30T09_09_12.bak' WITH REPLACE
296 |
297 |-----Porównywanie -----
298 |/*****
299 |
300 |USE admin_db
301 |GO

```

100 %

Messages

Processed 384 pages for database 'db_restore', file 'pwx_db_Z3' on file 1.
 Processed 1 pages for database 'db_restore', file 'pwx_db_Z3_log' on file 1.
 RESTORE DATABASE successfully processed 385 pages in 0.048 seconds (62.530 MB/sec).
 Completion time: 2020-11-30T09:13:31.0210192+01:00

Rys.21 Odtworzenie bazy z utworzonego wcześniej backupu do bazy `db_restore`

Utworzenie oraz testy procedur do porównywania bazy z logiem

Procedury porównujące bazę z logiem sparametryzowałem, tak, aby była możliwość porównania loga z dowolną wybraną bazą oraz wybranej bazy z logiem (porównywanie obustronne).

```

315 |GO
316 |
317 |ALTER PROCEDURE dbo.compare_log_with_db (@db nvarchar(100))
318 |AS
319 |BEGIN
320 |    CREATE TABLE #TT ([numer_faktury] int, [nip_klienta] int, [data] datetime, [anulowana] bit )
321 |    DECLARE @sql nvarchar(1000)
322 |    SET @sql = 'SELECT f.numer AS numer_faktury, k.NIP AS nip_klienta, f.[data], f.anulowana FROM ' + @db + '.dbo.faktura f
323 |    INNER JOIN ' + @db + '.dbo.klient k ON k.id_klienta = f.id_klienta'
324 |    INSERT INTO #TT exec (@sql)
325 |
326 |    SELECT * FROM admin_db.dbo.LOG_FA
327 |    EXCEPT
328 |    SELECT * FROM #TT
329 |    DROP TABLE #TT
330 |END
331 |GO

```

%

Messages

Commands completed successfully.
 Completion time: 2020-11-30T09:16:23.4262099+01:00

Rys.22 Tworzenie procedury do porównywania loga z bazą

```

341 |BEGIN
342 |    DECLARE @stmt nvarchar(100)
343 |    SET @stmt = 'CREATE PROCEDURE dbo.compare_db_with_log AS '
344 |    EXEC sp_sqlexec @stmt
345 |END
346 |GO
347 |
348 |ALTER PROCEDURE dbo.compare_db_with_log (@db nvarchar(100))
349 |AS
350 |BEGIN
351 |    CREATE TABLE #TT ([numer_faktury] int, [nip_klienta] int, [data] datetime, [anulowana] bit )
352 |    DECLARE @sql nvarchar(1000)
353 |    SET @sql = 'SELECT f.numer AS numer_faktury, k.NIP AS nip_klienta, f.[data], f.anulowana FROM ' + @db + '.dbo.faktura f
354 |    INNER JOIN ' + @db + '.dbo.klient k ON k.id_klienta = f.id_klienta'
355 |    INSERT INTO #TT exec (@sql)
356 |
357 |    SELECT * FROM #TT
358 |    EXCEPT
359 |    SELECT * FROM admin_db.dbo.LOG_FA
360 |    DROP TABLE #TT
361 |END
362 |GO
363 |

```

30 %

Messages

Commands completed successfully.
 Completion time: 2020-11-30T09:18:36.5340521+01:00

Rys.23 Tworzenie procedury do porównywania bazy z logiem

Na potrzeby testów procedur porównujących wprowadziłem parę zmian pomiędzy logiem i odtworzoną bazą z fakturami.

```

364  /*****
365  -- Testy porównywania
366  EXEC dbo.compare_db_with_log @db='db_restore'
367  EXEC dbo.compare_log_with_db @db='db_restore'
368  *****/
369  /*****
370  -- Zapytania pomocnicze do testów
371  SELECT * FROM admin_db.dbo.LOG_FA
372  SELECT * FROM pwx_db_Z3.dbo.faktura

```

100 %

Results Messages

	numer_faktury	nip_klienta	data	anulowana
1	1003	7876537	2015-01-01 00:00:00.000	0
2	1027	7876537	2007-01-01 00:00:00.000	0
3	7622	7876537	2018-01-01 00:00:00.000	0

Rys.24 Test procedury porównującej log z bazą

```

371  EXEC dbo.compare_db_with_log @db='db_restore'
372  EXEC dbo.compare_log_with_db @db='db_restore'
373  *****/
374  /*****
375  -- Zapytania pomocnicze do testów
376  SELECT * FROM admin_db.dbo.LOG_FA
377  SELECT * FROM pwx_db_Z3.dbo.faktura

```

100 %

Results Messages

	numer_faktury	nip_klienta	data	anulowana
1	1003	7876537	2015-01-01 00:00:00.000	0
2	1027	7876537	2007-01-01 00:00:00.000	0
3	1222	1247895	2011-01-01 00:00:00.000	0
4	1232	214657	2020-07-08 00:00:00.000	0
5	1234	214657	2019-11-21 00:00:00.000	0
6	2318	9874698	2021-02-25 00:00:00.000	0
7	5432	214657	2020-04-01 00:00:00.000	0
8	7622	7876537	2018-01-01 00:00:00.000	0
9	7863	1247895	2020-03-01 00:00:00.000	0
10	9877	265444	2020-01-14 00:00:00.000	0

Rys.25 Kolejny test -- wykonany po usunięciu wszystkich danych z tabel w bazie *db_restore* (efekt porównania wskazuje nam na elementy, których brakuje)

Jak widać na powyższych zrzutach ekranu, otrzymałem pożądany efekt -- procedura wskazała na elementy, które różnią się pomiędzy logiem a bazą lub na odwrót -- na tej podstawie jesteśmy w stanie wnioskować jakie zmiany zachodziły w bazie oraz jakie są rozbieżności pomiędzy bazami.

Cały kod wykorzystany do realizacji zadania

```
-- Maciej Czarkowski 292810
-- Administrowanie bazami danych Z3
/***** Tworzę nową bazę danych pwx_db_Z3 oraz tabele faktura, klient, pozycja na
potrzeby zadania *****/
IF NOT EXISTS (SELECT d.name
                FROM sys.databases d
                WHERE (d.database_id > 4)
                AND   (d.[name] = N'pwx_db_Z3')
)
BEGIN
    CREATE DATABASE pwx_db_Z3
END
GO
/***** Usuwanie tabel *****/
USE pwx_db_Z3
GO

DROP TABLE pozycje
DROP TABLE faktura
DROP TABLE klient
/***** Tworzę nowe tabele *****/

-- tabela klient
USE pwx_db_Z3
GO

IF NOT EXISTS
(
    SELECT 1
        from sysobjects o (NOLOCK)
        WHERE (o.[name] = N'klient')
        AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
    CREATE TABLE dbo.klient
    (
        [id_klienta]                int                NOT NULL IDENTITY constraint
        pk_klienta primary key
        , [NIP]                     nvarchar(20)        NOT NULL
        , [nazwa]                   nvarchar(100)       NOT NULL
        , [adres]                   nvarchar(100)       NOT NULL
    )
END
GO

-- tabela faktura
USE pwx_db_Z3
GO

IF NOT EXISTS
(
    SELECT 1
        from sysobjects o (NOLOCK)
        WHERE (o.[name] = N'faktura')
        AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
```

```

CREATE TABLE dbo.faktura
(
    [id_faktury]    int          NOT NULL IDENTITY constraint pk_faktury primary key
    , [id_klienta]  int          NOT NULL constraint pk_fk_klienta foreign key
references klient(id_klienta)
    , [data]        datetime     NOT NULL
    , [numer]       int          NOT NULL
    , [anulowana]   bit          NOT NULL
)
END
GO

-- tabela pozycje

USE pwx_db_Z3
GO

IF NOT EXISTS
(
    SELECT 1
        from sysobjects o (NOLOCK)
        WHERE (o.[name] = N'pozycje')
        AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)
BEGIN
    CREATE TABLE dbo.pozycje
    (
        [id_faktury] int          NOT NULL constraint pk_fk_faktury foreign key
references faktura(id_faktury)
        , [opis]     nvarchar(100) NOT NULL
        , [cena]     float          NOT NULL
    )
END
GO

-----
/***** Baza administracyjna *****/
-- Tworzę bazę administracyjną
IF NOT EXISTS (SELECT d.name
                FROM sys.databases d
                WHERE (d.database_id > 4)
                AND   (d.[name] = N'admin_db'))
)
BEGIN
    CREATE DATABASE admin_db
END
GO

USE admin_db
GO
/** Usuwanie tabeli **/
DROP TABLE LOG_FA

-- Tworzę tabelę LOG_FA
IF NOT EXISTS
(
    SELECT 1
        from sysobjects o (NOLOCK)
        WHERE (o.[name] = N'LOG_FA')
        AND   (OBJECTPROPERTY(o.[ID], N'IsUserTable')=1)
)

```

```

BEGIN
    CREATE TABLE dbo.[LOG_FA]
    (
        [numer_faktury]      int                not null
    ,   [nip_klienta]        int                not null
    ,   [data]               datetime           not null
    ,   [anulowana]         bit                not null
    )
END
GO

-----
/***** Triggery *****/
USE pwx_db_Z3
GO
-- Trigger na UPDATE klienta
DROP TRIGGER [dbo].[block_client_changes]
GO

CREATE TRIGGER [dbo].[block_client_changes]
ON [dbo].[klient]
FOR UPDATE
AS
BEGIN
    IF EXISTS
    (
        SELECT * FROM INSERTED ins
        JOIN DELETED del on ins.id_klienta = del.id_klienta
        WHERE ins.NIP != del.NIP
    )
    BEGIN
        ROLLBACK TRANSACTION
        RAISERROR('BŁĄD! NIE MOŻESZ EDYTOWAĆ NIPU', 11 , 1);
    END
END
GO

--test triggera
--UPDATE klient SET NIP=129777 WHERE id_klienta=1
-- Trigger na UPDATE faktury
DROP TRIGGER [dbo].[block_invoice_changes]

CREATE TRIGGER [dbo].[block_invoice_changes]
ON [dbo].[faktura]
FOR UPDATE
AS
BEGIN
    IF EXISTS
    (
        SELECT * FROM inserted ins
        JOIN deleted del on ins.id_faktury = del.id_faktury
        WHERE ( ins.numer != del.numer OR ins.id_klienta != del.id_klienta)
    )
    BEGIN
        ROLLBACK TRANSACTION
        RAISERROR('BŁĄD! NIE MOŻESZ EDYTOWAĆ ID_KLIENTA ORAZ NUMERU FAKTURY', 11 , 1);
    END
END
GO

```



```
-- Trigger na INSERT nowej faktury/faktur (może być kilka na raz w jednym zapytaniu)
DROP TRIGGER [dbo].[insert_faktura_trigger]
GO

CREATE TRIGGER [dbo].[insert_faktura_trigger]
ON faktura
AFTER INSERT
AS
BEGIN
    INSERT INTO admin_db.dbo.LOG_FA
    SELECT ins.numer as numer_faktury, k.NIP as nip_klienta, ins.[data], ins.anulowana
    FROM INSERTED ins JOIN klient k on ins.id_klienta = k.id_klienta
END
GO

--test triggera (sprawdzam czy są nowe rekordy w admin_db w LOG_FA po wstawieniu faktur)
USE admin_db
GO

SELECT * FROM dbo.LOG_FA
-----
/***** Backup bazy *****/

USE admin_db
GO

/* backup pojedynczej bazy*/
IF NOT EXISTS
(
    SELECT 1
    from sysobjects o (NOLOCK)
    WHERE    (o.[name] = 'Z3_database_backup')
    AND      (OBJECTPROPERTY(o.[ID], 'IsProcedure')=1)
)
BEGIN
    DECLARE @stmt nvarchar(100)
    SET @stmt = 'CREATE PROCEDURE dbo.Z3_database_backup AS '
    EXEC sp_sqlEXEC @stmt
END
GO

ALTER PROCEDURE dbo.Z3_database_backup
AS
BEGIN
    DECLARE @path nvarchar(1000), @fname nvarchar(256), @sql nvarchar(200), @db
    nvarchar(100)
    SET @db = 'pwx_db_Z3'
    SET @path = N'C:\temp\'
    SET @db = LTRIM(RTRIM(@db))
    SET @fname = REPLACE(REPLACE(CONVERT(nchar(19), GETDATE()), 126), N':',
    N'_' ), '-' , '_' )
    SET @fname = @path + RTRIM(@db) + @fname + N'.bak'
    SET @sql = 'backup database ' + @db + ' to DISK= ''' + @fname + ''''
    EXEC sp_sqlEXEC @sql
END
GO
```

```

EXEC dbo.Z3_database_backup
/***** Odtworzenie bazy z pliku *****/

RESTORE DATABASE db_restore FROM DISK = N'C:\temp\pwx_db_Z32020_11_30T09_09_12.bak' WITH
REPLACE

-----

/***** Porównywanie baz *****/

USE admin_db
GO
/*****
-- Porównywanie loga z tabelą faktura w bazie podanej jako argument
IF NOT EXISTS
(
    SELECT 1
        from sysobjects o (NOLOCK)
        WHERE    (o.[name] = 'compare_log_with_db')
        AND      (OBJECTPROPERTY(o.[ID], 'IsProcedure')=1)
)
BEGIN
    DECLARE @stmt nvarchar(100)
    SET @stmt = 'CREATE PROCEDURE dbo.compare_log_with_db AS '
    EXEC sp_sqlEXEC @stmt
END
GO

ALTER PROCEDURE dbo.compare_log_with_db (@db nvarchar(100))
AS
BEGIN
    CREATE TABLE #TT ([numer_faktury] int, [nip_klienta] int, [data] datetime,
[anulowana] bit )
    DECLARE @sql nvarchar(1000)
    SET @sql = 'SELECT f.numer AS numer_faktury, k.NIP AS nip_klienta, f.[data],
f.anulowana FROM ' + @db + '.dbo.faktura f
    INNER JOIN ' + @db + '.dbo.klient k ON k.id_klienta = f.id_klienta'
    INSERT INTO #TT exec (@sql)

    SELECT * FROM admin_db.dbo.LOG_FA
    EXCEPT
    SELECT * FROM #TT
    DROP TABLE #TT
END
GO
/*****
-- Porównywanie loga z tabelą faktura w bazie podanej jako argument
IF NOT EXISTS
(
    SELECT 1
        from sysobjects o (NOLOCK)
        WHERE    (o.[name] = 'compare_db_with_log')
        AND      (OBJECTPROPERTY(o.[ID], 'IsProcedure')=1)
)
BEGIN
    DECLARE @stmt nvarchar(100)
    SET @stmt = 'CREATE PROCEDURE dbo.compare_db_with_log AS '
    EXEC sp_sqlEXEC @stmt
END

```

```

GO
ALTER PROCEDURE dbo.compare_db_with_log (@db nvarchar(100))
AS
BEGIN
    CREATE TABLE #TT ([numer_faktury] int, [nip_klienta] int, [data] datetime,
[anulowana] bit )
    DECLARE @sql nvarchar(1000)
    SET @sql = 'SELECT f.numer AS numer_faktury, k.NIP AS nip_klienta, f.[data],
f.anulowana FROM ' + @db + '.dbo.faktura f
    INNER JOIN ' + @db + '.dbo.klient k ON k.id_klienta = f.id_klienta'
    INSERT INTO #TT exec (@sql)

    SELECT * FROM #TT
    EXCEPT
    SELECT * FROM admin_db.dbo.LOG_FA
    DROP TABLE #TT
END
GO

-----
/***** Wstawianie danych (przydatne po utworzeniu triggerów do ich testów) *****/
USE pwx_db_Z3
GO

DECLARE @id_mc int , @id_td int, @id_ms int, @id_mm int, @id_kd int
INSERT INTO klient VALUES ('7876537', 'Maciej Kasztanabe', 'Tbilisi')
SET @id_mc = SCOPE_IDENTITY()
INSERT INTO klient VALUES ('1247895', 'Tomasz Duszanski', 'Kutaisi')
SET @id_td = SCOPE_IDENTITY()
INSERT INTO klient VALUES ('7214657', 'Monika Sarzoghlu', 'Moskwa')
SET @id_ms = SCOPE_IDENTITY()
INSERT INTO klient VALUES ('2265444', 'Maria Maren', 'Jakuck')
SET @id_mm = SCOPE_IDENTITY()
INSERT INTO klient VALUES ('9874698', 'Karen Daniel', 'Houston')
SET @id_kd = SCOPE_IDENTITY()

INSERT INTO faktura VALUES
(@id_mc, convert(datetime,'20070101',112), 1027, 0),
(@id_mc, convert(datetime,'20150101',112), 1003, 0),
(@id_mc, convert(datetime,'20180101',112), 7622, 0),
(@id_td, convert(datetime,'20110101',112), 1222, 0),
(@id_td, convert(datetime,'20200301',112), 7863, 0),
(@id_ms, convert(datetime,'20200708',112), 1232, 0),
(@id_ms, convert(datetime,'20191121',112), 1234, 0),
(@id_ms, convert(datetime,'20200401',112), 5432, 0),
(@id_mm, convert(datetime,'20200114',112), 9877, 0),
(@id_kd, convert(datetime,'20210225',112), 2318, 0)
/*****/

-- Testy porównywania
EXEC dbo.compare_db_with_log @db='pwx_db_Z3'
EXEC dbo.compare_log_with_db @db='pwx_db_Z3'

-----
/*****/

-- Zapytania pomocnicze do testów
SELECT * FROM admin_db.dbo.LOG_FA
SELECT * FROM pwx_db_Z3.dbo.faktura

```