SPRAWOZDANIE 3
Aleksandra Kowalczyk
307414

Zagadnienie:

Nasze działanie ma na celu ochronę danych krytycznych dotyczących wystawionych faktur. Za każdym razem przy dodaniu nowej faktury do tabeli w naszej bazie administracyjnej dodawane są krytyczne dane dotyczące tej faktury. Dzięki temu, przy awarii bazy faktur będziemy w stanie za pomocą tej tabeli z logami w bazie administracyjnej odtworzyć wszystkie faktury (ponieważ backup nie jest aktualizowany z każdym rekordem, więc dane tam mogą być nieaktualne).

Kod:

```
-- Trigger insert dodający nowy rekord do tabeli LOG_FA za każdym razem gdy dodajemy
nową fakturę

USE DB_faktury
GO

IF (OBJECT_ID('[dbo].insert_fk') IS NOT NULL) DROP TRIGGER [dbo].insert_fk
GO
CREATE TRIGGER [dbo].insert_fk
ON [dbo].Faktura
FOR INSERT AS
BEGIN
    INSERT INTO DB_STAT.dbo.LOG_FA(numer_faktury, nip_klienta, data_wystawienia,
anulowana)
    SELECT i.numer, k.NIP, i.data_wystawienia, i.anulowana
    FROM inserted i JOIN Klient k ON i.id_klienta = k.id_klienta
END
GO
```

```
-- Trigger update, który aktualizuje wartość ‘anulowana’ gdy ulegnie zmianie (nie
wolno nam usunąć faktury, może zostać ona jedynie anulowana np. przez klienta ale
musi pozostać w bazie)


IF (OBJECT_ID('[dbo].update_fk') IS NOT NULL) DROP TRIGGER [dbo].update_fk
GO
CREATE TRIGGER [dbo].update_fk
ON [dbo].Faktura
FOR UPDATE AS
BEGIN
    IF EXISTS (SELECT 1 FROM inserted i join deleted d ON (i.id_faktury =
d.id_faktury)
    WHERE NOT (i.anulowana = d.anulowana)
    )
    INSERT INTO DB_STAT.dbo.LOG_FA(anulowana) SELECT i.anulowana FROM inserted i join
deleted d ON (i.id_faktury = d.id_faktury)
WHERE NOT(i.anulowana = d.anulowana)
END
GO
```

```
-- Procedura backup bazy (przekazujemy jako argument bazę na której backup chcemy
dokonać)

USE DB_STAT
GO
CREATE PROCEDURE bk_db
@db nvarchar(100) ,@path nvarchar(200)
AS
BEGIN
declare @fname nvarchar(1000)
SET @path = LTRIM(RTRIM(@path))
IF @path NOT LIKE N'%\'
SET @path = @path + N'\'

SET @fname = REPLACE(REPLACE(CONVERT(nchar(19), GETDATE(), 126), N':', N'_'),'-','_')
SET @fname = @path + RTRIM(@db)  + @fname + N'.bak'

DECLARE @sql nvarchar(1000)

SET @sql = 'backup database ' + @db + ' to DISK= N''' + @fname + ''''
EXEC sp_sqlexec @sql
END
GO
```

```
--Procedura sprawdzająca jakich faktur brakuje w bazie w porównaniu z logiem:


USE DB_STAT
IF NOT EXISTS
(SELECT 1
from sysobjects o (NOLOCK)
WHERE(o.[name] ='BRAKUJACE_FK_IN_DB')
AND (OBJECTPROPERTY(o.[ID],'IsProcedure')=1)
)
BEGIN
    DECLARE @stmt nvarchar(100)
    SET @stmt ='CREATE PROCEDURE dbo.BRAKUJACE_FK_IN_DB AS '
    EXEC sp_sqlexec @stmt
END
GO
USE DB_STAT
GO
ALTER PROCEDURE dbo.BRAKUJACE_FK_IN_DB
( @db nvarchar(100)
)
AS
    DECLARE @stmt nvarchar(150)
    SET @stmt ='SELECT l.* FROM DB_STAT.dbo.LOG_FA l LEFT JOIN DB_faktury.dbo.Faktura
f ON
    f.numer=l.numer_faktury WHERE f.id_klienta IS NULL'
    EXEC('USE '+@db +';'+@STMT)
GO
```

Postępowanie:

Postępowanie pokażemy na przykładzie.

0. Przygotowania:

a) Tworzymy bazę DB_faktury

```
IF NOT EXISTS (
   SELECT name
   FROM sys.databases
   WHERE name = N'DB_faktury'
)
CREATE DATABASE [DB_faktury]
GO

ALTER DATABASE [DB_faktury] SET QUERY_STORE=ON
GO
```

b) Tworzymy tabelę Klient w naszej bazie DB_faktury

```
USE DB_faktury

IF OBJECT_ID('dbo.Klient', 'U') IS NOT NULL
DROP TABLE dbo.Klient
GO
CREATE TABLE dbo.Klient
(
   id_klienta        INT    NOT NULL   PRIMARY KEY,
   NIP      [NVARCHAR](20)  NOT NULL,
   nazwa  [NVARCHAR](100)  NOT NULL,
   adres     [NVARCHAR](100)  NOT NULL
);
GO
```

c) Tworzymy tabelę Faktura w naszej bazie DB_faktury

```
IF OBJECT_ID('dbo.Faktura', 'U') IS NOT NULL
DROP TABLE dbo.Faktura
GO

CREATE TABLE dbo.Faktura
(
   id_faktury        INT    NOT NULL    PRIMARY KEY,
   id_klienta      INT  NOT NULL,
   data_wystawienia  DATETIME  NOT NULL,
   numer    INT  NOT NULL,
   anulowana BIT NOT NULL
);
GO
```

d) Tworzymy tabelę Pozycje w naszej bazie DB_faktury

```
IF OBJECT_ID('dbo.Pozycje', 'U') IS NOT NULL
DROP TABLE dbo.Pozycje
GO
CREATE TABLE dbo.Pozycje
(
   id_faktury        INT    NOT NULL   PRIMARY KEY,
   opis      [NVARCHAR](20)  NOT NULL,
   cena      [NVARCHAR](20)  NOT NULL,
);
GO
```

e) Teraz używamy naszej bazy administracyjnej DB_STAT i w niej tworzymy naszą tabelę z logami LOG_FA:

```
USE DB_STAT

IF OBJECT_ID('dbo.LOG_FA', 'U') IS NOT NULL
DROP TABLE dbo.LOG_FA
GO

CREATE TABLE dbo.LOG_FA
(
   numer_faktury        INT    NOT NULL   PRIMARY KEY,
   nip_klienta      [NVARCHAR](20)  NOT NULL,
   data_wystawienia  DATETIME  NOT NULL,
   anulowana BIT NOT NULL
);
GO
```

1. Wstawiamy 6 faktur i 3 klientów:

```
USE DB_faktury

INSERT INTO dbo.Faktura
  (  [id_faktury], [id_klienta] , [data_wystawienia] , [numer] ,[anulowana])

VALUES
    ( 1,1,GETDATE(), 112, 0),
    ( 2,2,GETDATE(), 113, 0),
    ( 3,1,GETDATE(), 114, 0),
    ( 4,2,GETDATE(), 115, 0),
    ( 5,3,GETDATE(), 116, 1),
    ( 6,1,GETDATE(), 117, 0)
GO

INSERT INTO dbo.Klient
   ([id_klienta], [NIP] ,  [nazwa] ,  [adres])
VALUES
   ( 1, N'5730289333', N'SERWIS_OPON', N'Australia'),
   ( 2, N'8327619344', N'GOSPODARSTWO_ROLNE',N'India'),
   ( 3, N'1294629983', N'SKLEP_ZAOPATRZENIOWY', N'Germany')

GO
```

2. Robimy backup bazy:

Backup bazy DB_faktury uruchamiamy poleceniem (pierwszy argument to nasza baza, drugi do ścieżka gdzie chcemy backup zapisać):

```
exec bk_db  N'DB_faktury', N'C:\backup'
```

3. Dodajemy jeszcze 3 faktury i 1 klienta:

```
USE DB_faktury

INSERT INTO dbo.Faktura
  (  [id_faktury], [id_klienta] , [data_wystawienia] , [numer] ,[anulowana])

VALUES
    ( 7,1,GETDATE(), 118, 0),
    ( 8,4,GETDATE(), 119, 0),
    ( 9,3,GETDATE(), 120, 0)
GO


INSERT INTO dbo.Klient
   ([id_klienta], [NIP] ,  [nazwa] ,  [adres])
VALUES
   ( 4, N'5639820011', N'agroturystyka', N'Poland')

GO
```

Nasz trigger insert uruchomi się tutaj automatycznie i nowe rekordy zostaną dodane do tabeli LOG_FA.

4. Odtwarzamy wykonany przez nas Backup do nowej bazy (BK_XXX)

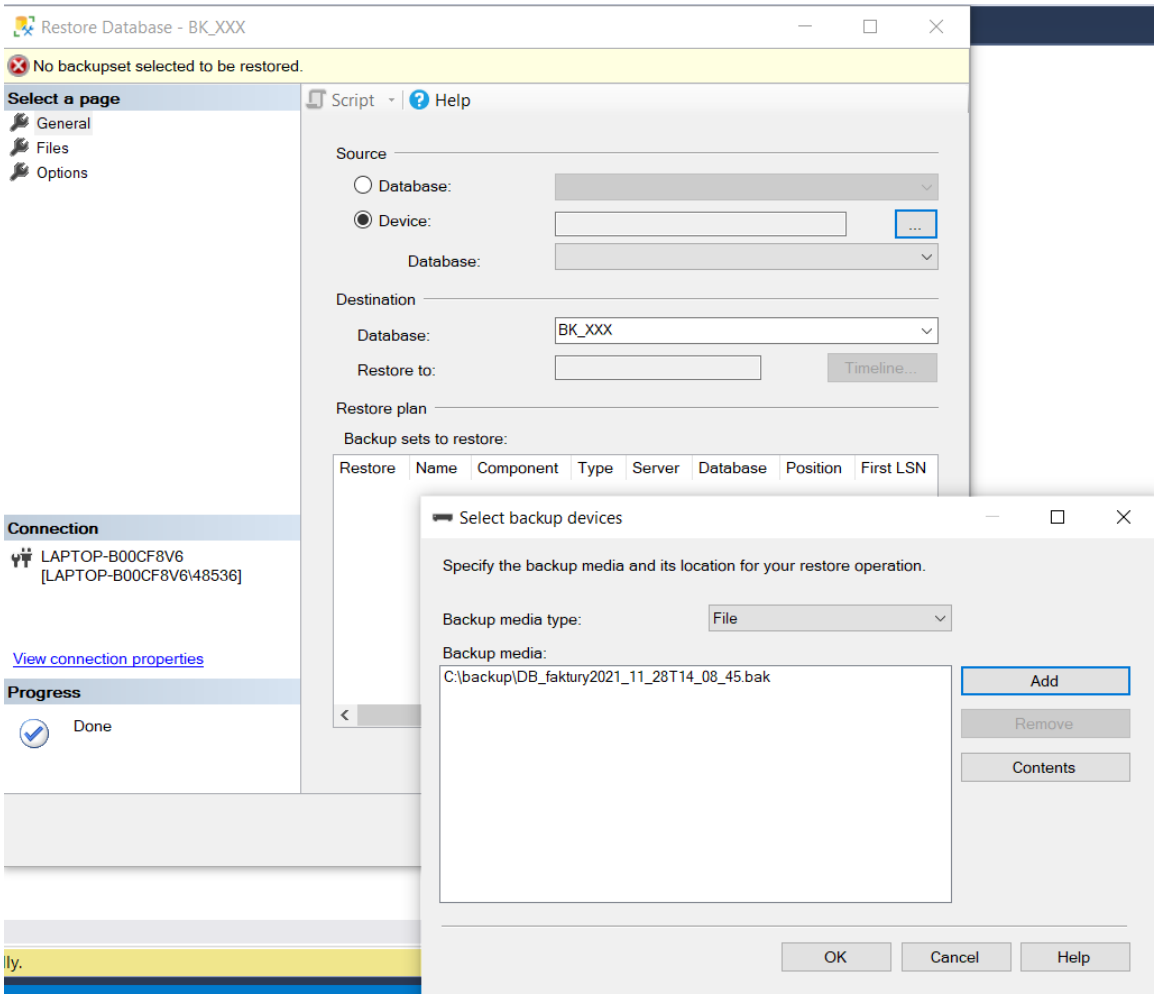
Najpierw tworzymy bazę BK_XXX:

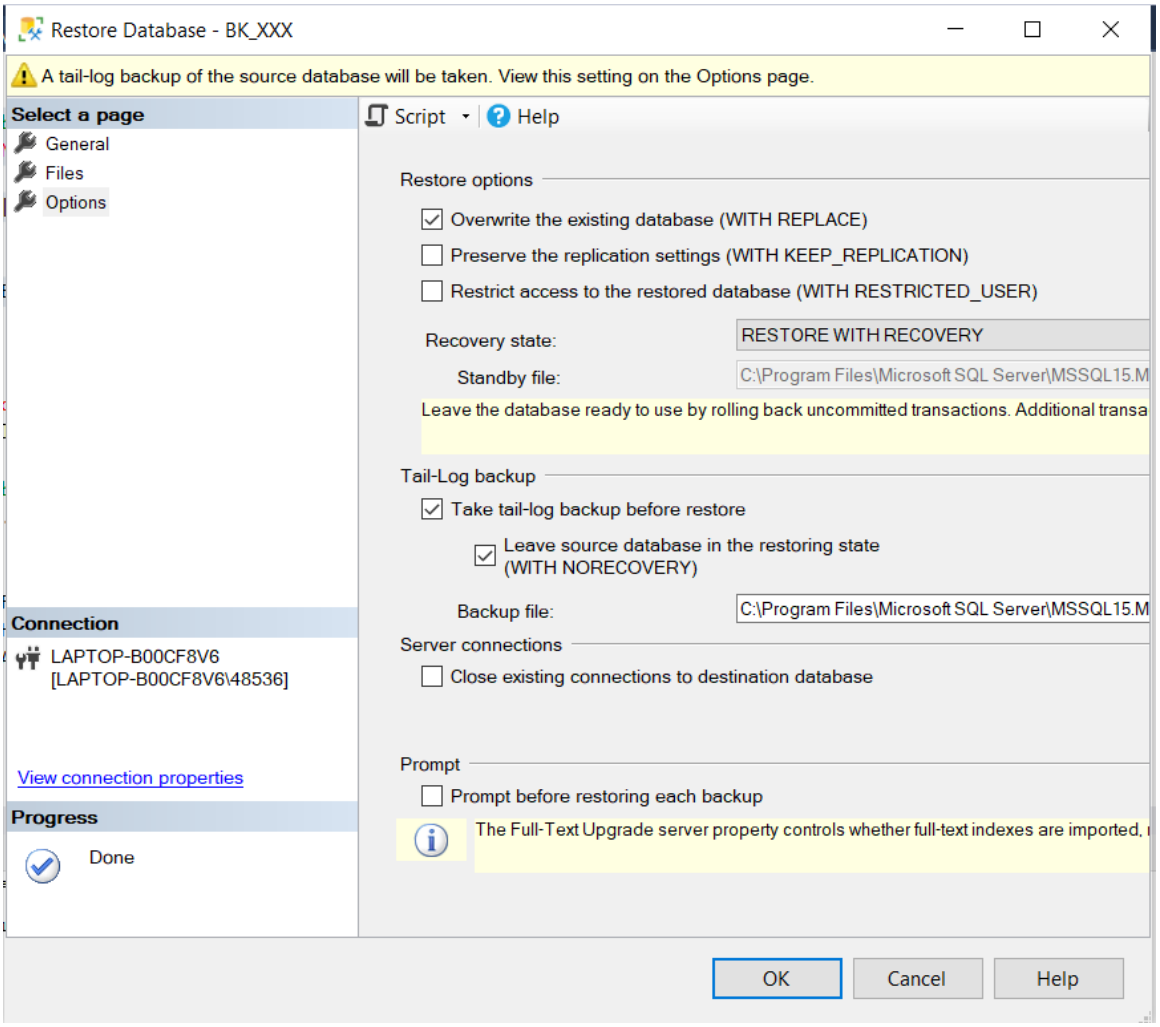
```
IF NOT EXISTS (
   SELECT name
   FROM sys.databases
   WHERE name = N'BK_XXX'
)
CREATE DATABASE [BK_XXX]
GO

ALTER DATABASE [BK_XXX] SET QUERY_STORE=ON
GO
```

Odświeżamy teraz listę naszych baz, by zobaczyć nowoutworzoną bazę BK_XXX.

Klikamy na naszej bazie BK_XXX prawym przyciskiem myszy i wybieramy **Tasks -> Restore -> Database…**

Wybieramy Source: Device, dodajemy plik z naszym backupem (… po prawej stronie). Wchodzimy w zakładkę Options i zaznaczamy overwrite the existing database.





5. Ostatnim krokiem jest wywołanie procedury pokazującej co jest w logu a czego nie ma w odtworzonej bazie BK_XXX

Wywołujemy ją poleceniem:

```
EXEC DB_STAT.dbo.BRAKUJACE_FK_IN_DB @db='BK_XXX'
```

Widzimy, że w BK_XXX brakuje 3 faktur, które dodaliśmy po backupie:

esults | Messages

numer_faktury	nip_klienta	data_wystawienia	anulowana
118	5730289333	2021-11-28 14:20:16.910	0
119	5639820011	2021-11-28 14:20:16.910	0
120	1294629983	2021-11-28 14:20:16.910	0