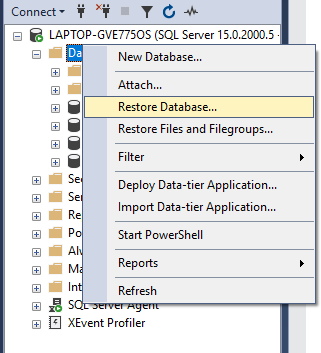
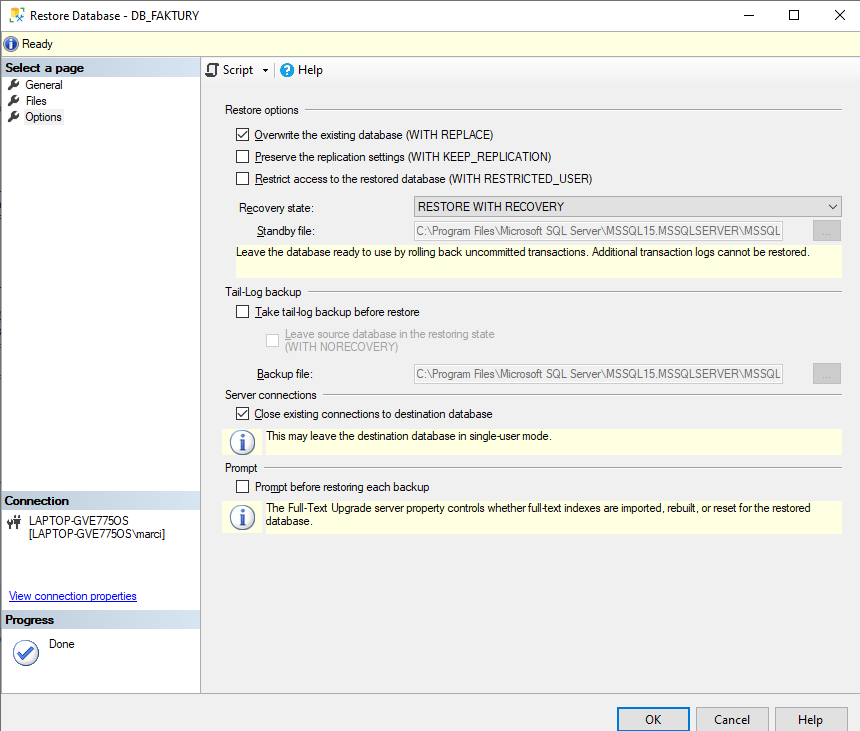
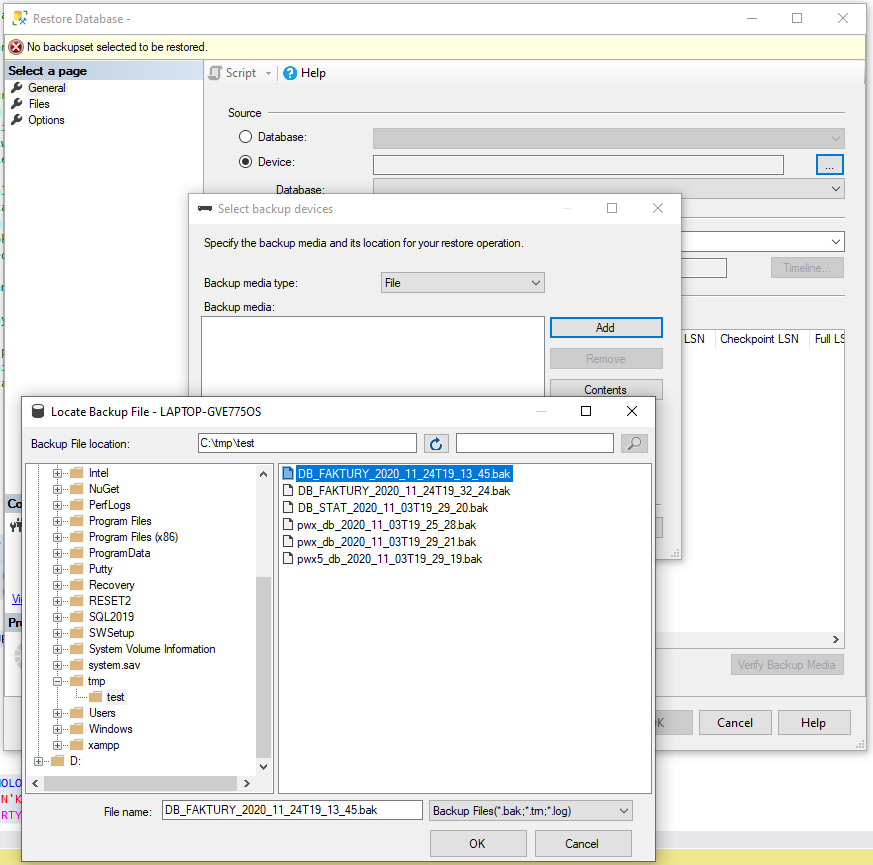
# Proces odtwarzania bazy z pliku

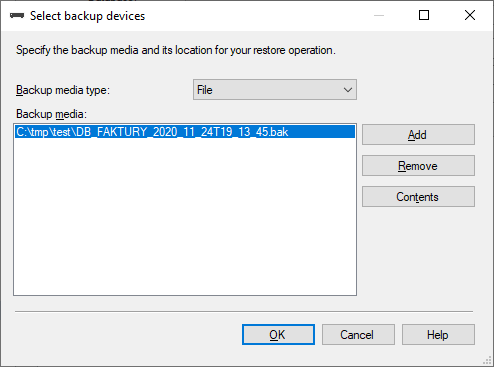
1. Klikamy prawym przyciskiem na databases i wybieramy opcję Restore Database…
2. Chcą odtworzyć kopię bazy w istniejącej już bazie przechodzimy do zakładki Options i zaznaczmy tam nadpisanie istniejącej bazy oraz zamknięcie istniejących połączeń z docelową bazą danych



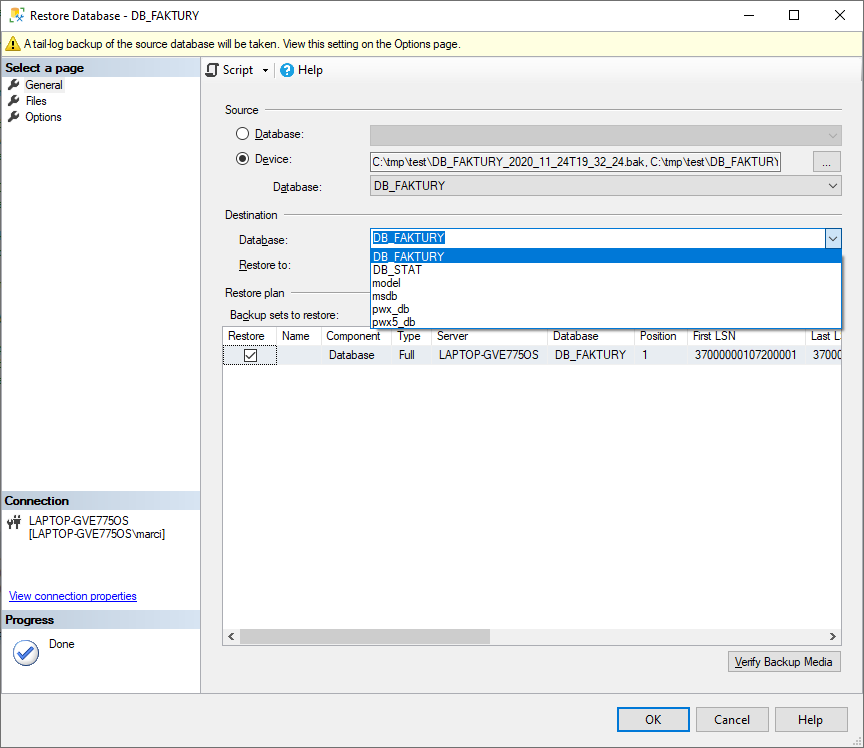
1. W zakładce general w source wybieramy opcję odtworzenia bazy z pliku na urządzeniu (device) następnie klikając przycisk oznaczony jako … otwiera nam się okno w którym klikamy add aby dodać plik z którego chcemy odtworzyć bazę danych. Odnajdujemy ten plik w naszym komputerze klikamy na niego w okienku po prawej stronie a następnie klikamy przycisk ok



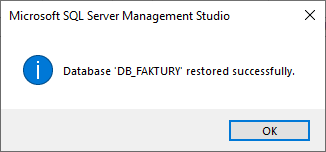
1. Zamykamy również okno Select backup devices zaznaczając wybrany wcześniej plik oraz klikając ok



1. Wybieramy odpowiednią bazę danych

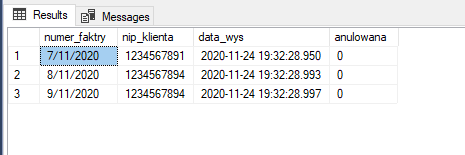


1. Klikamy ok zatwierdzając odtworzenie bazy danych
2. Poniższe okno potwierdza pomyślne odtworzenie bazy danych



# Sprawdzanie czego brakuje w bazie

Uruchamiamy procedurę sprawdzenie, podając jej jako argument nazwę bazy (polecenie EXEC sprawdzenie @db = 'DB\_FAKTURY')

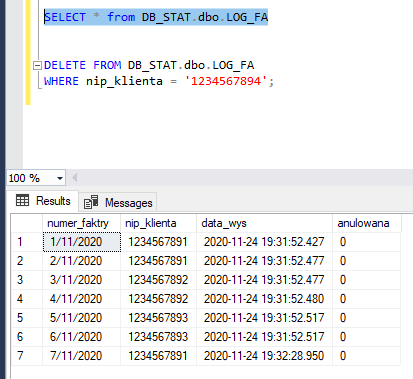


Zostają nam wyświetlone faktury których brakuje, w bazie danych.

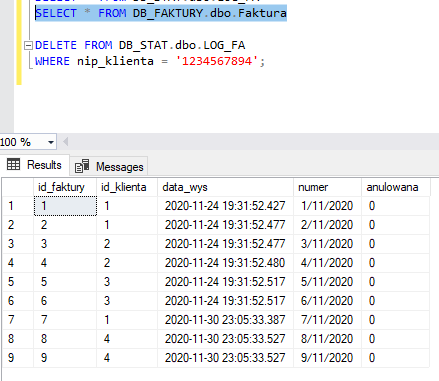
Należy w tym przypadku dodać do bazy danych brakujące faktury.

# Sprawdzanie czego brakuje w logu + dopisanie brakujących rekordów

Zawartość loga:

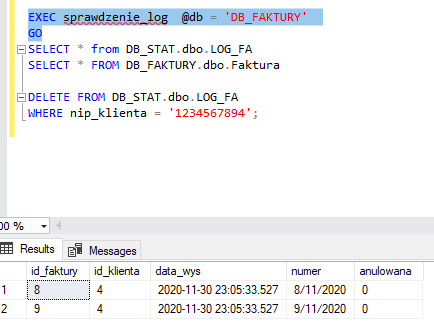


Zawartość tabeli Faktury

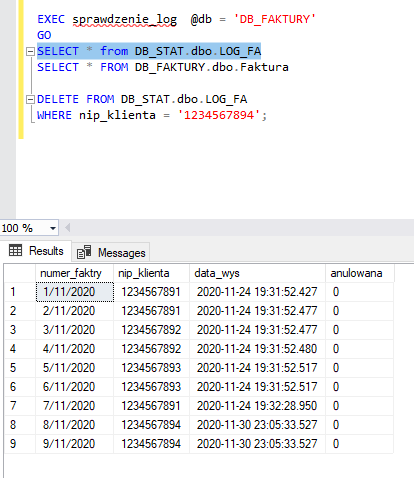


Uruchamiamy procedurę EXEC sprawdzenie\_log @db = 'DB\_FAKTURY', która automatycznie dodaje brakujące rekordy (jeżeli takie występują)

Zostały wyświetlone brakujące rekordy:



Oraz zostały dodane do bazy danych:



# Kod SQL

-- 1

-- tworzenie bazy

IF NOT EXISTS (SELECT d.name

FROM sys.databases d

WHERE (d.database\_id > 4) -- systemowe mają ID poniżej 5

AND (d.[name] = N'DB\_FAKTURY')

)

BEGIN

CREATE DATABASE DB\_FAKTURY

END

GO

USE DB\_FAKTURY

GO

IF NOT EXISTS

( SELECT 1

from sysobjects o (NOLOCK)

WHERE (o.[name] = N'Klienent')

AND (OBJECTPROPERTY(o.[ID],N'IsUserTable')=1)

)

BEGIN

CREATE TABLE dbo.Klient

( id\_klienta int NOT NULL IDENTITY

CONSTRAINT PK\_Klient PRIMARY KEY

, NIP nvarchar(20) NOT NULL

, nazwa nvarchar(100) NOT NULL

, adres nvarchar(100) NOT NULL

)

END

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\_Faktura PRIMARY KEY

, id\_klienta int NOT NULL CONSTRAINT FK\_Faktura FOREIGN KEY

REFERENCES dbo.Klient(id\_klienta)

, data\_wys datetime NOT NULL

, numer nvarchar(100) NOT NULL

, anulowana bit NOT NULL

)

END

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 FK\_Pozycje FOREIGN KEY

REFERENCES dbo.Faktura(id\_faktury)

, opis nvarchar(20) NOT NULL

, cena float NOT NULL

)

END

GO

-- 2

USE DB\_STAT

GO

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\_faktry nvarchar(100) NOT NULL

, nip\_klienta nvarchar(20) NOT NULL

, data\_wys datetime NOT NULL

, anulowana bit NOT NULL

)

END

GO

-- 3,4

USE DB\_FAKTURY

GO

CREATE TRIGGER wystawienie\_faktury ON Faktura FOR INSERT

AS

BEGIN

INSERT INTO DB\_STAT.dbo.LOG\_FA

SELECT i.numer, k.NIP, i.data\_wys, i.anulowana

FROM inserted i

JOIN Klient k on (k.id\_klienta = i.id\_klienta)

END

GO

CREATE TRIGGER update\_faktury ON Faktura AFTER UPDATE

AS

BEGIN

IF EXISTS (

SELECT \* FROM inserted i

JOIN deleted d on ((d.numer != i.numer OR d.id\_klienta != i.id\_klienta) AND d.id\_faktury = i.id\_faktury))

BEGIN

ROLLBACK

END

END

GO

CREATE TRIGGER update\_klienta ON Klient AFTER UPDATE

AS

BEGIN

IF EXISTS (

SELECT \* FROM inserted i

JOIN deleted d on (d.NIP != i.NIP AND d.id\_klienta = i.id\_klienta))

BEGIN

ROLLBACK

END

END

GO

-- 5

USE DB\_FAKTURY

INSERT INTO Klient VALUES ('1234567891', 'Mirabelki', 'Warszawa, Kwaśna 1')

INSERT INTO Klient VALUES ('1234567892', 'Pietrzykowski', 'Warszawa, Twarda 1')

INSERT INTO Klient VALUES ('1234567893', 'Jan Nowak', 'Warszawa, Mała 1')

INSERT INTO Faktura VALUES (1, GETDATE(), '1/11/2020', 0)

INSERT INTO Pozycje VALUES (1, 'mleko', 100)

INSERT INTO Faktura VALUES (1, GETDATE(), '2/11/2020', 0)

INSERT INTO Pozycje VALUES (2, 'płatki', 10)

INSERT INTO Faktura VALUES (2, GETDATE(), '3/11/2020', 0)

INSERT INTO Pozycje VALUES (3, 'książki', 200)

INSERT INTO Faktura VALUES (2, GETDATE(), '4/11/2020', 0)

INSERT INTO Pozycje VALUES (4, 'woda', 400)

INSERT INTO Faktura VALUES (3, GETDATE(), '5/11/2020', 0)

INSERT INTO Pozycje VALUES (5, 'chleb', 110)

INSERT INTO Faktura VALUES (3, GETDATE(), '6/11/2020', 0)

INSERT INTO Pozycje VALUES (6, 'bułki', 220)

SELECT \* FROM Faktura

SELECT \* FROM Klient

-- backup

EXEC DB\_STAT.dbo.bk\_db @db = 'DB\_FAKTURY', @path = 'C:\tmp\test'

GO

INSERT INTO Klient VALUES ('1234567894', 'Wisienka', 'Warszawa, Szpakowa 1')

INSERT INTO Faktura VALUES (1, GETDATE(), '7/11/2020', 0)

INSERT INTO Pozycje VALUES (7, 'inne', 100)

INSERT INTO Faktura VALUES (4, GETDATE(), '8/11/2020', 0)

INSERT INTO Pozycje VALUES (8, 'nasiona', 1000)

INSERT INTO Faktura VALUES (4, GETDATE(), '9/11/2020', 0)

INSERT INTO Pozycje VALUES (9, 'nawóz', 20000)

-- 6 - 'odtworzenie starego backupu'

SELECT \* FROM Faktura

SELECT \* FROM Klient

SELECT \* FROM DB\_STAT.dbo.LOG\_FA l WHERE NOT EXISTS

( SELECT \* FROM Faktura f

WHERE l.numer\_faktry = f.numer )

GO

CREATE PROCEDURE sprawdzenie @db nvarchar(100) AS

BEGIN

SET @db = LTRIM(RTRIM(@db))

DECLARE @sql nvarchar(3000)

SET @sql = N'SELECT \* FROM DB\_STAT.dbo.LOG\_FA l WHERE NOT EXISTS

( SELECT \* FROM '+ @db + '.dbo.Faktura f

WHERE l.numer\_faktry = f.numer )'

EXEC sp\_sqlexec @sql

END

GO

EXEC sprawdzenie @db = 'DB\_FAKTURY'

GO

INSERT INTO DB\_STAT.dbo.LOG\_FA

SELECT f.numer, k.NIP, f.data\_wys, f.anulowana FROM DB\_FAKTURY.dbo.Faktura f

JOIN DB\_FAKTURY.dbo.Klient k on (k.id\_klienta = f.id\_klienta)

WHERE NOT EXISTS

( SELECT \* FROM DB\_STAT.dbo.LOG\_FA l

WHERE l.numer\_faktry = f.numer )

GO

CREATE PROCEDURE sprawdzenie\_log @db nvarchar(100) AS

BEGIN

SET @db = LTRIM(RTRIM(@db))

DECLARE @sql1 nvarchar(3000)

DECLARE @sql2 nvarchar(3000)

SET @sql1 = N'SELECT \* FROM '+ @db + '.dbo.Faktura f WHERE NOT EXISTS

( SELECT \* FROM DB\_STAT.dbo.LOG\_FA l

WHERE l.numer\_faktry = f.numer )'

EXEC sp\_sqlexec @sql1

SET @sql2 = N'INSERT INTO DB\_STAT.dbo.LOG\_FA

SELECT f.numer, k.NIP, f.data\_wys, f.anulowana FROM '+ @db + '.dbo.Faktura f

JOIN ' + @db + '.dbo.Klient k on (k.id\_klienta = f.id\_klienta)

WHERE NOT EXISTS

( SELECT \* FROM DB\_STAT.dbo.LOG\_FA l

WHERE l.numer\_faktry = f.numer )'

EXEC sp\_sqlexec @sql2

END

GO

DELETE FROM DB\_STAT.dbo.LOG\_FA

WHERE nip\_klienta = '1234567894';

EXEC sprawdzenie\_log @db = 'DB\_FAKTURY'

GO

SELECT \* from DB\_STAT.dbo.LOG\_FA

SELECT \* FROM DB\_FAKTURY.dbo.Faktura