



## Sprawozdanie z projektu nr 2

### Zaawansowane systemy baz danych

#### Słowo wstępu:

Przy realizacji zadania zauważałem potrzebę drobnej modyfikacji bazy. Dodany został status psa 'AcceptedToAdoption'. Również lekkiej modyfikacji uległa tabela Shelters.FamiliarizationVisit jedną datę nullową DateOfVisit zastąpiłem dwoma nullowymi StartDate oraz EndDate, również dodałem nullowe pole RejectionReason (wykorzystywane i pokazane dalej przy omówieniu trigger'ów).

#### Funkcje użytkownika (UDF):

Zdecydowałem się przygotować dwie funkcje. Obie są udostępnione do użytku pracownikom schroniska (choć pierwszą można by również udostępnić klientom).

Pierwsza funkcja `Shelter.DaysSinceLastMedicalProcedure`, pozwala dla wybranego psa sprawdzić, ile dni upłyнуło od ostatniego zabiegu. Możliwe jest podanie fragmentu nazwy zabiegu, aby sprawdzić kiedy wykonano konkretny zabieg. W przypadku podania NULL system wyliczy nam ile dni upłynęło od ogólnie ostatniego zabiegu. Myślę, że funkcja jest ważna pod kątem przeprowadzania rutynowych zabiegów (np. regularne odrobaczanie, raz na 3 miesiące) jak i również wyznaczanie odpowiednich odstępów między zabiegami (dla niektórych psów możemy nie chcieć wykonywać zabiegów częściej niż jeden na tydzień).

-- Ile dni upłynęło od ostatniego odrobaczania psa X	
	SELECT Shelter.DaysSinceLastMedicalProcedure('4DE75DBC-9136-4952-83E1-C13CED7D1EC9', 'Flea & Tick Prevention')
1	(No column name)

1 236

**Zrzut ekranu nr 1:** weryfikacja kiedy wykonano odrobaczanie u psa X

```
-- Ile dni upłynęło od ostatniego zabiegu psa X  
SELECT Shelter.DaysSinceLastMedicalProcedure('4DE75DBC-9136-4952-83E1-C13CED7D1EC9', NULL)
```

	(No column name)
1	205

**Zrzut ekranu nr 2:** weryfikacja kiedy wykonano jakikolwiek zabieg u psa X

Druga funkcja *Shelter.DaysSinceLastAdoptionRequest*, pozwala pracownikom na proste i szybkie skanowanie kiedy dany pies otrzymał ostatni wniosek o adopcję. W przypadku, gdy pies nie otrzymał jeszcze żadnego wniosku funkcja zwróci -1. Możemy zatem w prosty sposób sprawdzić, które psy cieszą się mniejszym zainteresowaniem (np. sprawdzić, które psy nie miały żadnego wniosku przez ostatni miesiąc) i dalej analizować przyczyny.

```
-- widzimy, że jeszcze nikt nie zgłaszał się po tego psa  
SELECT Shelter.DaysSinceLastAdoptionRequest('4DE75DBC-9136-4952-83E1-C13CED7D1EC9')
```

	(No column name)
1	-1

**Zrzut ekranu nr 3:** weryfikacja kiedy zgłoszono wniosek o psa X

```
-- ten pies otrzymał wniosek o adopcję x dni temu  
SELECT Shelter.DaysSinceLastAdoptionRequest('4A6871FB-BBA7-45AA-ADF7-63FA689F83FA')
```

	(No column name)
1	15

**Zrzut ekranu nr 4:** weryfikacja kiedy zgłoszono wniosek o psa Y

Skrypt z powyższymi UDF znajduje się w załączonym pliku: *UDF.sql*.

## Procedury:

Zdefiniowałem trzy relatywnie proste procedury do dodawania przez pracownika do systemu: klientów, psy, schroniska. Dalej przedstawię również kilka bardziej zaawansowanych procedur, które poza tym, że działają na wielu tabelach i są opatrzone transakcjami oraz posiadają szczegółową obsługę błędów – te kluczowe dotyczą zgłoszenia chęci adopcji oraz zapisania się na wizytę zapoznawczą z psem.

*Shelter.AddClient* – procedura, którą dodajemy użytkownika i jego dane poufne. Z racji wykonywania trzech insertów procedura oparta jest o transakcję tak, by pilnować spójności danych.

```

EXEC Shelter.AddClient
    @FirstName = 'Jan',
    @LastName = 'Nowak',
    @Email = 'jan.nowak@example.com',
    @PhoneNumber = '123456789',
    @DateOfBirth = '2001-09-21'
GO

SELECT *
FROM Shelter.Users AS u
JOIN Shelter.UserCredentials AS uc ON uc.UserId = u.Id
WHERE u.FirstName='Jan' AND u.LastName = 'Nowak'
GO

```

Results Messages

Id	FirstName	LastName	RoleId	Id	Email	PhoneNumber	DateOfBirth	UserId
DE7D7DF9-719A-4726-939D-4069F640E235	Jan	Nowak	DA3609E9-9CFE-4940-BE7-6BFF6695A3D9	62D0C60C-006C-409E-985A-41B3E0402422	jan.nowak@example.com	123456789	2001-09-21 00:00:00.0000000	DE7D7DF9-71

**Zrzut ekranu nr 5:** weryfikacja dodania nowego klienta, przy użyciu procedury

*Shelter.AddDog* – procedura, którą pracownik dodaje psa do systemu. Od razu rejestrowana jest karta zdrowia psa. Tutaj również zastosowano transakcję, ponieważ inserty wykonujemy do 3 tabel.

```

EXEC Shelter.AddDog
    @Name = 'Max',
    @Breed = 'Bulldog',
    @EstimatedBirthDate = '2020-07-07',
    @ShelterId = '20C49F67-764B-4E80-969F-9EE046B4548A',
    @Weight = 5
GO

SELECT COUNT(DISTINCT d.[Name])
FROM Shelter.Dogs AS d
JOIN Shelter.DogCharacteristics AS dc ON dc.DogId = d.Id
JOIN Shelter.MedicalCards AS mc ON mc.DogId = d.Id
WHERE d.[Name] = 'Max' AND d.Breed='Bulldog'

```

Results Messages

(No column name)
1

**Zrzut ekranu nr 6:** weryfikacja dodania nowego psa, przy użyciu procedury

*Shelter.AddShelter* – prosta procedura, którą użytkownik dodaje schronisko do systemu. W tym przypadku nie zastosowałem transakcji, ponieważ rekord dodajemy tylko do jednej tabeli.

```

EXEC Shelter.AddShelter
    @Name = 'Safe Haven for Hounds',
    @AddressId = '1D7A7639-89F1-4051-8B2B-6B73781B035A'
GO

SELECT COUNT(*)
FROM Shelter.Shelters AS s
WHERE s.[Name] = 'Safe Haven for Hounds'

```

Results Messages

(No column name)
1

**Zrzut ekranu nr 7:** weryfikacja dodania nowego schroniska, przy użyciu procedury

*Shelter.SubmitAdoptionRequest* – bardziej zaawansowana procedura. Pozwala klientowi, bądź pracownikowi (jeśli klient zwróci się z taką prośbą) o złożenie wniosku o adopcję danego psa. Procedura zawiera walidację działania – sprawdzenie, czy pies o podanym Id istnieje, czy pies jest dostępny (status Available). Dalszą weryfikacją jest sprawdzenie, czy mamy użytkownika o UserId w systemie, czy użytkownik jest klientem (rola Client) oraz czy użytkownik posiada założoną kartę adopcyjną. W przypadku nie spełnienia, któregokolwiek warunku procedura podniesie wyjątek informujący dokładnie jaką była przyczyna wycofania działań. Cała procedura jest opatrzona transakcją.

```
-- Scenariusz pozytywny (pies istnieje i jest dostępny, użytkownik istnieje, jest klientem schroniska i ma założoną kartę adopcyjną)
EXEC Shelter.SubmitAdoptionRequest
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',
    @DogId = '3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
GO

--SELECT COUNT(*)
-- FROM Shelter.AdoptionRecords AS ar
-- JOIN Shelter.AdoptionCards AS ac ON ac.Id = ar.AdoptionCardId
-- WHERE ac.UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2' AND ar.DogId='3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'

1 % < <
Results Messages
(No column name)
1
```

### Zrzut ekranu nr 8: weryfikacja pozytywnego scenariusza zgłoszenia wniosku o adopcję

Poniżej zaprezentuję poprawne działanie procedury w różnych wariantach niepoprawnych danych.

```
-- Scenariusz negatywny (pies nie istnieje)
EXEC Shelter.SubmitAdoptionRequest
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',
    @DogId = '00000000-1111-2222-3333-444444444444'
GO

-- Scenariusz negatywny (pies nie jest dostępny - inny status niż Available)
EXEC Shelter.SubmitAdoptionRequest
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',
    @DogId = 'A1379186-B9B8-40B6-8F63-0290BE97F9A6'
GO

-- Scenariusz negatywny (użytkownik nie istnieje)
EXEC Shelter.SubmitAdoptionRequest
    @UserId = '00000000-1111-2222-3333-444444444444',
    @DogId = '3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
GO

-- Scenariusz negatywny (użytkownik nie jest klientem)
EXEC Shelter.SubmitAdoptionRequest
    @UserId = '5459FD6A-EC98-4E04-9CCE-83E04B9AE948',
    @DogId = '3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
GO

% < <
Messages
Msg 50000, Level 16, State 1, Procedure Shelter.SubmitAdoptionRequest, Line 70 [Batch Start Line 218]
Pies o podanym ID nie istnieje.
Msg 50000, Level 16, State 1, Procedure Shelter.SubmitAdoptionRequest, Line 70 [Batch Start Line 223]
Pies nie jest dostępny do adopcji.
Msg 50000, Level 16, State 1, Procedure Shelter.SubmitAdoptionRequest, Line 70 [Batch Start Line 229]
Użytkownik o podanym ID nie istnieje.
Msg 50000, Level 16, State 1, Procedure Shelter.SubmitAdoptionRequest, Line 70 [Batch Start Line 235]
Użytkownik nie jest klientem schroniska.

Completion time: 2024-10-29T19:47:41.9210282+01:00
```

### Zrzut ekranu nr 9: reakcja systemu na błędne warianty

W sprawozdaniu pomijam test dotyczący złożenia wniosku przez użytkownika bez karty adopcyjnej. Dodawanie użytkowników powinno odbywać tylko przez dedykowaną procedurę, która od razu zakłada użytkownikowi kartę adopcyjną. Taka niespójność mogłaby powstać tylko przy manualnym zakładaniu użytkownika i zabezpieczyłem to dla pewności, że dane są spójne (kod procedury w pliku *Procedures.sql*).

*Shelter.UpdateAdoptionRecordStatus* – procedura pozwalająca pracownikowi na zmianę statusu wniosku adopcyjnego. Procedura waliduje istnienie wniosku o podanym id, czy wniosek nie jest już zamknięty (tzn. status odrzucony lub zrealizowany), czy status, na który zmieniamy jest poprawny. Procedura zmienia status wniosku i aktualizuje jego datę.

Test pozytywny tutaj pomijam (procedura jest wykorzystywana przy dalszych testach trigger'ów). Poniżej testy scenariuszów negatywnych:

```
-- przygotowanie wniosku:  
EXEC Shelter.SubmitAdoptionRequest  
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',  
    @DogId = '4DE75DBC-9136-4952-83E1-C13CED7D1EC9'  
  
DECLARE @SubmittedAdoptionRequestId uniqueidentifier  
SELECT @SubmittedAdoptionRequestId = ar.Id  
    FROM Shelter.AdoptionRecords AS ar  
    JOIN Shelter.AdoptionCards AS ac ON ac.Id = ar.AdoptionCardId  
WHERE DogId = '4DE75DBC-9136-4952-83E1-C13CED7D1EC9' AND ac.UserId='470E2057-1FD6-48BA-BA75-2C4C1303A5D2'  
  
-- Scenariusz negatywny (wniosek adopcyjny nie istnieje)  
EXEC Shelter.UpdateAdoptionRecordStatus  
    @AdoptionRecordId = '00000000-1111-2222-3333-444444444444',  
    @NewStatusId = '494d8c27-ad71-43e3-a216-f0be0450f135'  
  
-- Scenariusz negatywny (niepoprawne Id nowego statusu)  
EXEC Shelter.UpdateAdoptionRecordStatus  
    @AdoptionRecordId = @SubmittedAdoptionRequestId,  
    @NewStatusId = '00000000-1111-2222-3333-444444444444'  
  
-- Scenariusz negatywny (wniosek już odrzucony albo sfinalizowany)  
--najpierw odrzucony  
EXEC Shelter.UpdateAdoptionRecordStatus  
    @AdoptionRecordId = @SubmittedAdoptionRequestId,  
    @NewStatusId = '0e5add7b-d4af-4a39-9a49-ee76e22ed69b'  
  
--próba zmiany odrzuconego na inny  
EXEC Shelter.UpdateAdoptionRecordStatus  
    @AdoptionRecordId = @SubmittedAdoptionRequestId,  
    @NewStatusId = '494d8c27-ad71-43e3-a216-f0be0450f135'  
GO  
  
00 %   
Messages  
  
(1 row affected)  
Msg 50000, Level 16, State 1, Procedure Shelter.UpdateAdoptionRecordStatus, Line 47 [Batch Start Line 294]  
Wniosek adopcyjny o podanym Id nie istnieje.  
Msg 50000, Level 16, State 1, Procedure Shelter.UpdateAdoptionRecordStatus, Line 47 [Batch Start Line 294]  
Nieznany status adopcji.  
  
(1 row affected)  
Msg 50000, Level 16, State 1, Procedure Shelter.UpdateAdoptionRecordStatus, Line 47 [Batch Start Line 294]  
Wniosek adopcyjny jest już zamknięty.  
  
Completion time: 2024-10-29T19:55:15.2166021+01:00
```

Zrzut ekranu nr 10: reakcja systemu na błędne warianty

*Sheleter.SubmitFamiliarizationVisitRequest* – nietrywialna procedura, analogiczna do procedury służącej do składania wniosków o adopcję. Tutaj również walidujemy istnienie i status psa oraz użytkownika i jego rolę. Co ważne to przy zgłoszeniu prośby o wizytę na zakres dat oraz pracownik ustawiane są na NULL. To pracownicy dalej mogą się przypisywać do wizyt i wyznaczać termin wizyty – ta logika biznesowa jest świadoma i ma na celu redukcję skomplikowania procesu i pilnowania przed zakleszczaniem wizyt (pracownicy decydują, jak ułożyć harmonogram wizyt, a system ich w tym wspiera wykrywając i blokując kolizje).

Testy działania procedury:

```
-- Scenariusz pozytywny (pies istnieje i jest dostępny, użytkownik istnieje i jest klientem schroniska)
EXEC Shelter.SubmitFamiliarizationVisitRequest
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',
    @DogId = '3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
GO

SELECT COUNT(*)
    FROM Shelter.FamiliarizationVisits AS fv
    WHERE fv.VisitorId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2' AND fv.DogId='3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
```

Results Messages  
(No column name)  
1

**Zrzut ekranu nr 11:** weryfikacja pozytywnego scenariusza zgłoszenia chęci odbycia wizyty zapoznawczej

```
-- Scenariusz negatywny (pies nie istnieje)
EXEC Shelter.SubmitFamiliarizationVisitRequest
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',
    @DogId = '00000000-1111-2222-3333-444444444444'
GO

-- Scenariusz negatywny (pies nie jest dostępny - inny status niż Available)
EXEC Shelter.SubmitFamiliarizationVisitRequest
    @UserId = '470E2057-1FD6-48BA-BA75-2C4C1303A5D2',
    @DogId = 'A13791B6-B9B8-40B6-8F63-0290BE97F9A6'
GO

-- Scenariusz negatywny (użytkownik nie istnieje)
EXEC Shelter.SubmitFamiliarizationVisitRequest
    @UserId = '00000000-1111-2222-3333-444444444444',
    @DogId = '3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
GO

-- Scenariusz negatywny (użytkownik nie jest klientem)
EXEC Shelter.SubmitFamiliarizationVisitRequest
    @UserId = '5459FD6A-EC98-4E04-9CCE-83E04B9AE948',
    @DogId = '3DAEC742-0FBB-4BEF-A7CB-22B18275A50F'
GO
```

Messages

```
Mag 50000, Level 16, State 1, Procedure Shelter.SubmitFamiliarizationVisitRequest, Line 57 [Batch Start Line 398]
Pies o podanym ID nie istnieje.
Mag 50000, Level 16, State 1, Procedure Shelter.SubmitFamiliarizationVisitRequest, Line 57 [Batch Start Line 403]
Pies nie jest dostępny do wizyty.
Mag 50000, Level 16, State 1, Procedure Shelter.SubmitFamiliarizationVisitRequest, Line 57 [Batch Start Line 409]
Użytkownik o podanym ID nie istnieje.
Mag 50000, Level 16, State 1, Procedure Shelter.SubmitFamiliarizationVisitRequest, Line 57 [Batch Start Line 415]
Użytkownik nie jest klientem schroniska.
```

Completion time: 2024-10-29T19:57:42.3357967+01:00

**Zrzut ekranu nr 12:** reakcja systemu na błędne warianty (SubmitFamiliarizationVisitRequest)

*Sheleter.UpdateFamiliarizationVisit* – złożona procedura, pozwalająca pracownikom na edycje wniosku o wizytę zapoznawczą – wariant 1 doprecyzowanie wizyty (przypisanie pracownika do wizyty oraz zaproponowanie terminu), wariant 2 to zamknięcie wizyty, czyli zmiana statusu na ‘Completed’ lub ‘Rejected’ wraz z podaniem przyczyny. Procedura zawiera rozbudowaną walidację wynikającą z logiki biznesowej. Standardowo walidujemy, czy pracownik i wizyty o podanych identyfikatorach istnieją oraz czy wizyta nie jest już zamknięta. Dalej walidujemy, czy podany zakres dat jest poprawny oraz co kluczowe, czy dany pracownik nie ma w proponowanym zakresie jakiejś innej wizyty – wykrywamy kolizje i nie dopuszczały takich wpisów.

```
-- Scenariusz pozytywny
EXEC Shelter.UpdateFamiliarizationVisit
    @FamiliarizationVisitId = @FamiliarizationVisitId,
    @EmployeeId = @TestEmployeeId,
    @StartDate = '2025-12-02 12:00',
    @EndDate = '2025-12-02 13:00',
    @VisitStatusId = '15fc7712-0554-48bd-b56f-b2f079fe5dc1';

SELECT fv.Id, fv.StartDate, fv.EndDate, fv.EmployeeId, vs.[Name], fv.RejectionReason
    FROM Shelter.FamiliarizationVisits AS fv
        JOIN Shelter.VisitStatuses AS vs ON vs.Id = fv.VisitStatusId
    WHERE fv.Id = @FamiliarizationVisitId;
```

Results Messages

Id	StartDate	EndDate	EmployeeId	Name	RejectionReason
F5274A97-D511-4A43-A737-6515EF626E54	2025-12-02 12:00:00.0000000	2025-12-02 13:00:00.0000000	5F5E17BA-AD01-4693-B561-D1AF584D55B9	Completed	NULL

**Zrzut ekranu nr 13:** weryfikacja pozytywnego scenariusza zgłoszenia chcącym odbycia wizyty zapoznawczej

```
-- Scenariusz negatywny (nieistniejący pracownik)
EXEC Shelter.UpdateFamiliarizationVisit
    @FamiliarizationVisitId = @FamiliarizationVisitId,
    @EmployeeId = '00000000-0000-0000-0000-000000000000',
    @StartDate = '2024-12-03 12:00',
    @EndDate = '2024-12-03 13:00',
    @VisitStatusId = '15fc7712-0554-48bd-b56f-b2f079fe5dc1';

-- Scenariusz negatywny (nieprawidłowy zakres dat)
EXEC Shelter.UpdateFamiliarizationVisit
    @FamiliarizationVisitId = @FamiliarizationVisitId,
    @EmployeeId = @TestEmployeeId,
    @StartDate = '2024-12-04 14:00',
    @EndDate = '2024-12-04 13:00',
    @VisitStatusId = '15fc7712-0554-48bd-b56f-b2f079fe5dc1';

-- Scenariusz negatywny (kolizja wizyt)
DECLARE @ConflictingVisitId uniqueidentifier = NEWID();
INSERT INTO Shelter.FamiliarizationVisits (Id, StartDate, EndDate, DogId, VisitorId, EmployeeId, VisitStatusId)
    VALUES (@ConflictingVisitId, '2024-12-02 12:30', '2024-12-02 13:30', @TestDogId, @VisitorId, @TestEmployeeId, '07fa688a-f536-4b3d-a286-828bc032a588');

EXEC Shelter.UpdateFamiliarizationVisit
    @FamiliarizationVisitId = @FamiliarizationVisitId,
    @EmployeeId = @TestEmployeeId,
    @StartDate = '2024-12-03 13:00',
    @EndDate = '2024-12-03 14:00',
    @VisitStatusId = '15fc7712-0554-48bd-b56f-b2f079fe5dc1';

100 %
```

Results Messages

(1 row affected)

Mag 50000, Level 14, State 1, Procedure Shelter.UpdateFamiliarizationVisit, Line 73 [Batch Start Line 163]  
Pracownik o podanym ID nie istnieje.  
Mag 50000, Level 14, State 1, Procedure Shelter.UpdateFamiliarizationVisit, Line 73 [Batch Start Line 163]  
Podano nieprawidłowy zakres dat

(1 row affected)

Mag 50000, Level 14, State 1, Procedure Shelter.UpdateFamiliarizationVisit, Line 73 [Batch Start Line 163]  
Wystąpiła kolizja dat. W podanym terminie pracownik odbiera inną wizytę.

**Zrzut ekranu nr 14:** reakcja systemu na błędne warianty  
(*UpdateFamiliarizationVisitRequest*)

*Sheleter.UpdateDogStatus* – procedura pozwala pracownikom na aktualizację statusu psa. Istnieją dwie sytuacje wykorzystania procedury. Gdy dodajemy psa do systemu otrzymuje on status ‘Registered’, wówczas pracownicy muszą dopełnić formalności (np. szczepienia, wpisy do innych systemów itp.) i dopiero po tym mogą zmienić status na

'Available' i udostępnić psa klientom. Drugi to zmiana statusu psa na uśpionego 'Euthanized'. Procedura zawiera walidację przed próbą aktualizacji nieistniejącego psa, który jest już adoptowany, czy podaniu nieprawidłowego id nowego statusu. Statusu nie możemy aktualizować manualnie na stany 'AcceptedToAdoption', czy 'Adopted' – te stany możemy otrzymać tylko w ramach innych procedur, gdy modyfikujemy wnioski adopcyjne, aby uniknąć niespójności danych.

Poniżej prezentuję testy procedury:

```
-- Scenariusz pozytywny: zmiana statusu dla nowo dodanego psa na dostępny (Available)
EXEC Shelter.UpdateDogStatus
    @DogId = @AlexDogId,
    @NewStatusId = '568ac4dd-5fa2-4d0d-b1c6-e13be5663d4c'

SELECT d.Id, d.[Name], d.[Breed], ds.[Name]
    FROM Shelter.Dogs AS d
        JOIN Shelter.DogStatuses AS ds ON ds.Id = d.DogStatusId
    WHERE d.[Name] = 'Alex' AND d.[Breed] = 'German Shepherd'
```

Id	Name	Breed	Name
B9163FA6-5307-402A-85AA-CF1074B3C09B	Alex	German Shepherd	Available

Zrzut ekranu nr 15: weryfikacja pozytywnego scenariusza aktualizacji statusu psa

```
-- Scenariusz negatywny: pies o podanym Id nie istnieje
EXEC Shelter.UpdateDogStatus
    @DogId = '00000000-1111-2222-3333-444444444444',
    @NewStatusId = '568ac4dd-5fa2-4d0d-b1c6-e13be5663d4c'

-- Scenariusz negatywny: próba zmiany statusu na inny niż "Available" lub "Euthanized"
EXEC Shelter.UpdateDogStatus
    @DogId = @AlexDogId,
    @NewStatusId = '1adf5a5b-342e-4341-9d37-9796367ab0a5'

-- Scenariusz negatywny: próba edycji statusu psa oddanego do adopcji
EXEC Shelter.UpdateDogStatus
    @DogId = 'BE2ECF0A-8A1E-4D30-B42A-945FAF8EE0A1',
    @NewStatusId = '1adf5a5b-342e-4341-9d37-9796367ab0a5'
```

Completion time: 2024-10-31T22:30:45.8781110+01:00
Msg 50000, Level 16, State 1, Procedure Shelter.UpdateDogStatus, Line 45 [Batch Start Line 481] Pies o podanym Id nie istnieje. Msg 50000, Level 16, State 1, Procedure Shelter.UpdateDogStatus, Line 45 [Batch Start Line 481] Niepoprawny status psa. Msg 50000, Level 16, State 1, Procedure Shelter.UpdateDogStatus, Line 45 [Batch Start Line 481] Niepoprawny status psa.

Zrzut ekranu nr 16: reakcja systemu na błędne warianty (UpdateDogStatus)

Sposób opakowania transakcji z obsługą rzucania wyjątków zaczerpnąłem z forum:  
<https://stackoverflow.com/questions/31274274/t-sql-throw-exception-within-a-transaction>

Skrypt z wszystkimi procedurami znajduje się w załączonym pliku: *Procedures.sql*.  
Testy w dedykowanym skrypcie: *ProceduresTests.sql*.

## Wyzwalače (ang. trigger'y):

Pierwszy trigger to *Shelter.Trigger\_OnAdoptionRecordStatusChange* (trigger typu AFTER UPDATE). Pilnuje on spójności danych przy zmianach statusu wniosku adopcyjnego. Gdy pracownik zdecyduje się zaakceptować klientowi wniosek adopcyjny, wówczas pozostałe wnioski o danego psa powinny zostać automatycznie odrzucone uzupełniając odpowiednio pole *RejectionReason*. Analogicznie wszelkie zaplanowane wizyty zapoznawcze powinny również być odwołane oraz zmianie powinien ulec status danego psa.

Przy zmianie statusu wniosku z Accepted na Realised, wniosek jest zamykany (zmiana statusu), a pies adoptowany (zmiana statusu na Adopted).

Poniżej prezentuję przykładowy test na zachowanie trigger'a (pełen kod testu dostępny jest w pliku *TriggersTests.sql*, tutaj pokażę tylko rezultat): dodanych zostaje 3 użytkowników, dwóch z nich złożyło wniosek o adopcję, a trzeci prośbę o wizytę zapoznawczą. Pracownik zdecydował się zaakceptować wniosek pierwszego użytkownika. Pozostali powinni mieć odrzucony wniosek i wizytę, status psa powinien się zmienić na 'AcceptedToAdoption':

```

EXEC Shelter.SubmitAdoptionRequest
    @UserId = @UserId1,
    @DogId = @PixelDogId;

EXEC Shelter.SubmitAdoptionRequest
    @UserId = @UserId2,
    @DogId = @PixelDogId;

EXEC Shelter.SubmitFamiliarizationVisitRequest
    @UserId = @UserId3,
    @DogId = @PixelDogId;

DECLARE @ExampleAdoptionRecordId uniqueidentifier;
SELECT @ExampleAdoptionRecordId = Id
    FROM Shelter.AdoptionRecords
        WHERE AdoptionCardId = (SELECT Id FROM Shelter.AdoptionCards WHERE UserId = @UserId1);

-- akceptacja wniosku pierwszego użytkownika
EXEC Shelter.UpdateAdoptionRecordStatus
    @AdoptionRecordId = @ExampleAdoptionRecordId,
    @NewStatusId='494d8c27-ad71-43e3-a216-f0be0450f135'

SELECT ar.Id, ar.RejectionReason, ads.[Name], ar.LastUpdateDate
    FROM Shelter.AdoptionRecords AS ar
        JOIN Shelter.AdoptionStatuses AS ads ON ads.Id = ar.AdoptionStatusId
            WHERE DogId = @PixelDogId

SELECT fv.Id, fv.EmployeeId, fv.VisitorId, fv.StartDate, fv.EndDate, fv.RejectionReason, vs.[Name]
    FROM Shelter.FamiliarizationVisits AS fv
        JOIN Shelter.VisitStatuses AS vs ON vs.Id = fv.VisitStatusId
            WHERE DogId = @PixelDogId

SELECT d.Id, d.[Name], ds.[Name]
    FROM Shelter.Dogs AS d
        JOIN Shelter.DogStatuses AS ds ON ds.Id = d.DogStatusId
            WHERE d.Id = @PixelDogId
GO

```

00 % ▾

Results		Messages	
1	58CF5904-BBB7-452D-B575-011E80FCB853C	RejectionReason	Name
1	NULL	Accepted	LastUpdateDate
2	1FAB8C18-83B1-4BE9-A445-D3C9E21F26D3	Wniosek innego klienta dla tego psa został zaak...	Rejected
			2024-10-29 21:59:00.2500000
1	0071C9C9-C88A-42B3-B319-FF60D3CE417F	EmployeeId	VisitorId
1	NULL	E759224F-1780-48F0-996D-72F46F6CE1CE	StartDate
1		NULL	EndDate
1			RejectionReason
1			Name
1	170DFB09-9716-4599-A73B-368F0ECSA6AD	Pixie	AcceptedToAdoption

**Zrzut ekranu nr 17:** poprawne stan systemu po wyzwoleniu trigger'a

Obserwujemy, że akceptowany wniosek dostał datę aktualizacji, pozostały wniosek i wizyta zostały odrzucone, a status psa zmienił się na 'AcceptedToAdoption'.

Drugi test to zmiana statusu wniosku z zaakceptowanego na zrealizowany. Trigger powinien w takiej sytuacji zaktualizować datę we wniosku oraz status psa:

```

EXEC Shelter.UpdateAdoptionRecordStatus
    @AdoptionRecordId = @ExampleAdoptionRecordId,
    @NewStatusId='e9497513-d728-4221-b48e-5b34d022b387'

SELECT ar.Id, ar.DogId, ads.[Name], ar.LastUpdateDate
FROM Shelter.AdoptionRecords AS ar
JOIN Shelter.AdoptionStatuses AS ads ON ads.Id = ar.AdoptionStatusId
WHERE ar.Id = @ExampleAdoptionRecordId

SELECT d.Id, d.[Name], ds.[Name]
FROM Shelter.Dogs AS d
JOIN Shelter.DogStatuses AS ds ON ds.Id = d.DogStatusId
WHERE d.Id = @DogId
%
```

Results			
Id	DogId	Name	LastUpdateDate
EA64EB67-F4F9-4D76-9B77-2F394842348D	528AA7B6-303D-4F34-8782-D17001EBD1CA	Realised	2024-10-29 18:54:33.1800000

Messages		
Id	Name	Name
528AA7B6-303D-4F34-8782-D17001EBD1CA	Pixie	Adopted

### Zrzut ekranu nr 18: poprawne stan systemu po wyzwoleniu trigger'a

Drugi trigger to *Shelter.Trigger\_UpdateDogStatus* (trigger typu AFTER UPDATE) – reaguje on na zmianę statusu psa na uśpiony „Euthanized” – w takim nieszczęśliwym wypadku musimy anulować wszystkie interakcje związane z psem (wnioski adopcyjne oraz wizyty zapoznawcze).

Test sprawdzający poprawne działanie trigger'a: dodane są dwa wnioski i jedna wizyta, zmieniliśmy status psa. Spodziewany wynik to anulowanie wszystkich zgłoszeń:

```

-- zmiana statusu na 'Euthanized'
EXEC Shelter.UpdateDogStatus
    @DogId = @HippieDogId,
    @NewStatusId = 'c482f14b-eb89-4c65-bb2c-2ef7b9fbf692'

SELECT ar.Id, ar.RejectionReason, ads.[Name], ar.LastUpdateDate
FROM Shelter.AdoptionRecords AS ar
JOIN Shelter.AdoptionStatuses AS ads ON ads.Id = ar.AdoptionStatusId
WHERE DogId = @HippieDogId

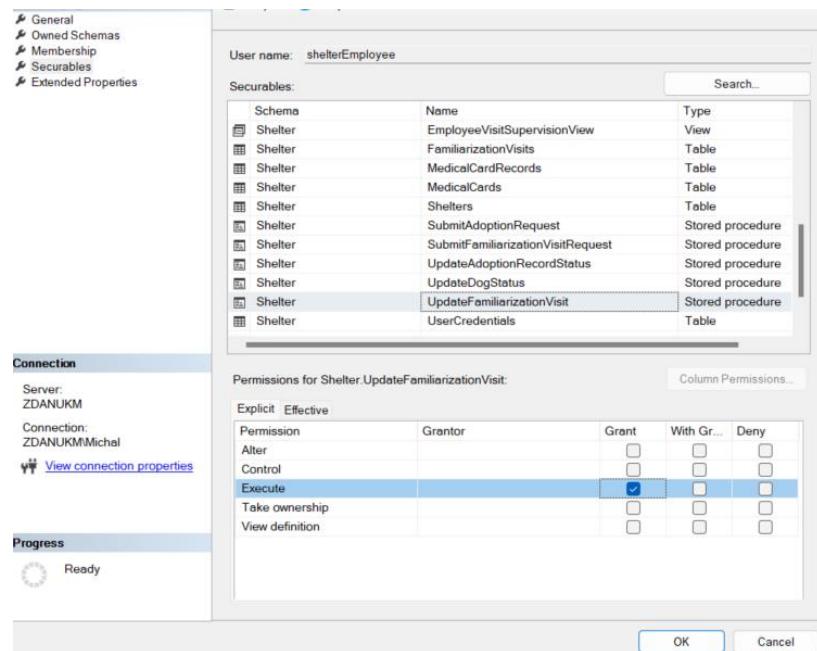
SELECT fv.Id, fv.EmployeeId, fv.VisitorId, fv.StartDate, fv.EndDate, fv.RejectionReason, vs.[Name]
FROM Shelter.FamiliarizationVisits AS fv
JOIN Shelter.VisitStatuses AS vs ON vs.Id = fv.VisitStatusId
WHERE DogId = @HippieDogId
%
```

Results					
Id	RejectionReason	Name	LastUpdateDate		
4713974B-1302-42EE-ABD4-80C7308C615B	Wniosek został odrzucony w związku z uśpieniem p...	Rejected	2024-10-29 22:02:35.4800000		
CAB9C4F7-A916-4DF0-9630-E0F15040EA74	Wniosek został odrzucony w związku z uśpieniem p...	Rejected	2024-10-29 22:02:35.4800000		

Messages						
Id	EmployeeId	VisitorId	StartDate	EndDate	RejectionReason	Name
65DF02E9-BC9E-4747-851A-281D5C0E69BF	NULL	800EB77D-BB29-4403-AF22-E58A886CB44D	NULL	NULL	Wniosek został odrzucony w związku z uśpieniem p...	Rejected

### Zrzut ekranu nr 19: poprawne stan systemu po wyzwoleniu trigger'a

Skrypt z trigger'ami znajduje się w pliku *Trigers.sql*. Testy trigger'ów wydzieliłem do oddzielnego skryptu *TriggersTests.sql*.



**Zrzut ekranu nr 20:** nadanie odpowiednich praw wykonywania procedur dla użytkowników

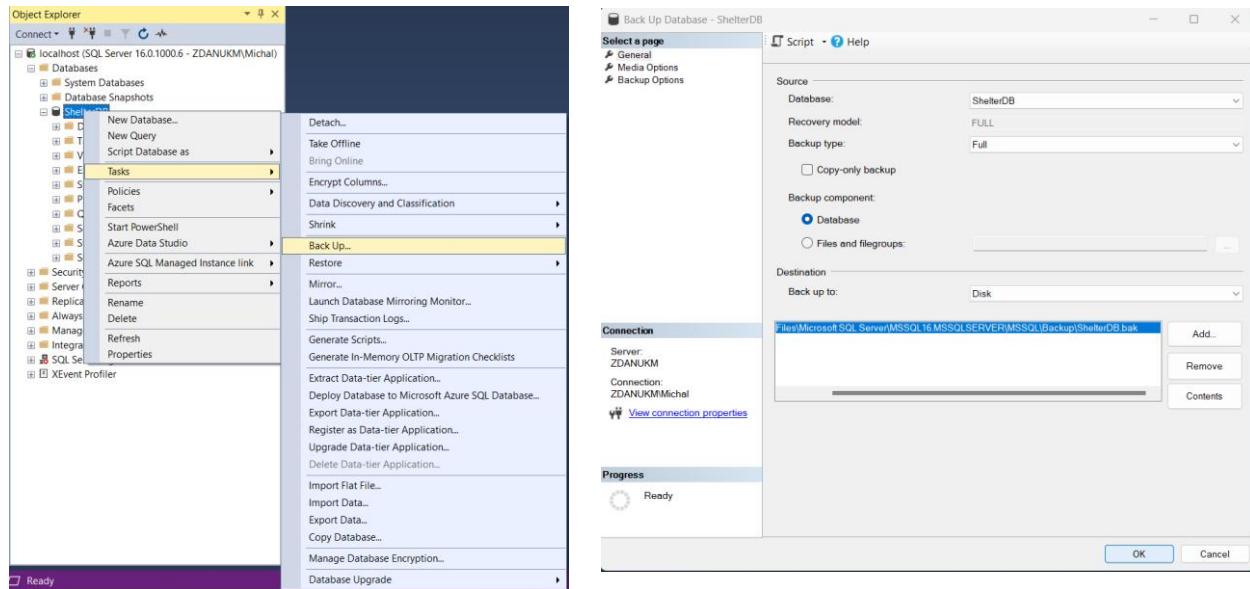
Po utworzeniu wszystkich elementów programowalnych i testach, poprzez eksplorator uprawnień rozdzieliłem dostęp do wykonywania procedur i funkcji użytkownikom shelterClient oraz shelterEmployee.

### Podsumowanie elementów programowalnych:

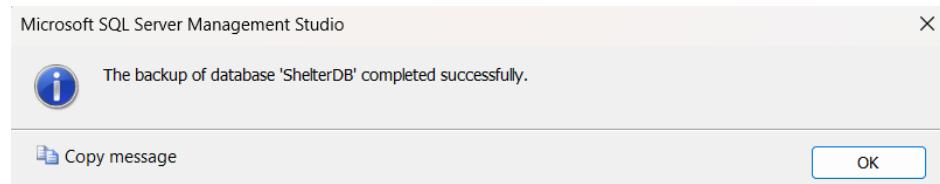
Zaimplementowałem 2 UDF, 3 procedury proste (dodanie rekordów do tabel z tabelami powiązanymi), 5 bardziej zaawansowanych procedur (zawierających transakcje oraz szczegółowe walidacje i obsługę nieprawidłowych danych) – obsługa wniosków adopcyjnych, wizyt zapoznawczych oraz zmiana statusu psa, 2 triggers – które są kluczowe dla logiki biznesowej i spójności danych całej aplikacji. Myślę, że te 12 elementów programowalnych w zupełności wystarczy by zapewnić pełną wartość funkcjonalną dla tego rozmiaru i stopnia zaawansowania projektu.

## Kopia zapasowa bazy danych:

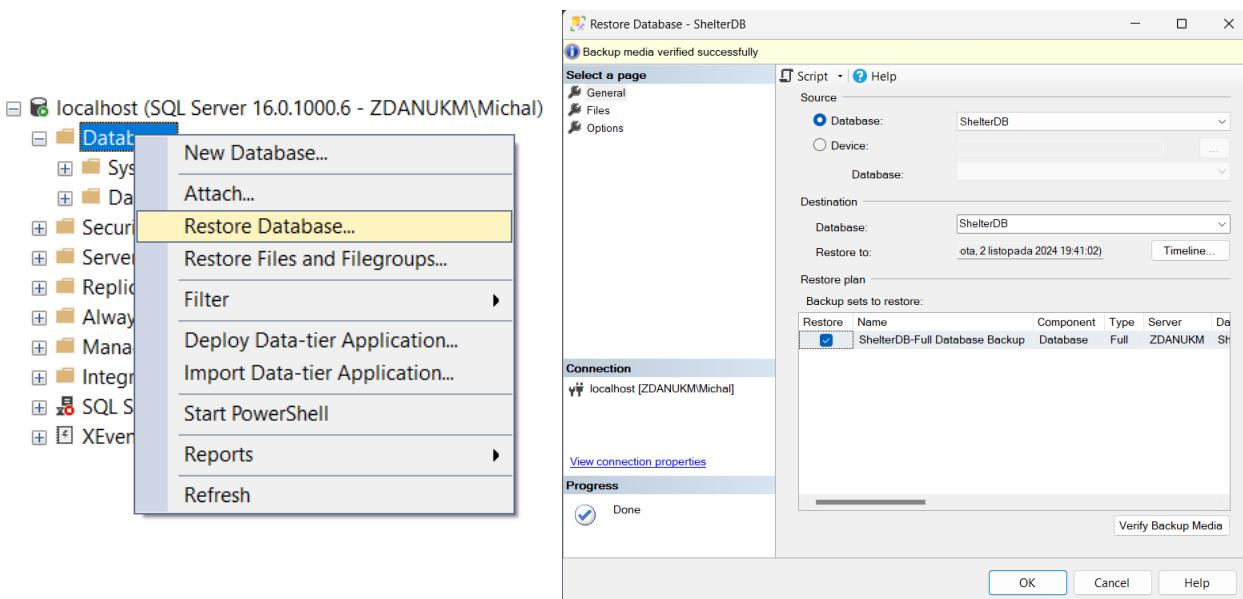
Proces wykonywania kopii oraz przywracania bazy zrealizowałem przy wykorzystaniu GUI w SSMS, ponieważ narzędzie oferuje proste i intuicyjne przejście przez proces.



Zrzut ekranu nr 21: tworzenie pliku z kopią .bak

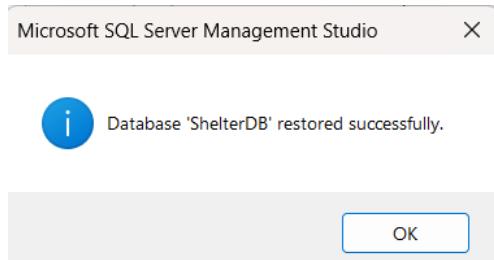


Zrzut ekranu nr 22: komunikat poprawnie wykonanej kopii bazy



Zrzuty ekranu nr 23, 24: przywracanie bazy

Przy graficznym wykonywaniu kopii SSMS automatycznie proponował zapisanie pliku .bak do standardowego katalogu kopii zapasowych bazy (C:\Program Files\Microsoft SQL Server\MSSQL16.MYSQLSERVER\MSSQL\Backup), przez co przy przywracaniu od razu widzimy na liście bazę. Gdybyśmy zgrywali bazę z zewnątrz / otrzymali od kogoś backup należy go umieścić w tym katalogu.



**Zrzut ekranu nr 25:** komunikat pomyślnego przywrócenia bazy

#### **Testy poprawności przywrócenia bazy danych:**

1. przejrzenie, czy przywrócone zostały wszystkie struktury, czyli tabele z ograniczeniami, funkcje, procedury, trigerr'y oraz użytkownicy z uprawnieniami
2. dowolne zapytanie celem sprawdzenia zawartości tabeli

A screenshot of the Microsoft SQL Server Management Studio Object Explorer. The left pane shows the database structure under "localhost (SQL Server 16.0.1000.6 - ZDANUKM\Michał)". It includes Databases (System Databases, Database Snapshots, ShelterDB), Tables (System Tables, FileTables, External Tables, Graph Tables, Shelter.Addresses, Shelter.AdoptionCards, Shelter.AdoptionRecords, Columns, Keys, Constraints, Triggers, Indexes, Statistics, Shelter.AdoptionStatuses, Shelter.DogCharacteristics, Shelter.Dogs, Shelter.DogStatuses, Shelter.FamiliarizationVisits, Shelter.MedicalCardRecords, Shelter.MedicalCards, Shelter.Shelters, Shelter.UserCredentials, Shelter.UserRoles, Shelter.Users, Shelter.VisitStatuses, Dropped Ledger Tables). The right pane shows the database structure details, including Views, System Views, and various shelter-related tables like ClientAdoptionHistoryView, ClientAvailableDogsView, EmployeeDogManagementView, EmployeeVisitSupervisionView, Dropped Ledger Views, External Resources, Synonyms, Programmability, Stored Procedures, System Stored Procedures, and Functions. Below these are the Security section and a list of users: dbo, guest, INFORMATION\_SCHEMA, shelterAdmin, shelterClient, shelterEmployee, and sys.

**Zrzuty ekranu nr 26, 27, 28:** przywrócone wszystkie struktury bazy oraz użytkownicy

	[Id]	[Name]	[Breed]	[EstimatedBirthDate]	[DogStatusId]	[ShelterId]
1	A13791B6-B9B8-40B6-8F63-0290BE97F9A6	Vassily	Beagle	2024-07-18	43051A55-003B-4F4B-A4D3-2BFF8F43027C	7AE8B17D-5BB4-4E8C-9EE4-F6B79A8F7B62
2	3D5E9852-66AC-45B6-9472-21DE41A24A98	Veronique	Labrador Retriever	2023-08-21	E3DB7A04-A42B-4820-B3F2-B6C29DFCA8C7	04408D9D-5A70-4A2B-9F2C-3D71AF43B293
3	3DAEC742-0FB8-4BEF-A7CB-22B18275A50F	Burton	Golden Retriever	2023-07-18	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	20C49F67-764B-4E80-969F-9EE046B4548A
4	E17E4E53-E247-40FD-B4B3-3F303B218EEA	Buddy	Golden Retriever	2022-07-07	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	20C49F67-764B-4E80-969F-9EE046B4548A
5	20116B0F-1753-450E-85AA-5682B807C9ABB	Belva	Golden Retriever	2024-04-03	43051A55-003B-4F4B-A4D3-2BFF8F43027C	04408D9D-5A70-4A2B-9F2C-3D71AF43B293
6	662B10F6-9D5F-40A7-94B0-5AE603F53360	Nikkie	Poodle	2023-11-18	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	20C49F67-764B-4E80-969F-9EE046B4548A
7	4A6871FB-BBA7-45AA-ADF7-63FA689F83FA	Earlie	German Shepherd	2024-06-15	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	7AE8B17D-5BB4-4E8C-9EE4-F6B79A8F7B62
8	BE2ECF0A-8A1E-4D30-B42A-945FAF8EE0A1	Jilleen	Labrador Retriever	2023-07-18	43051A55-003B-4F4B-A4D3-2BFF8F43027C	20C49F67-764B-4E80-969F-9EE046B4548A
9	C1EF83D2-C69E-4FE0-8C61-9CB52A4AAFA	Austen	German Shepherd	2023-07-18	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	7AE8B17D-5BB4-4E8C-9EE4-F6B79A8F7B62
10	44080FC9-1BB6-40FE-8438-A3776633D45	Hanson	Golden Retriever	2024-05-11	43051A55-003B-4F4B-A4D3-2BFF8F43027C	20C49F67-764B-4E80-969F-9EE046B4548A
11	4DE750BC-9136-4952-83E1-C13CE7D71EC9	Boone	German Shepherd	2022-06-16	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	C8EACCD5-23B5-4D87-A6B1-8F197121B37C
12	A45E3D6A-94A0-4DED-8D4D-D44C8C32D645	Becky	Poodle	2023-08-21	C482F14B-EB89-4C65-BB2C-2EF7B9FBF692	7AE8B17D-5BB4-4E8C-9EE4-F6B79A8F7B62
13	CAFE6B43-4BD7-40D0-9789-D982E4BBBF4	Padget	Labrador Retriever	2020-05-15	E3DB7A04-A42B-4820-B3F2-B6C29DFCA8C7	7AE8B17D-5BB4-4E8C-9EE4-F6B79A8F7B62
14	94C56C5C-2E53-4BDA-A64A-F75AB0A9F16B	Fred	German Shepherd	2023-07-26	568AC4DD-5FA2-4D0D-B1C6-E13BE5663D4C	04408D9D-5A70-4A2B-9F2C-3D71AF43B293
15	7E3F6CE2-2B1E-4200-8780-F938A8F97C1D	Simon	Golden Retriever	2023-07-21	C482F14B-EB89-4C65-BB2C-2EF7B9FBF692	7AE8B17D-5BB4-4E8C-9EE4-F6B79A8F7B62

**Zrzuty ekranu nr 29:** zapytanie o listę psów zwracane dane, co potwierdza przywrócenie całej bazy

## Drugi SZBD:

Wybrany przeze mnie drugi system to PostgreSQL, ponieważ jest on również w TOP4 najczęściej wykorzystywanych baz relacyjnych i jest powszechnie używany na rynku. Instalacja na Windowsie nie była skomplikowana, wymagała „przeklinania” instalatora, gdzie należało skonfigurować dostęp użytkownika admina, ustawić port i lokalizację. Skorzystałem z poradnika:

<https://ultahost.com/knowledge-base/install-postgresql-on-windows/>

```
C:\Program Files\PostgreSQL\17\bin>psql --version
psql (PostgreSQL) 17.0
```

```
C:\Program Files\PostgreSQL\17\bin>
```

**Zrzuty ekranu nr 30:** weryfikacja instalacji Postgresa

## Przegląd głównych różnic w dialektach T-SQL oraz PL-SQL:

Przede wszystkich główne różnice wiążą się z inną składnią: tworzenie struktur, zmienne, bazowe funkcje systemowe, znaczek końca bloku etc. Również niektóre typy danych występują tylko w jednym SZDB i należy szukać odpowiedników i dokonać konwersji (najważniejsze zmiany w moim przypadku dotyczą unikalnych identyfikatorów i dat). Co więcej mechanika obsługi bloków TRY CATCH z obsługą walidacji za pomocą wyjątków jest inna. Trigger'y również się różni, ponieważ nie mogą zawierać ciała zapytania, a wywoływać dedykowaną funkcję.

	T-SQL	PL-SQL
Typy danych	<ul style="list-style-type: none"> <li>• UNIQUEIDENTIFIER oraz polecenie NEWID()</li> <li>• DATETIME2 oraz polecenie GETDATE()</li> </ul>	<ul style="list-style-type: none"> <li>• UUID oraz polecenie uuid_generate_v4()</li> <li>• TIMESTAMP oraz polecenie NOW()</li> </ul>
Sprawdzenie czy pole jest nullem	ISNULL(expression, value)	COALESCE(expression, value)
Definiowanie z aktualizacją procedur i funkcji	CREATE OR ALTER PROCEDURE	CREATE OR REPLACE PROCEDURE
Instrukcje warunkowe i obsługa błędów	Blok kodu BEGIN TRY ... END TRY BEGIN CATCH ... END CATCH, gdzie w bloku TRY w instrukcjach wyrzucamy RAISEERROR('msg')	Jeden blok BEGIN END, pod koniec bloku – tutaj wyrzucamy RAISE EXCEPTION 'msg', subblok EXCEPTION
Prefix właściciela	dbo	public
Nazwy tabel/schematów	-	w cudzysłowach „”
Terminator	GO	;
Zmienne (deklaracje, przypisywanie wartości)	<pre>DECLARE @firstName NVARCHAR(50); DECLARE @lastName NVARCHAR(50) = 'Kowalski'</pre>	<pre>DECLARE first_name VARCHAR(50) := 'Jan'; wartości last_name VARCHAR(50);</pre>
Konkatenacja ciągu znaków	SET @fullName = @firstName + '' + @lastName;	full_name := first_name    ''    last_name;
Trigerr'y	Mogą zawierać bezpośrednio w ciele logikę i zapytania	Muszą wywoływać wydzieloną, dedykowaną funkcję z logiką biznesową

Lista źródeł, z których dowiadywałem się jakie różnice występują w dialektach / jaka jest obsługa konkretnych struktur:

- <https://stackoverflow.com/questions/1043265/what-is-the-difference-between-sql-pl-sql-and-t-sql>
- <https://hevodata.com/learn/t-sql-vs-p-l-sql/>
- <https://bryteflow.com/sql-server-vs-postgres-how-to-migrate-sql-to-postgresql/>
- [https://wiki.postgresql.org/wiki/Microsoft\\_SQL\\_Server\\_to\\_PostgreSQL\\_Migration\\_by\\_Ian\\_Harding](https://wiki.postgresql.org/wiki/Microsoft_SQL_Server_to_PostgreSQL_Migration_by_Ian_Harding)
- <https://neon.tech/postgresql/postgresql-plpgsql/postgresql-exception>

## Narzędzia do migracji danych między wybranymi SZDB:

Migracja bazy danych z systemu MSSQL do PostgreSQL wymaga przede wszystkim przekształcenia schematów danych, procedur oraz innych funkcji charakterystycznych dla SZBD. Na rynku mamy dostęp do kilku narzędzi, które pozwalają uprościć ten proces i uniknąć całkowicie manualnego przeprowadzania migracji. Do najpopularniejszych należą:

1. **pgLoader** – narzędzie open-source, które automatyzuje proces migracji danych z różnych źródeł, w tym MSSQL do PostgreSQL. Oferuje automatyczną konwersję schematów, typów oraz transformacje danych, co znacząco może skrócić proces migrowania. Należy jednak pamiętać, że pgLoader nie wspiera migracji procedur składowanych, co może wymagać dodatkowego wysiłku w celu ich ręcznego odtworzenia w docelowej bazie danych.
  2. **Babelfish** - narzędzie ETL, które umożliwia łatwą migrację danych między różnymi systemami baz danych. Charakteryzuje się intuicyjnym interfejsem graficznym, który pozwala na definiowanie procesów migracji i transformacji danych. Dodatkowo, Babelfish stara się optymalizować migracje dużych zbiorów danych, co przyspiesza proces i minimalizuje ryzyko błędów. Dodatkowo, obsługuje różne formaty danych i może być zintegrowany z innymi narzędziami do przetwarzania danych, aby zwiększyć swoją funkcjonalność.
  3. **Sqlserver2pgsql** – narzędzie open-source, które konwertuje schemat bazy SQL Server na PostgreSQL oraz tłumaczy dane z wykorzystaniem Pentaho Data Integrator. Narzędzie ma na celu zapewnienie płynnego przejścia z SQL Server do PostgreSQL, obsługując powszechnie różnice w schematach i odpowiednio tłumacząc typy danych. Niestety, podobnie jak inne narzędzia migracyjne, Sqlserver2pgsql nie migruje procedur składowanych, co może stanowić wyzwanie dla użytkowników polegających na skomplikowanej logice biznesowej zawartej w swoich bazach danych.
- 
- <https://pgloader.readthedocs.io/en/latest/ref/mssql.html>
  - <https://www.albatrossmigrations.com/database-migration-guides/sql-server-to-postgresql>
  - <https://www.vinchin.com/database-tips/how-to-migrate-sql-server-to-postgresql-in-2-ways.html>

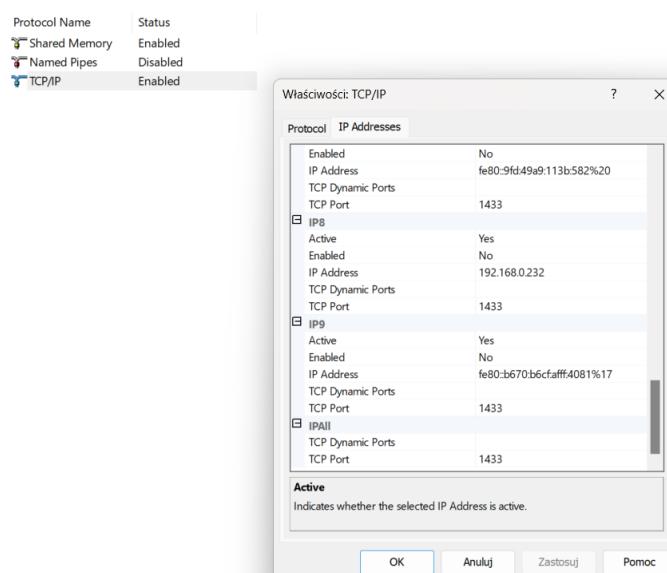
## Migracja:

W związku z tym, że narzędzie pgloader ma bardzo słabą kompatybilność z Windows'em zdecydowałem się podjąć próby uruchomienia jego w dockerze. Oczywiście nie było to proste i po drodze otrzymałem szereg różnych błędów (naprawdę dużo). Przede wszystkim należało:

- do pliku konfiguracyjnego `pg_hba.conf` Postgres'a należało dodać wpis umożliwiający łączenie się do bazy z dockera:

```
host all all 192.168.0.0/24 md5
```

- w ustawieniach konfiguratora Sql Server Configuration Manager odblokować połączenia przez protokół TCP/IP, dodać w sekcji IPAOO port 1433, a na localhostie zaznaczyć flagę enabled
- w serwerze MSSQL w sekcji Connections zaznaczyć flagę „Allow remote connections to this server”
- utworzyć server Postgresa o nazwie hosta localhost wraz z pustą bazą ShelterDB
- przetestować połączenia do baz z poziomu terminala komendami psql oraz sqlcmd



## Zrzuty ekranu nr 31: sekcja konfiguracji

Następnie uruchomiłem obraz z narzędziem pgLoader polecienniem:  
`docker run -it --rm dimitri/pgloader bash`

Dalej przygotowałem plik test.conf, który ma określić szczegóły konfiguracji i zasila program pgLaoder:

```
load database
```

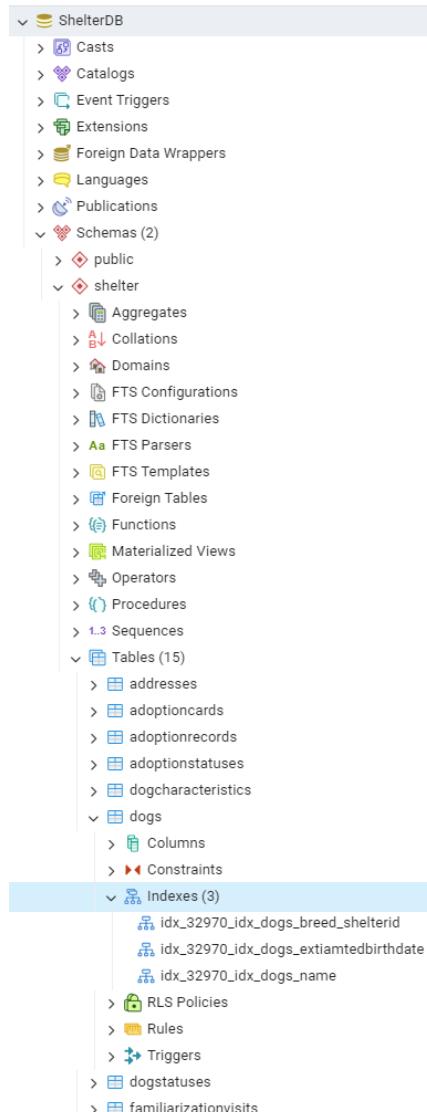
```
from mssql://shelterAdmin:X@host.docker.internal:1433/ShelterDB
```

```
into postgres://postgres:X@host.docker.internal:5432/ShelterDB
```

```
ALTER SCHEMA 'Shelter' RENAME TO 'shelter';
```

```
Terminal
root@f7e9373f5986:/# pgloader /tmp/test.load
2024-11-04T17:05:22.000000Z LOG pgloader version "3.6.7-devel"
2024-11-04T17:05:22.210825Z LOG Migrating from #<MSSQL CONNECTION mssql://shelterAdmin@host.docker.internal:1433/ShelterDB {1007E34813}>
2024-11-04T17:05:22.210825Z LOG Migrating to #<PGSQL CONNECTION postgres://postgres@host.docker.internal:5432/ShelterDB {1007E35853}>
Max connections reached, increase value of TDS_MAX_CONN
shelter.shelters      0      4     0.4 kB      0.220s
shelter.users         0      8     0.7 kB      0.350s
shelter.userroles     0      3     0.1 kB      0.210s
shelter.visitstatuses 0      3     0.1 kB      0.250s
-----
COPY Threads Completion   0      4     0.478s
    Create Indexes          0      22    0.250s
Index Build Completion   0      22    0.248s
    Reset Sequences         0      0     0.000s
        Primary Keys         0      14    0.050s
    Create Foreign Keys     0      12    0.050s
    Create Triggers          0      0     0.000s
    Install Comments         0      0     0.000s
-----
Total Import time       ✓     124   13.8 kB      1.128s
root@f7e9373f5986:/#
```

### Zrzuty ekranu nr 32: uruchomienie migracji w pgLoader



### Zrzuty ekranu nr 33: widok poprawnie zmigrowanej struktury bazy w Postgresie (ETL poprawnie przeniosły wszystkie tabele, klucze, indeksy i ograniczenia)

Poprawność wykonania migracji danych sprawdziłem prostym SELECTem na tabeli shelters:

The screenshot shows a PostgreSQL client interface. At the top, there are tabs for 'Query' (which is selected) and 'Query History'. Below that is a code editor containing the following SQL query:

```
1 v SELECT * FROM shelter.shelters
2 ORDER BY id ASC LIMIT 100
3
```

Below the code editor is a toolbar with various icons for file operations like saving, opening, and printing.

At the bottom of the interface is a table titled 'shelters' with four columns: 'id' (PK), 'name', and 'addressid'. The table contains four rows of data:

	id [PK] uuid	name	addressid uuid
1	04408d9d-5a70-4a28-9f2c-3d71af43b293	Wagging Hearts Dog Rescue	1d7a7639-89f1-4051-8b2b-6b73781b03...
2	20c49f67-764b-4e80-969f-9ee046b4548a	Furry Friends Sanctuary	c24e37e6-f31c-4116-8247-85e4b3ef7a53
3	7ae8b17d-5bb4-4e8c-9ee4-f6b79a8f7b62	Paws & Haven Rescue Center	7ae8b17d-5bb4-4e8c-9ee4-f6b79a8f7b62
4	c8eacdd5-23b5-4d87-a681-8f197121b37c	Happy Tails Dog Shelter	4e5b894b-c28f-4a93-9a01-77c5f56b80a9

Zrzut ekranu nr 34: zawartość tabeli shelters

Co ważne migracja udała się, ale jest ona częściowa i wynika to z ograniczeń narzędzia. Udało się poprawnie zmigrować cały schemat bazy – poprawnie skopiowane zostały tabele, ograniczenia na nich, klucze główne i obce oraz indeksy. Konwersja typów również przebiegła pomyślnie. Dane także zostały przekopiowane co jest kluczowe. Niestety pgLoader nie umożliwia pełnej migracji – zabrakło przeniesienia funkcji, widoków, procedur oraz triggerów (na stronie producenta wyczytałem, że funkcje programowalne nie są wspierane przy migracji). Przypuszczam, że wynika to ze zbyt dużych różnic między dialekta i SZDB. Mimo wszystko taka migracja (przy posiadaniu wiedzy i praktycznego doświadczenia, którego mi brakowało/brakuje) jest satysfakcjonująca i pozwala skrócić cały proces. Elementy programowalne dokończę migrować manualnie – wygeneruję przez SSMS skrypty SQL, które dostosuję pod dialekt PL-SQL i uruchomię w postgresie.

Manualne poprawki zostały zastosowane poprzez uruchomienie skryptów generujących widoki, funkcje UDF, procedury i trigger'y. Dostępne są one w kilku plikach:

- *Views\_translated\_to\_PLSQL.sql*.
- *UDF\_translated\_to\_PLSQL.sql*.
- *Procedures\_translated\_to\_PLSQL.sql*.
- *Triggers\_translated\_to\_PLSQL.sql*.

Poniżej zaprezentuję rezultat poprawnie domigrowanych manualnie elementów, z którymi nie poradziło sobie narzędzie ETL.

Query History

```
1 v SELECT * FROM shelter.clientavailabledogsview
2 v LIMIT 100
3
```

Data Output Messages Notifications

	dogname text	dogbreed text	estimatedbirthyear numeric	estimatedbirthmonth numeric	dogweight integer	dogdescription text	sheltername text
1	Earlie	German Shephe...	2024	6	5	[null]	Paws & Haven Resc
2	Buddy	Golden Retriever	2022	7	6	[null]	Furry Friends Sanct
3	Boone	German Shephe...	2022	6	8	Affectionate and gentle, perfect for families and active owners.	Happy Tails Dog Sh
4	Burton	Golden Retriever	2023	7	14	Curious and energetic, known for its excellent sense of smell.	Furry Friends Sanct
5	Fred	German Shephe...	2023	7	7	A small, cheerful dog known for its long, flowing coat.	Wagging Hearts Do
6	Nikkie	Poodle	2023	11	8	Extremely smart and active, excelling in herding and agility tasks.	Furry Friends Sanct
7	Austen	German Shephe...	2023	7	12	A small, sturdy dog with a lovable personality and signature bat-like ea...	Paws & Haven Resc

Zrzuty ekranu nr 35: poprawnie zmigrowany widok clientavailabledogsview

Query History

```
1 -- Ile dni upłyнуło od ostatniego odrobaczania psa X
2 v SELECT shelter.dayssincelastmedicalprocedure('4de75dbc-9136-4952-83e1-c13ced7d1ec9',
3 'Flea & Tick Prevention');
```

Data Output Messages Notifications

	dayssincelastmedicalprocedure integer
1	244

Zrzuty ekranu nr 36: poprawnie działająca funkcja obliczająca ile dni upłynęło od zabiegu dla podanego psa

Query History

```
1 v CALL shelter.addshelter(
2     paramname => 'Happy Tails Shelter',
3     paramaddressid => 'c24e37e6-f31c-4116-8247-85e4b3ef7a53'
4 );
5
6 -- Verify by checking the shelters table
7 SELECT * FROM shelter.shelters WHERE name = 'Happy Tails Shelter';
8
```

Data Output Messages Notifications

	id [PK] uuid	name text	addressid uuid
1	d63ef689-6490-4d91-86ac-989f4216c47f	Happy Tails Shelter	c24e37e6-f31c-4116-8247-85e4b3ef7a53

Zrzuty ekranu nr 37: poprawnie działająca prosta procedura dodająca schronisko

Przedstawię dokładne testy tylko jednej bardziej zaawansowanej procedury, testy do pozostałych wyglądająby analogicznie. Wybrałem zgłaszanie wniosku adopcyjnego:

The screenshot shows the pgAdmin interface with a query editor and a data output window. The query editor contains a PL/pgSQL procedure for submitting an adoption request. The data output window shows a single row with a count of 1, indicating one record was affected.

```

Query Query History
1 -- procedura do składania wniosków adopcyjnych - Shelter.SubmitAdoptionRequest
2 -- Scenariusz pozytywny (pies istnieje i jest dostępny, użytkownik istnieje, jest klientem schroniska)
3 CALL shelter.submitadoptionrequest(
4     paramuserid := '470e2057-1fd6-48ba-ba75-2c4c1303a5d2',
5     paramdogid := '3daec742-0fbb-4bef-a7cb-22b18275a50f'
6 );
7 |
8 v SELECT COUNT(*)
9      FROM shelter.adoptionrecords AS ar
10 JOIN shelter.adoptioncards AS ac ON ac.id = ar.adoptioncardid
11 WHERE ac.userid = '470e2057-1fd6-48ba-ba75-2c4c1303a5d2'
12   AND ar.dogid = '3daec742-0fbb-4bef-a7cb-22b18275a50f';
13
Data Output Messages Notifications
count bigint
1 1

```

### Zrzuty ekranu nr 38: scenariusz pozytywny

The screenshot shows the pgAdmin interface with a query editor and a messages window. The query editor contains a PL/pgSQL procedure for submitting an adoption request with a non-existing dog ID. The messages window displays an error message: "Error in submit\_adoption\_request: Pies o podanym ID nie istnieje." (Dog with the given ID does not exist).

```

14 -- Scenariusz negatywny (pies nie istnieje)
15 v CALL shelter.submitadoptionrequest(
16     paramuserid := '470e2057-1fd6-48ba-ba75-2c4c1303a5d2',
17     paramdogid := '00000000-1111-2222-3333-444444444444'
18 );
19
Data Output Messages Notifications
ERROR: Error in submit_adoption_request: Pies o podanym ID nie istnieje.
CONTEXT: funkcja PL/pgSQL shelter.submitadoptionrequest(uuid,uuid), wiersz 53 w RAISE
BŁĄD: Error in submit_adoption_request: Pies o podanym ID nie istnieje.
SQL state: P0001

```

### Zrzuty ekranu nr 39: scenariusz negatywny: pies o podanym id nie istnieje

The screenshot shows the pgAdmin interface with a query editor and a messages window. The query editor contains a PL/pgSQL procedure for submitting an adoption request with a dog that is not available. The messages window displays an error message: "Error in submit\_adoption\_request: Pies nie jest dostępny do adopcji." (Dog is not available for adoption).

```

20 -- Scenariusz negatywny (pies nie jest dostępny - inny status niż Available)
21 v CALL shelter.submitadoptionrequest(
22     paramuserid := '470e2057-1fd6-48ba-ba75-2c4c1303a5d2',
23     paramdogid := 'a13791b6-b9b8-40b6-8f63-0290be97f9a6'
24 );
25
Data Output Messages Notifications
ERROR: Error in submit_adoption_request: Pies nie jest dostępny do adopcji.
CONTEXT: funkcja PL/pgSQL shelter.submitadoptionrequest(uuid,uuid), wiersz 53 w RAISE
BŁĄD: Error in submit_adoption_request: Pies nie jest dostępny do adopcji.
SQL state: P0001

```

### Zrzuty ekranu nr 40: scenariusz negatywny: wybrany pies jest niedostępny

The screenshot shows the pgAdmin interface with a query editor and a messages window. The query editor contains a PL/pgSQL procedure for submitting an adoption request with a user who is not a client. The messages window displays an error message: "Error in submit\_adoption\_request: Użytkownik nie jest klientem schroniska." (User is not a shelter client).

```

26 -- Scenariusz negatywny (użytkownik nie jest klientem)
27 v CALL shelter.submitadoptionrequest(
28     paramuserid := '5459fd6a-ec98-4e04-9cce-83e04b9ae948',
29     paramdogid := '3daec742-0fbb-4bef-a7cb-22b18275a50f'
30 );
31
Data Output Messages Notifications
ERROR: Error in submit_adoption_request: Użytkownik nie jest klientem schroniska.
CONTEXT: funkcja PL/pgSQL shelter.submitadoptionrequest(uuid,uuid), wiersz 53 w RAISE
BŁĄD: Error in submit_adoption_request: Użytkownik nie jest klientem schroniska.
SQL state: P0001

```

### Zrzuty ekranu nr 41: scenariusz negatywny: użytkownik nie jest klientem schroniska

Migracja trigerr'ów wymagała więcej wysiłków, ponieważ muszą one wywoływać funkcje (w tym dialekcie nie możemy bezpośrednio w ciele trigger'a umieszczać logiki), dlatego wydzielone zostały odpowiednie bloki funkcji. Poza tym należało zastosować zmiany wynikające z innej obsługi wyjątków.

Poniżej zaprezentuję poprawne działanie trigerr'a reagującego na zmianę statusu wniosku adopcyjnego. Scenariusz: dwóch klientów zgłosiło wniosek adopcyjny na danego psa, klient trzeci złożył prośbę o odbycie wizyty zapoznawczej (analogiczny scenariusz zrealizowany był w systemie MSSQL).

```

80 v SELECT ar.id, ar.rejectionreason, ads.name AS status_name, ar.lastupdatedate
81   FROM shelter.adoptionrecords AS ar
82     JOIN shelter.adoptionstatuses AS ads ON ads.id = ar.adoptionstatusid
83 WHERE ar.dogid = '94C56C5C-2E53-4BDA-A64A-F75AB0A9F16B';
84
  
```

Data Output Messages Notifications

	id uuid	rejectionreason text	status_name text	lastupdatedate timestamp with time zone
1	a3c07edf-95b8-42fa-8a3e-021d1f0b07cc	[null]	Accepted	2024-11-04 22:15:52.435562+01
2	1e5eb45c-42d1-4d87-9ec0-7961f8de5a61	Wniosek innego klienta dla tego psa został zaakceptowa...	Rejected	2024-11-04 22:15:52.435562+01

**Zrzuty ekranu nr 42:** stan wniosków adopcyjnych po akceptacji wybranego

```

85 v SELECT fv.id, fv.employeeid, fv.visitordid, fv.startdate, fv.enddate, fv.rejectionreason, vs.name AS visit_status
86   FROM shelter.familiarizationvisits AS fv
87     JOIN shelter.visitstatuses AS vs ON vs.id = fv.visitstatusid
88 WHERE fv.dogid = '94C56C5C-2E53-4BDA-A64A-F75AB0A9F16B';
89
  
```

Data Output Messages Notifications

	visitorid uuid	startdate timestamp with time zone	enddate timestamp with time zone	rejectionreason text	visit_status text
1	iae948	2024-07-09 12:00:00+02	2024-07-09 13:00:00+02	[null]	Completed
2	721680ec-b8db-80e5-50492dc9a923	[null]	[null]	Wniosek innego klienta dla tego psa został zaakceptowa...	Rejected

**Zrzuty ekranu nr 43:** poprawna aktualizacja wizyt (nie zmienione wizyty ukończone, wizyta planowana zamknięte i uzupełniony powód odwołania)

```

90 v SELECT d.id, d.name, ds.name AS status_name
91   FROM shelter.dogs AS d
92     JOIN shelter.dogstatuses AS ds ON ds.id = d.dogstatusid
93 WHERE d.id = '94C56C5C-2E53-4BDA-A64A-F75AB0A9F16B';
94
  
```

Data Output Messages Notifications

	id uuid	name text	status_name text
1	94c56c5c-2e53-4bda-a64a-f75ab0a9f16b	Fred	AcceptedToAdoption

**Zrzuty ekranu nr 44:** poprawna aktualizacja statusu psa przez trigger

Ostatnim elementem migracji było odwzorowanie użytkowników i nadanie im odpowiednich uprawnień do widoków, procedur oraz funkcji.

Problemy: w chwili testów procedury otrzymałem błąd z brakiem dostępu do tabeli Dogs (postgres chciał pośrednio uprawnienia do tej tabeli, aby wykonać zapytanie w procedurze). Nie chcemy oczywiście dawać dostępów do wszystkich tabel, bo grozi to

powstaniem niespójności i jest to wyciek uprawnień, chcemy dawać tym użytkownikom dostęp do bazy poprzez procedury, funkcje i widoki. W związku z tych zmodyfikowałem procedury i dodałem dopisek SECURITY DEFINER przy nagłówkach, aby wykonywać procedurę z uprawnieniami ownera i nie rozszerzać nadmiernie uprawnień. Dopiero po tej zmianie mogłem poprawnie korzystać z procedur.

```
ShelterDB=# CREATE USER shelter WITH PASSWORD '████████';
CREATE ROLE
ShelterDB=# ALTER ROLE shelterclient WITH LOGIN;
ALTER ROLE
ShelterDB=# GRANT CONNECT ON DATABASE "ShelterDB" TO shelterclient;
GRANT
ShelterDB=# ALTER ROLE shelterclient SET search_path TO shelter;
ALTER ROLE
ShelterDB=# GRANT SELECT ON "shelter".clientadoptionhistoryview TO shelterclient;
GRANT
ShelterDB=# GRANT USAGE ON SCHEMA shelter TO shelterclient;
GRANT
ShelterDB=# GRANT SELECT ON "shelter".clientavailabledogsview TO shelterclient;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE "shelter".submitadoptionrequest(uuid, uuid) TO shelterclient;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE "shelter".submitfamiliarizationvisitrequest(uuid, uuid) TO shelterclient;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE "shelter".submitdogstatus(uuid, integer) TO shelterclient;
GRANT
ShelterDB=# GRANT EXECUTE ON FUNCTION shelter.dayssincefirstmedicalprocedure(UUID, VARCHAR) TO shelterclient;
GRANT
ShelterDB=# GRANT EXECUTE ON FUNCTION shelter.dayssincelastadoptionrequest(UUID) TO shelterclient;
GRANT
```

### Zrzuty ekranu nr 45: tworzenie użytkownika shelterclient oraz nadanie uprawnień

```
ShelterDB=> SELECT * FROM "shelter".Users;
BŁĄD: permission denied for table users
ShelterDB=> SELECT * FROM "shelter".clientadoptionhistoryview;
           clientfirstname | clientlastname | dogname | requestlastupdate | adoptionstatus | rejectionreason
-----+-----+-----+-----+-----+-----+
Timofei   | Frudd      | Vassily | 2024-09-08 00:00:00+02 | Realised
Karylin   | Freire     | Burton   | 2024-11-04 21:29:02.038398+01 | FormApplied
Karylin   | Freire     | Belva    | 2024-10-02 00:00:00+02 | Realised
Waylon    | Hause      | Earlie   | 2024-10-12 00:00:00+02 | Rejected   | Adoption request was denied due to the home not meeting necessary safety requirements for the dog.
Timofei   | Frudd      | Jilleen   | 2024-09-05 00:00:00+02 | Accepted
Son       | Ardling    | Hanson   | 2024-10-06 00:00:00+02 | Realised
Piotr     | Kowalski   | Bella    | 2024-11-04 22:07:14.786583+01 | Rejected   | Wniosek innego klienta dla tego psa zosta| zaakceptowany
Anna      | Nowak      | Bella    | 2024-11-04 22:07:14.786583+01 | Accepted
Michał   | Wiśniewski | Fred     | 2024-11-04 22:15:52.435562+01 | Rejected   | Wniosek innego klienta dla tego psa zosta| zaakceptowany
Katarzyna | Nowicka    | Fred     | 2024-11-04 22:15:52.435562+01 | Accepted
(10 wierszy)

ShelterDB=> CALL shelter.submitadoptionrequest('00000000-0000-0000-0000-000000000111', '00000000-0000-0000-0000-000000000222');
BŁĄD: Error in submit_adoption_request: Pies o podanym ID nie istnieje.
```

### Zrzuty ekranu nr 46: testy uprawnień dla shelterclient

```
ShelterDB=# CREATE USER shelteremployee WITH PASSWORD '████████';
CREATE ROLE
ShelterDB=# ALTER ROLE shelteremployee SET search_path TO shelter;
ALTER ROLE
ShelterDB=# GRANT CONNECT ON DATABASE "ShelterDB" TO shelteremployee;
GRANT
ShelterDB=# GRANT USAGE ON SCHEMA shelter TO shelteremployee;
GRANT
ShelterDB=# GRANT SELECT ON "shelter".employeedogmanagementview TO shelteremployee;
GRANT
ShelterDB=# GRANT SELECT ON "shelter".employeevisitsupervisionview TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.addclient(character varying, character varying, character varying, character varying, date) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.adddog(character varying, character varying, date, uuid, integer, character varying) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.addshelter(character varying, uuid) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.submitadoptionrequest(uuid, uuid) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.updateadoptionrecordstatus(uuid, uuid) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE "shelter".submitfamiliarizationvisitrequest(uuid, uuid) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.updatefamiliarizationvisit(uuid, uuid, timestamp with time zone, timestamp with time zone, uuid, text) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON PROCEDURE shelter.updatedogstatus(uuid, uuid) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON FUNCTION shelter.dayssincefirstmedicalprocedure(UUID, VARCHAR) TO shelteremployee;
GRANT
ShelterDB=# GRANT EXECUTE ON FUNCTION shelter.dayssincelastadoptionrequest(UUID) TO shelteremployee;
GRANT
```

### Zrzuty ekranu nr 47: tworzenie użytkownika shelteremployee oraz nadanie uprawnień

```
PS C:\Users\Michał> psql -h localhost -U shelteremployee -d ShelterDB
Hasło użytkownika shelteremployee:

psql (17.0)
OSTRZEŻENIE: strona kodowa konsoli (852) jest różna od kodowania Windows (1250)
8-bitowe znaki mogą nie wyglądać poprawnie. Przejrzyj odnośną
stronę "Notes for Windows users" by poznac szczegóły.

Wpisz "help" by uzyskać pomoc.

ShelterDB=> SELECT * FROM "shelter".Users;
ERROR: permission denied for table users
ShelterDB=> SELECT "shelter".dayssincelastmedicalprocedure('4de75dbc-9136-4952-83e1-c13ced7dlec9', 'Flea & Tick Prevention');
dayssincelastmedicalprocedure
-----
245
(1 wiersz)

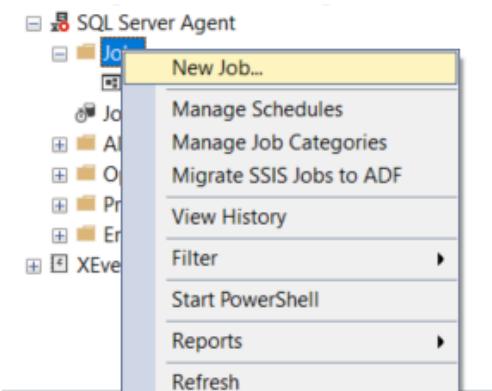
ShelterDB=> SELECT * FROM "shelter".employedogmanagementview;
 dogname | breed | estimatedbirthyear | estimatedbirthmonth | dogstatus | dogweight | sheltername | dogdescription | shelteraddress
-----+-----+-----+-----+-----+-----+-----+-----+-----+
 Earlie | German Shepherd | 2024 | Microchipping | 6 | Available | Paws & Haven Rescue Center | Lillian, Kremienki, 15-333
 Vassily | Beagle | 2024 | Microchipping | 7 | Adopted | Paws & Haven Rescue Center | Lillian, Kremienki, 15-333
 Buddy | Golden Retriever | 2022 | Microchipping | 7 | Available | Furry Friends Sanctuary | Meadow Ridge, Marseille, 13-123
 Boone | German Shepherd | 2022 | 6 | Available | 8 | Affectionate and gentle, perfect for families and active owne
(4 wierszy)
```

### Zrzuty ekranu nr 48: testy uprawnień dla shelteremployee

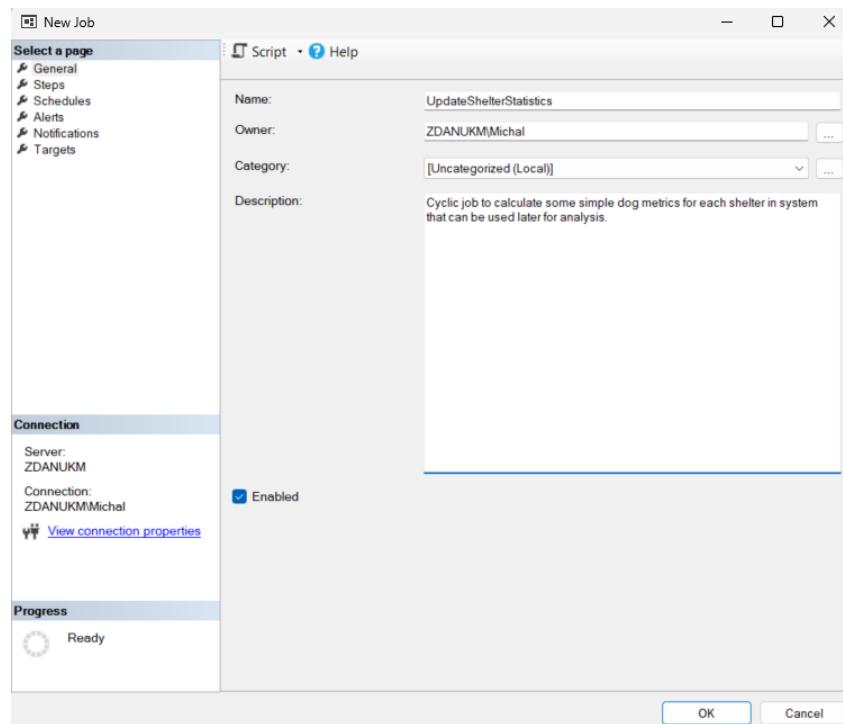
Polecenia SQL, którymi utworzyłem użytkowników i nadałem im uprawnienia znajdują się w pliku: *Define\_users\_with\_privileges\_postgres.sql*.

### Automatyzacja:

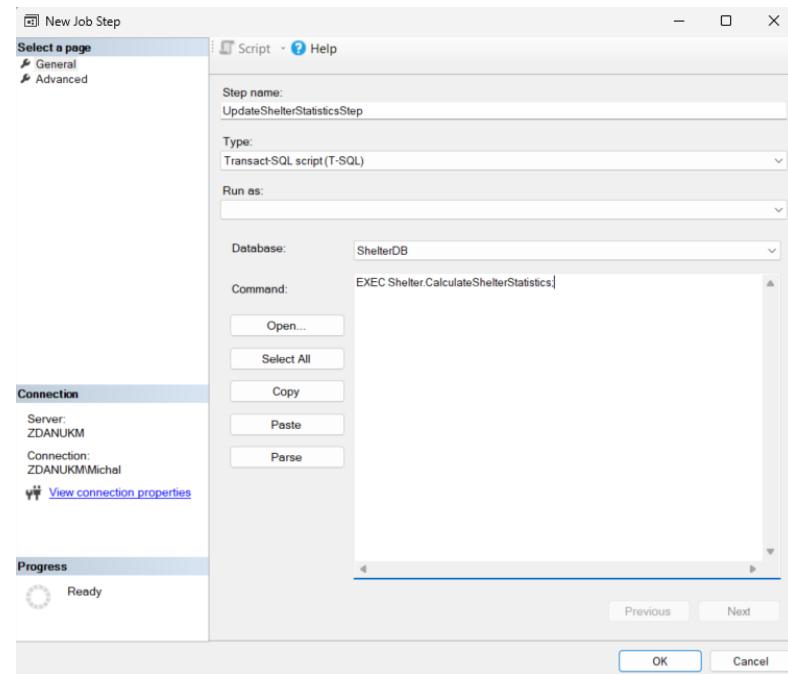
Do automatyzacji przygotowałem nową tabelę *Shelter.ShelterStatistics*. Z punktu widzenia kierownika placówki chciałbym uzyskiwać regularne metryki/statystyki na bazie, których jestem w stanie analizować jak prosperuje moje schronisko. Jako takie bardzo proste metryki może nam służyć liczenie, ile psów mamy w schronisku, ile ma status ‘Zarejestrowany’/’Dostępne’/’GotowyDoAdopcji’/’Adoptowany’/’Uśpiony’, czy liczba psów, które zostały poddane zabiegowi microchipping’u. Na bazie takich dziennych statystyk możemy wykresować krzywe i analizować trendy jak działa nasza palcówka. Ręczne codzienne wywoływanie zapytań i liczenie statystyk byłoby uciążliwe i jest doskonałym zadaniem do automatyzacji przez job'a, który mógłby cyklicznie raz dziennie w godzinach nocnych uruchamiać procedurę, która policzy statystyki z danego dnia i zapisze rekordy dla każdego schroniska. Skrypt z tworzeniem tabeli do statystyk i procedurą liczącą te proste metryki znajduje się w pliku: *Automation.sql*. Poniżej zaprezentuję kolejne kroki i efekty automatyzacji:



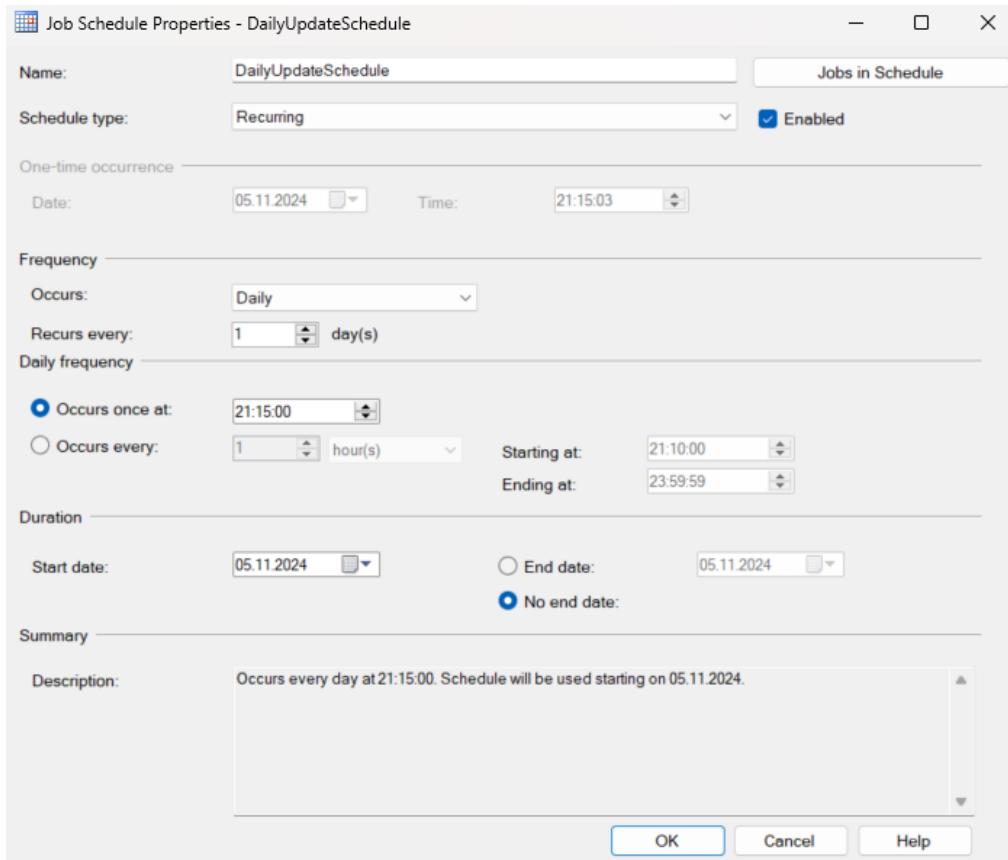
### Zrzuty ekranu nr 49: dodawanie job'a



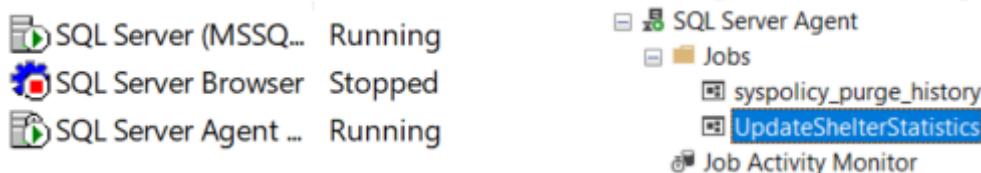
**Zrzuty ekranu nr 50:** definiowanie nazwy i opisu job'a



**Zrzuty ekranu nr 51:** definiowanie kroków job'a (w moim przypadku jeden krok – uruchomienie procedury liczącej statystyki schronisk)



**Zrzuty ekranu nr 52:** definiowanie harmonogramu dla job'a – cyklicznie, raz dziennie (do testów ustawiona godzina 21, w rzeczywistości chcielibyśmy ustawić to w środku nocy, aby był jak najmniej inwazyjny)



**Zrzuty ekranu nr 53:** włączenie SQL Agent'a oraz widok dodanego job'a

Name	Enabled	Status	Last Run...	Last Run	Next Run	Category	Runnable	Schedul...	Category ID
syspolicy_purge...	yes	Idle	Unknown	never	06.11.2024 02:00:00	[Uncate...	yes	yes	0
UpdateShelterS...	yes	Idle	Succee...	05.11.2024 21:24:00	06.11.2024 21:24:00	[Uncate...	yes	yes	0

**Zrzuty ekranu nr 54:** monitor job'ów (job uruchomiony z sukcesem)

```

SELECT TOP (1000) [Id]
      ,[ShelterId]
      ,[MetricsDate]
      ,[TotalDogs]
      ,[RegisteredDogsCount]
      ,[AvailableDogsCount]
      ,[AcceptedToAdoptionCount]
      ,[AdoptedDogsCount]
      ,[EuthanizedDogsCount]
      ,[MicrochippedDogsCount]
  FROM [ShelterDB].[Shelter].[ShelterStatistics]
  
```

	Id	ShelterId	MetricsDate	TotalDogs	RegisteredDogsCount	AvailableDogsCount	AcceptedToAdoptionCount	AdoptedDogsCount	EuthanizedDogsCount	MicrochippedDogsCount
1	6E54E571-00M8-4...	04408D9D-5A70-4A28-9F2C-3D71AF43B293	2024-11-05	3	1	1	0	1	0	1
2	168F50A9-4C76-4F...	7AE8B17D-5B84-4E8C-9EE4-F6B79AF7B62	2024-11-05	6	1	2	0	1	2	2
3	DA86F087-72B8-46...	20C49F67-764B-4E0D-969F-9E0E04684548A	2024-11-05	5	0	3	0	2	0	2
4	A465B53F-1539-42...	C8EACDD5-23B5-4D87-A681-8F197121B37C	2024-11-05	1	0	1	0	0	0	1

Zrzuty ekranu nr 55: poprawnie zapisane metryki po wywołaniu job'a

## Wnioski:

- W tym projekcie poznałem trigger'y, z którymi nie miałem wcześniej przyjemności obcować. Są świetnym mechanizmem do trzymania spójności danych w bazie i implementacji zasad logiki biznesowej.
- Procedury to również świetny element programowalny, standard TSQL pozwala na proste pisanie relatywnie zaawansowanych procedur, które zawierają solidną walidację i dbają o spójne atomowe wykonywanie działań (z użyciem transakcji).
- Wykonywanie kopii zapasowej (przynajmniej w MSSQL) to prosta procedura, jednak bardzo ważna – skoro jest to tali szybki proces to warto byśmy pamiętały o regularnym wykonywaniu back-up'ów, bo koszt jest niewielki, a daje nam to bezpieczeństwo.
- Wykonywanie migracji między dwoma SZDB to bardzo, bardzo trudne zadanie. Na tej części straciłem dużo czasu (naprawdę wiele godzin 😬). Przy tym procesie należy dokładnie skonfigurować środowiska. Dostępne narzędzia często nie gwarantują nam pełnej migracji, ale niewątpliwie są warte użycia, ponieważ mogą znacząco skrócić czas całej migracji – nie musimy wówczas manualnie przerabiać wszystkich skryptów, dostosowywać ich do danego dialekta, a następnie eksportować i importować danych. Przez to, że taki proces jest bardzo żmudny, łatwo popełnić błąd – to kolejna zaleta narzędzi ETL (pomagają nam uniknąć takich błędów).
- Pomimo przepalenia wielu godzin na migracji między systemami uważam to za wartościowe doświadczenie, z którego wyciągnąłem kilka wniosków. Choć jest to skrajnie rzadka sytuacja to warto chociaż raz przejść przez ten proces.
- Znajomość jednego SZDB wcale nie znaczy, że jesteśmy ekspertami we wszystkich bazach relacyjnych. Zdecydowanie systemy różnią się między sobą, a dialekty wcale nie są aż tak zbliżone jak mi się wydawało. Szczególnie różne są procedury i trigger'y.
- Job'y to bardzo solidny mechanizm do automatyzacji prac, które powinny być wykonywane cyklicznie. Myszę, że mogą one nieść dużą wartość biznesową dla systemu, gdzie potrzebujemy regularnie aktualizować statusy i udostępniać bieżące statystyki albo tak jak w moim prostym przykładzie doskonale się sprawdzą, gdy chcemy wyliczać statystyki. Konfiguracja ich jest prosta i całkiem obszerna – możemy tworzyć całkiem zaawansowane zadani.