Projekt Zaliczeniowy
Komputerowy System Wspomagania Wynajmu Krótkoterminowego

Krystian Duma - Grupa Z501 - Nr. Albumu 7763

Grudzień 2018

Spis treści

1 Krótki opis słowny projektu

Projekt zawiera założenia do bazy danych przechowujacej podstawowe informacje o wybranych funkcjach systemu informatycznego wspierajacego funkcjonowanie agencji wynajmu krótkoterminowego domów, mieszkań lub innych obiektów.

2 Założenia do projektu

Przyjęte zostały następujące założenia do projektu

1. Podstawowe Obiekty
   - `Obiekt` - obiekt najmu - np. konkretny dom lub mieszkanie,
   - `Użytkownik` - osoba wynajmująca mieszkanie lub dom,
2. Przechowywane zadania (transakcje)
   - `Najem` - transakcja zwiazana z wynajęciem `Obiektu` przez `Użytkownika`.
3. Szczegoły opisu
   - `Użytkownik` - potrzeba przechowania informacji: nazwisko klienta, imie klienta, wiek klienta, adres zamieszkania klienta, telefon klienta, płec klienta oraz login używany do logowania do bazy danych.
   - `Obiekt` - potrzeba przechowania informacji: nazwa własna obiektu, adres obiektu, dzienna stawka najmu obiektu, kategoria obiektu, obecny status najmu obiektu (informacja czy dany obiekt jest obecnie wolny lub zajęty), opis obiektu oraz inne atrybuty odpowiednie dla zgromadzonych obiektów.
     - Każdy obiekt może znajdować się w wielu różnych kategoriach,
     - Dla uproszczenia inne atrybuty będą znajdować się w opisie danego obiektu.
   - `Najem` - potrzeba przechowania informacji: użytkownika-najemca, wynajmowany obiekt, data rozpoczęcia najmu, data zakończenia najmu, koszt najmu.
     - Najem to transakcja tylko jednego `Użytkownika` i tylko jednego `Obiektu`,
     - Dla uproszczenia najem jest liczony od godziny 00:00 do godziny 23:59,
     - Jeden `Obiekt` może być w danym czasie wynajęty tylko jednemu użytkownikowi.
4. Użytkownicy i Uprawnienia
   - Administrator ma dostęp do danych wszystkich użytkowników,
   - Każdy `Użytkownik` ma założone oddzielne konto serwera SQL,
   - Użytkownicy nie widzą danych oraz wypożyczeń innych użytkowników.

3 Środowisko Projektowe

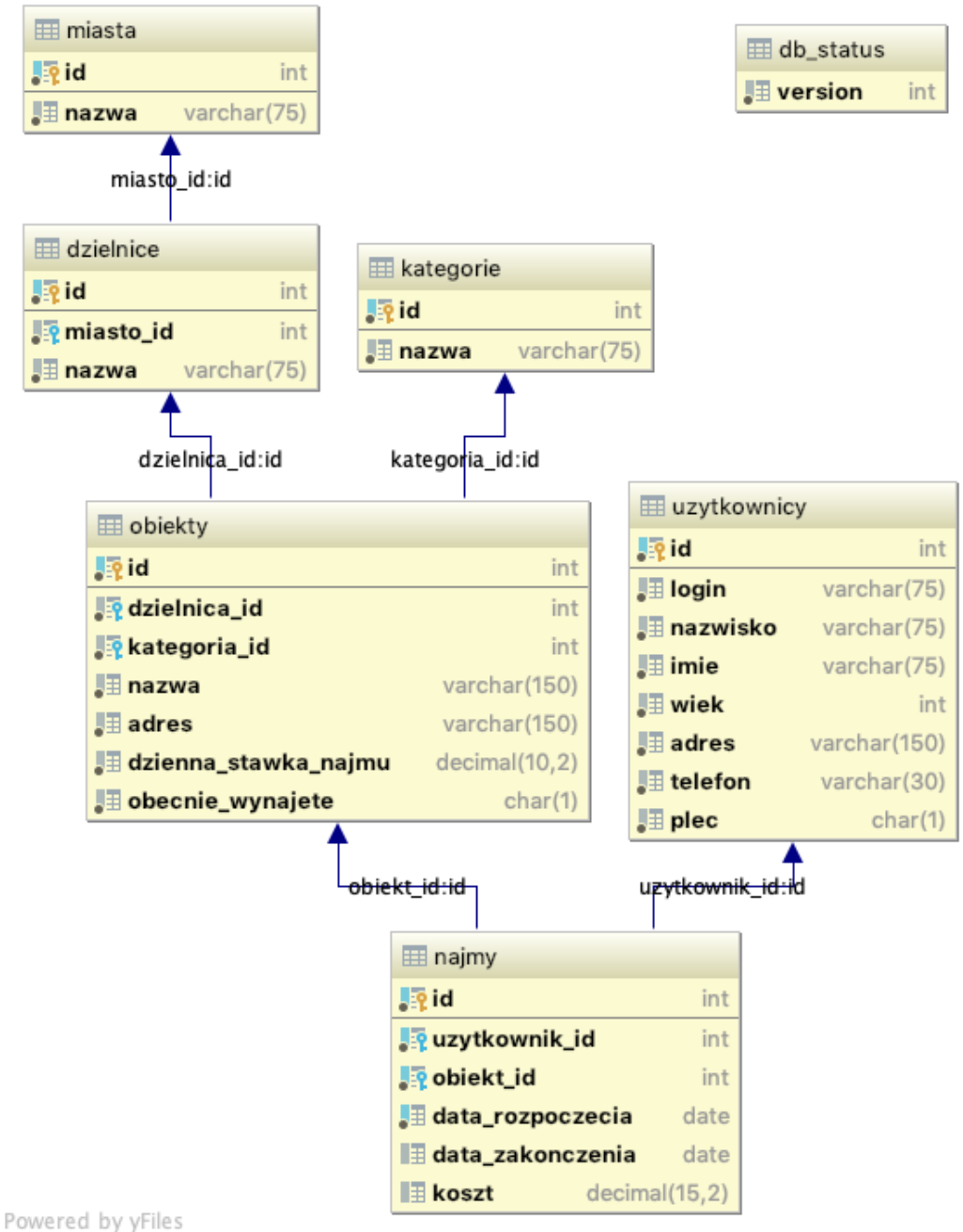
Środowiskiem uruchomieniowym jest baza danych Microsoft SQL Server 2017 uruchomiona w kontenerze Docker'a. Jako obraz bazowy został wybrany obraz mcr.microsoft.com/mssql/server:2017-latest-ubuntu który zawiera najaktualniejszą obecnie wersję Microsoft SQL Server 2017 uruchomioną na systemie Linux - Ubuntu Server. Do obrazu zostały doinstalowane dodatkowe narzędzia umożliwiające przygotowanie plików wyjściowych: tego dokumentu pdf (LaTeX) oraz skryptów tworzących i usuwających obiekty z bazy (PHP). Dodatkowo na serwerze została skonfigurowana opcja `contained database authentication` dzięki której możliwe jest tworzeniu i autoryzacja użytkowników w bazie SQL.
Jako aplikację służącą do łączenia się i wykonywania poleceń wykorzystane zostały aplikacje:

- Dołączona do SQL Server'a aplikacja wiersza poleceń - sqlcmd
- Środowisko IDE od czeskiej firmy JetBrains - DataGrip
- Środowisko IDE od Microsoft'u - SQL Server Management Studio (SSMS)

4 Model fizyczny bazy danych

Na Rysunku 1 znajduje się schemat (diagram tabel) wygenerowanej przez skrypt: skrypt_tworzacy_obiekty_w_bazie_danych.sql.

Rysunek 1: Diagram tabel wygenerowanej bazy danych



5 Skrypt tworzący obiekty w bazie danych

5.1 Model wersjonowania bazy danych

Jak można zauważyć na Rysunku 1, w bazie danych znajduje się jedna dodatkowa tabela `db_status` z jednym polem `version` - służy ona do przechowywania wersji bazy danych. Każda operacja w skrypcie tworzącym sprawdza i porównuje obecną oraz oczekiwaną wersję dla danej operacji. Dzięki temu zabiegowi nie będzie można uruchomić danej operacji dla jednej bazy danych wielokrotnie. Dodatkowo aktualizacja istniejącej bazy danych do najnowszej wersji będzie uproszczona - wystarczy uruchomić najnowszą wersję skryptu, a wykonane zostaną tylko nowe operacje dodane od ostatniego uruchomienia skryptu instalacyjnego. Każda operacja jest opakowana zgodnie z szablonem z listingu 1.

```
PRINT 'Wersja X: ''<<< OPIS OPERACJI >>>'''
IF EXISTS(SELECT * FROM sys.tables WHERE name = N'db_status')
  BEGIN
    IF EXISTS(SELECT * FROM db_status WHERE version = X)
      BEGIN

    <<< MIEJSCE NA KOD >>>

        UPDATE db_status SET version = 1 WHERE version = X;
        PRINT 'Wersja X: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji X';
      END
    ELSE
      BEGIN
        IF EXISTS(SELECT * FROM db_status WHERE version < X)
          BEGIN
            RAISERROR ('Wersja X: Baza danych jest w za niskiej wersji (wymagana jest wersja X) aby zainstalowac migracje', 11, 2);
          END
        ELSE
          BEGIN
            PRINT 'Wersja X: Migracja ju  zostala zainstalowana wczesniej';
          END
      END
  END
ELSE
  BEGIN
    RAISERROR ('Wersja X: Nie znaleziono tabeli wersjonowania bazy danych', 11, 1);
  END
```

Listing 1: Szablon kodu wersjonowanego

Dodatkowo w przypadku wystąpienia jakichkolwiek błędów jest przewidziana procedura ich łapania - na listingu 2 widzimy zawartość bloku `CATCH` skryptu instalacyjnego. Skrypt został przygotowany w taki sposób aby w przypadku wystąpienia błędu przerywał działanie[1] i przechodził od razu do bloku `CATCH`.

```
BEGIN CATCH

  SELECT
    ERROR_NUMBER() AS ErrorNumber,
    ERROR_SEVERITY() AS ErrorSeverity,
    ERROR_STATE() AS ErrorState,
    ERROR_PROCEDURE() AS ErrorProcedure,
    ERROR_LINE() AS ErrorLine,
    ERROR_MESSAGE() AS ErrorMessage;

END CATCH;
```

Listing 2: Blok CATCH w skrypcie tworzącym

5.2 Wynik uruchomienia całego skryptu tworzącego obiekty w trybie wsadowym

Jak widać na listingu 3, skrypt podaje bardzo dokładne informacje na temat aktualnie wykonywanej operacji. W większości przypadków wystąpienia ciągu tekstowego `(1 rows affected )`, następuje zmiana aktualnej wersji bazy danych w tabeli wersjonowania - `db_status`.

[1] Aby wywołanie funkcji `RAISERROR` przekazało kontrolę do bloku `CATCH`, parametr `severity` musi mieć wartość z zakresu od `11` do `19`. Wartości poniżej nie powodują przerwania skryptu, a wartości powyżej terminują połączenie z bazą danych.

```
1
2  (1 rows affected)
3  Tabela wersjonowania zostala utworzona
4  Wersja 1: 'Utworzenie tabeli z miastami'
5
6  (1 rows affected)
7  Wersja 1: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 1
8  Wersja 2: 'Utworzenie tabeli z dzielnicami'
9
10 (1 rows affected)
11 Wersja 2: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 2
12 Wersja 3: 'Utworzenie relacji pomiedzy miastami a dzielnicami'
13
14 (1 rows affected)
15 Wersja 3: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 3
16 Wersja 4: 'Utworzenie tabeli z kategoriami'
17
18 (1 rows affected)
19 Wersja 4: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 4
20 Wersja 5: 'Utworzenie tabeli z obiektami'
21
22 (1 rows affected)
23 Wersja 5: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 5
24 Wersja 6: 'Utworzenie relacji pomiedzy dzielnicami a obiektami'
25
26 (1 rows affected)
27 Wersja 6: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 6
28 Wersja 7: 'Utworzenie relacji pomiedzy kategoriami a obiektami'
29
30 (1 rows affected)
31 Wersja 7: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 7
32 Wersja 8: 'Utworzenie tabeli z uzytkownikami'
33
34 (1 rows affected)
35 Wersja 8: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 8
36 Wersja 9: 'Utworzenie tabeli z najmami'
37
38 (1 rows affected)
39 Wersja 9: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 9
40 Wersja 10: 'Utworzenie indeksu unikatowego w tabeli z najemcami'
41
42 (1 rows affected)
43 Wersja 10: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 10
44 Wersja 11: 'Utworzenie relacji pomiedzy uzytkownikami a najmami'
45
46 (1 rows affected)
47 Wersja 11: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 11
48 Wersja 12: 'Utworzenie relacji pomiedzy obiektami a najmami'
49
50 (1 rows affected)
51 Wersja 12: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 12
52 Wersja 13: 'Utworzenie indeksu unikatowego w tabeli z uzytkownikami'
53
54 (1 rows affected)
55 Wersja 13: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 13
56 Wersja 14: 'Utworzenie widoku z lista wszystkich najmow'
57
58 (1 rows affected)
59 Wersja 14: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 14
60 Wersja 15: 'Utworzenie widoku z lista popularnosci obiektow'
61
62 (1 rows affected)
63 Wersja 15: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 15
64 Wersja 16: 'Utworzenie widoku z lista niewynajmowanych obiektow'
65
66 (1 rows affected)
67 Wersja 16: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 16
68 Wersja 17: 'Utworzenie funkcji wyliczajacej koszt najmu obiektu'
69
70 (1 rows affected)
71 Wersja 17: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 17
72 Wersja 18: 'Utworzenie funkcji wyliczajacej koszt konkretnego najmu'
73
74 (1 rows affected)
```

```
Wersja 18: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 18
Wersja 19: 'Utworzenie triggeru aktualizujacego koszt najmu'

(1 rows affected)
Wersja 19: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 19
Wersja 20: 'Utworzenie funkcji wyswietlajacej adresowke uzytkownika'

(1 rows affected)
Wersja 20: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 20
Wersja 21: 'Utworzenie funkcji wyswietlajacej spozniajacych sie uzytkownikow'

(1 rows affected)
Wersja 21: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 21
Wersja 22: 'Utworzenie roli administratora systemu'

(1 rows affected)
Wersja 22: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 22
Wersja 23: 'Nadanie uprawnien roli administratora systemu'

(1 rows affected)
Wersja 23: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 23
Wersja 24: 'Utworzenie uzytkownika administracyjnego'

(1 rows affected)
Wersja 24: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 24
Wersja 25: 'Przypisanie uzytkownikowi administracyjnemu roli administratora systemu'

(1 rows affected)
Wersja 25: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 25
Wersja 26: 'Utworzenie roli operatora systemu'

(1 rows affected)
Wersja 26: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 26
Wersja 27: 'Nadanie uprawnien roli operatora systemu'

(1 rows affected)
Wersja 27: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 27
Wersja 28: 'Utworzenie uzytkownika operatora'

(1 rows affected)
Wersja 28: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 28
Wersja 29: 'Przypisanie uzytkownikowi operatora roli operatora systemu'

(1 rows affected)
Wersja 29: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 29
Wersja 30: 'Utworzenie roli uzytkownika systemu'

(1 rows affected)
Wersja 30: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 30
Wersja 31: 'Utworzenie procedury skladowanej wynajmowania obiektow'

(1 rows affected)
Wersja 31: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 31
Wersja 32: 'Utworzenie procedury skladowanej do tworzenia uzytkownikow'

(1 rows affected)
Wersja 32: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 32
Wersja 33: 'Nadanie uprawnien roli uzytkownika systemu'

(1 rows affected)
Wersja 33: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 33
Wersja 34: 'Nadanie uprawnien wiersza w tabeli uzytkownicy'

(1 rows affected)
Wersja 34: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 34
Wersja 35: 'Nadanie uprawnien wiersza w tabeli najmy'

(1 rows affected)
Wersja 35: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 35
Wersja 36: 'Utworzenie triggeru aktualizujacego stan obiekty'

(1 rows affected)
Wersja 36: Migracja zostala zainstalowana pomyslnie - teraz baza jest w wersji 36
```

Listing 3: Wynik uruchomienia całego skryptu tworzącego obiekty w trybie wsadowym

5.3 Tabele

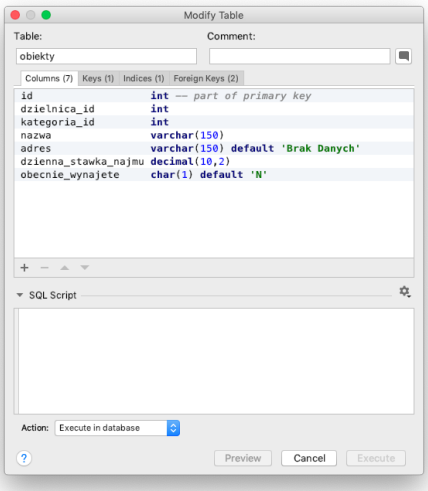
Wszystkie tabele są tworzone przez 13 skryptów `SQL`:

- Utworzenie tabeli z miastami
- Utworzenie tabeli z dzielnicami
- Utworzenie relacji pomiedzy miastami a dzielnicami
- Utworzenie tabeli z kategoriami
- Utworzenie tabeli z obiektami
- Utworzenie relacji pomiedzy dzielnicami a obiektami
- Utworzenie relacji pomiedzy kategoriami a obiektami
- Utworzenie tabeli z uzytkownikami
- Utworzenie tabeli z najmami
- Utworzenie indeksu unikatowego w tabeli z najmami
- Utworzenie relacji pomiedzy uzytkownikami a najmami
- Utworzenie relacji pomiedzy obiektami a najmami
- Utworzenie indeksu unikatowego w tabeli z uzytkownikami

Tworzenie relacji pomiędzy tabelami oraz indeksów zostało oddzielone od operacji tworzenia poszczególnych tabel - celem tego działania jest lepsza organizacja skryptów. Dodatkowo oddzielając te operacje, w przypadku wystąpienia jakiegoś błędu jesteśmy w stanie określić co i gdzie się "wysypało".
Ze względu na to aby nie zajmować zbyt dużo miejsca, poniżej zostaną przedstawione tylko najważniejsze z powyższych skryptów.

5.3.1 Utworzenie tabeli z obiektami

Ponieważ polecenia `CREATE DEFAULT` oraz `CREATE RULE` zostały zdeprecjonowane i w kolejnych wersjach SQL Serwera prawdopodobnie zostaną usunięte zdecydowałem się umieścić wartości domyślne oraz reguły sprawdzające w definicjach konkretnych tabel.
W wyniku projektowania zostało dodatkowo ustalone że `dzienna_stawka_najmu` musi być większa od 0.
Status obiektu znajdujący się w polu `obecnie_wynajete` może przyjmować dwie wartości `T` oraz `N` - odpowiednio dla obiekty wynajętego oraz wolnego.



Rysunek 2: Tabela `obiekty` wyświetlona w programie DataGrip

```sql
CREATE TABLE obiekty (
  id INT PRIMARY KEY NOT NULL IDENTITY (1, 1),

  dzielnica_id INT NOT NULL,
  kategoria_id INT NOT NULL,

  nazwa VARCHAR(150) NOT NULL,
  adres VARCHAR(150) NOT NULL DEFAULT 'Brak Danych',
  dzienna_stawka_najmu DECIMAL(10, 2) NOT NULL CHECK (dzienna_stawka_najmu > 0),

  obecnie_wynajete CHAR(1) NOT NULL DEFAULT 'N' CHECK (obecnie_wynajete IN ('T', 'N')),
);
```

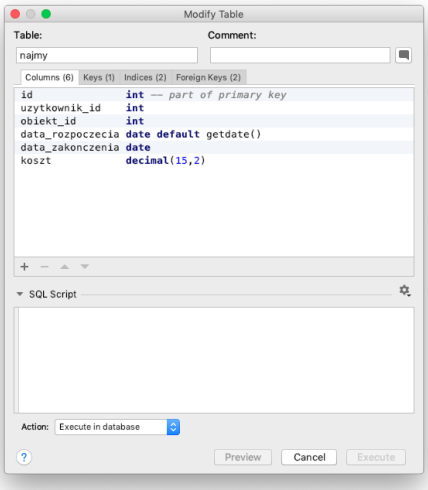
Listing 4: Skrypt tworzący tabelę `obiekty`

5.3.2 Utworzenie tabeli z najmami i utworzenie indeksu unikatowego w tabeli z najmami

Przyjmujemy że domyślną datą rozpoczęcia najmu jest data jego dodania do bazy.

```sql
CREATE TABLE najmy (
  id INT PRIMARY KEY NOT NULL IDENTITY (1, 1),

  uzytkownik_id INT NOT NULL,
  obiekt_id INT NOT NULL,

  data_rozpoczecia DATE NOT NULL DEFAULT getdate(),
  data_zakonczenia DATE NULL,
  koszt DECIMAL(15, 2) NULL,
);
```

Listing 5: Skrypt tworzący tabelę `najmy`



Rysunek 3: Tabela `najmy` wyświetlona w programie DataGrip

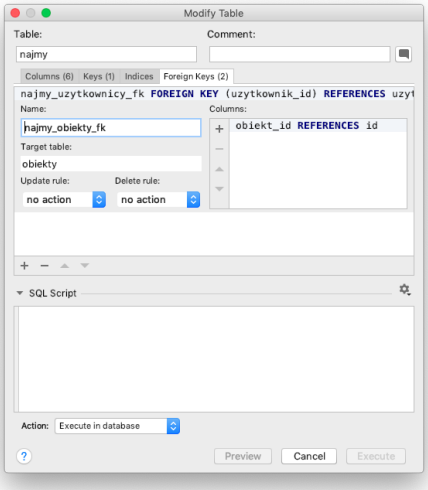
Ponieważ w założeniach projektowych przyjeliśmy że najem następuje od północy do godziny 23:59, to zostało wykorzystane pole typu `DATE`. Więc na przykład jeśli klient wynajmie dany obiekt tylko na jeden dzień to w polach `data_rozpoczecia` oraz `data_zakonczenia` wartość będzie ta sama. Dzięki temu możemy również utworzyć indeks unikatowy na pola `uzytkownik_id`, `obiekt_id` oraz `data_rozpoczecia`.

```sql
CREATE UNIQUE INDEX najemcy_ui
   ON najmy (uzytkownik_id, obiekt_id, data_rozpoczecia);
```

Listing 6: Skrypt tworzący indeks unikatowy w tabeli `obiekty`

5.3.3 Utworzenie relacji pomiędzy użytkownikami a najmami

Wszystkie tabele zostały połączone relacją (CONSTRAINT) typu klucz obcy (FOREIGN KEY) tak jak na przykładzie z listingu 7.



Rysunek 4: Relacja `najmy_obiekty_fk` wyświetlona w programie DataGrip

```
1 ALTER TABLE najmy
2   ADD CONSTRAINT najmy_obiekty_fk
3   FOREIGN KEY (obiekt_id)
4   REFERENCES obiekty(id);
```

Listing 7: Skrypt tworzący relację `najmy_obiekty_fk`

5.4 Widoki

Zostało stworzone kilka widoków:

- `lista_najmow` - Lista wszystkich najmów
- `lista_popularnosci_obiektow` - Lista obiektów wraz z ilością (popularnoscią) ich najmów
- `lista_niepopularnych_obiektow` - Lista obiektów które nie zostały nigdy wynajęte

5.4.1 `lista_najmow` - Lista wszystkich najmów

Widok ten zwraca listę wszystkich najmów, wraz z następującymi polami:

- nazwisko
- imie
- datę rozpoczecia najmu
- datę zakonczenia najmu
- nazwę wynajmowanego obiektu
- całkowity koszt najmu

	nazwisko	imie	data_rozpoczecia	data_zakonczenia	nazwa_obiektu	calkowity_koszt
22	Nazwisko 2	Imie 1	2018-12-11	2018-12-21	Nazwa 3	704.55
23	Nazwisko 3	Imie 3	2018-11-28	2018-12-25	Nazwa 4	1793.40
24	Nazwisko 2	Imie 4	2018-12-06	2018-12-23	Nazwa 3	2160.00
25	Nazwisko 3	Imie 3	2018-12-12	2018-12-21	Nazwa 4	1200.00
26	Nazwisko 2	Imie 2	2018-12-11	2018-12-25	Nazwa 4	1800.00
27	Nazwisko 2	Imie 4	2018-11-30	2018-12-24	Nazwa 3	633.75
28	Nazwisko 2	Imie 1	2018-12-04	2018-12-22	Nazwa 2	2280.00
29	Nazwisko 2	Imie 1	2018-12-03	2018-12-25	Nazwa 3	1473.15
30	Nazwisko 3	Imie 3	2018-12-08	2018-12-22	Nazwa 1	380.25
31	Nazwisko 2	Imie 4	2018-12-08	2018-12-19	Nazwa 3	768.60
32	Nazwisko 3	Imie 3	2018-12-04	2018-12-24	Nazwa 2	1345.05
33	Nazwisko 2	Imie 1	2018-12-25	<null>	Nazwa 2	<null>
34	Nazwisko 3	Imie 3	2018-12-23	<null>	Nazwa 1	<null>
35	Nazwisko 2	Imie 4	2018-12-25	<null>	Nazwa 4	<null>
36	Nazwisko 2	Imie 2	2018-12-24	<null>	Nazwa 2	<null>
37	Nazwisko 2	Imie 4	2018-12-23	<null>	Nazwa 2	<null>
38	Nazwisko 3	Imie 3	2018-12-26	<null>	Nazwa 3	<null>
39	Nazwisko 2	Imie 4	2018-12-27	<null>	Nazwa 2	<null>
40	Nazwisko 2	Imie 1	2018-12-23	<null>	Nazwa 1	<null>

Rysunek 5: Wyświetlony widok `lista_najmow`

```
CREATE VIEW lista_najmow
  AS
    SELECT nazwisko, imie, data_rozpoczecia, data_zakonczenia, nazwa nazwa_obiektu, koszt
    calkowity_koszt
        FROM najmy n
        JOIN uzytkownicy u on n.uzytkownik_id = u.id
        JOIN obiekty o on n.obiekt_id = o.id;
```

Listing 8: Skrypt tworzący widok `lista_najmow`

5.4.2 `lista_niepopularnych_obiektow` - Lista obiektów które nie zostały nigdy wynajęte

Widok ten zwraca listę obiektów które nie zostały nigdy, prze nikogo, wynajęte, wraz z następującymi polami:

- nazwę obiektu
- adres obiektu

Widok ten korzysta z innego (nieopisanego w tym dokumencie) widoku - `lista_popularnosci_obiektow`.

```
CREATE VIEW lista_niepopularnych_obiektow
  AS
    SELECT id, nazwa, adres
      FROM lista_popularnosci_obiektow
      GROUP BY id, nazwa, adres
      HAVING SUM(liczba_najmow) = 0;
```

Listing 9: Skrypt tworzący widok `lista_niepopularnych_obiektow`

	id	nazwa	adres
1	4	Nazwa 4	Adres 2
2	6	Nazwa 1	Adres 3
3	7	Nazwa 3	Adres 2
4	9	Nazwa 4	Adres 2
5	10	Nazwa 4	Adres 1
6	14	Nazwa 4	Adres 3
7	19	Nazwa 1	Adres 4
8	24	Nazwa 2	Adres 4
9	25	Nazwa 2	Adres 4
10	28	Nazwa 1	Adres 4
11	32	Nazwa 2	Adres 1
12	38	Nazwa 3	Adres 4
13	40	Nazwa 4	Adres 3
14	43	Nazwa 2	Adres 4
15	47	Nazwa 1	Adres 1
16	51	Nazwa 4	Adres 2
17	53	Nazwa 4	Adres 4
18	56	Nazwa 2	Adres 4
19	58	Nazwa 4	Adres 1
20	59	Nazwa 3	Adres 2
21	60	Nazwa 2	Adres 3
22	62	Nazwa 4	Adres 3

Rysunek 6: Wyświetlony widok `lista_niepopularnych_obiektow`

5.5 Funkcje Skalarne i Tabelarne

Zostały stworzone następujące funkcje:

- `koszt_najmu_obiektu` - Funkcja obliczająca koszt najmu wskazanego obiektu
- `koszt_najmu` - Funkcja obliczająca koszt konkretnego najmu
- `adresowka` - Funkcja generująca etykietę adresową dla użytkownika
- `opoznieni` - Funkcja generująca etykiety dla użytkowników którzy coś wynajęli ale jeszcze nie oddali

5.5.1 `koszt_najmu_obiektu` - Funkcja obliczająca koszt najmu wskazanego obiektu

`koszt_najmu_obiektu (@obiekt_id INT, @liczba_dni INT)` - Funkcja ta oblicza koszt najmu obiektu wskazanego w parametrze `@obiekt_id` przez liczbę dni określoną w parametrze `@liczba_dni`, zwracana wartość ma typ `DECIMAL(15, 2)` który jest kompatybilny z typem kolumny `koszt` w tabeli `najmy`. Funkcja ta została wydzielona w celu uniknięcia duplikacji kodu w funkcji skalarnej `koszt_najmu` z listingu 11 oraz w funkcji tabelarnej `opoznieni` z listingu 13 (strona 14).

```
CREATE FUNCTION koszt_najmu_obiektu (@obiekt_id INT, @liczba_dni INT)
  RETURNS DECIMAL(15, 2)
  AS
  BEGIN
    DECLARE @dzienna_stawka DECIMAL(10,2);

    SELECT @dzienna_stawka = dzienna_stawka_najmu
      FROM obiekty o
      WHERE id = @obiekt_id;

    RETURN @liczba_dni*@dzienna_stawka;
  END
```

Listing 10: Skrypt tworzący funkcję skalarną `koszt_najmu_obiektu`

5.5.2 `koszt_najmu` - Funkcja obliczająca koszt danego najmu

`koszt_najmu (@najem_id INT)` - Funkcja ta oblicza koszt konkretnego najmu wskazanego w parametrze `@najem_id`, zwracana wartość ma taki sam typ jak funkcja `koszt_najmu_obiektu` która jest wywoływana - czyli `DECIMAL(15, 2)` który jest kompatybilny z typem kolumny `koszt` w tabeli `najmy`. Przykład wykorzystania tej funkcji jest w triggerze z listingu 14 (strona 15).

```sql
CREATE FUNCTION koszt_najmu (@najem_id INT)
  RETURNS DECIMAL(15, 2)
  AS
  BEGIN
    DECLARE @koszt DECIMAL(15, 2);
    DECLARE @liczba_dni INT;
    DECLARE @obiekt_id INT;
    DECLARE @data_rozpoczecia DATE;
    DECLARE @data_zakonczenia DATE;

    SELECT @data_rozpoczecia = n.data_rozpoczecia,
           @data_zakonczenia = n.data_zakonczenia,
           @obiekt_id = o.id
      FROM najmy n
      JOIN obiekty o on n.obiekt_id = o.id
      WHERE n.id = @najem_id;

    IF @data_zakonczenia IS NULL
      RETURN NULL;

    SELECT @liczba_dni = (1+DATEDIFF(DAY, @data_rozpoczecia, @data_zakonczenia));

    SELECT @koszt = dbo.koszt_najmu_obiektu(@obiekt_id, @liczba_dni);

    RETURN @koszt;
  END
```

Listing 11: Skrypt tworzący funkcję skalarną `koszt_najmu`

5.5.3 `adresowka` - Funkcja generująca etykietę adresową dla użytkownika

`adresowka (@uzytkownik_id INT)` - Funkcja ta generuje zawartość etykiety adresowej dla użytkownika wskazanego w parametrze `@uzytkownik_id`, zwracana wartość ma typ VARCHAR(333)[2]. Przykład wykorzystania tej funkcji jest w funkcji tabelarnej z listingu 13.

```sql
CREATE FUNCTION adresowka (@uzytkownik_id INT)
  RETURNS VARCHAR(333) -- 75+75+150+30+3 = 333 - suma d ugo ci     czonych     p l
  AS
  BEGIN
    DECLARE @adresowka VARCHAR(330);

    SELECT @adresowka = nazwisko + ' ' + imie + CHAR(13) + telefon + CHAR(13) + adres
      FROM uzytkownicy
      WHERE id = @uzytkownik_id;

    RETURN @adresowka;
  END
```

Listing 12: Skrypt tworzący funkcję skalarną `adresowka`

5.5.4 `opoznieni` - Funkcja generująca etykiety dla użytkowników którzy coś wynajęli ale jeszcze nie oddali

`opoznieni (@liczba_dni INT)` - Funkcja ta generuje listę obiektów wraz z najemcami, które zostały wynajęte co najmniej liczbę dni wcześniej określoną parametrem `@liczba_dni`. Funkcja ta zwraca tabelę składającą się z identyfikatora najmu, nazwy wynajętego obiektu, liczby dni od rozpoczęcia najmu, szacunkowego kosztu tego najmu na dzień bieżący oraz etykiety adresowej do najemcy.

	najem_id	nazwa_obiektu	dni_od_wynajmu	szacunkowy_koszt_najmu	etykieta
1	33	Nazwa 1	3	599.96	Nazwisko 4 Imie 3␤+48 987 543 123␤Adres 4
2	36	Nazwa 1	4	600.00	Nazwisko 3 Imie 1␤+48 234 567 890␤Adres 2
3	37	Nazwa 4	3	480.00	Nazwisko 4 Imie 2␤+48 987 543 123␤Adres 2
4	38	Nazwa 3	4	320.25	Nazwisko 3 Imie 1␤+48 234 567 890␤Adres 2
5	48	Nazwa 1	4	320.25	Nazwisko 4 Imie 3␤+48 987 543 123␤Adres 4

Rysunek 7: Uruchomiona funkcja `opoznieni` z parametrem `@liczba_dni` równym 3

[2]Rozmiar pola bierze się z sumy długości użytych pól (`nazwisko` i `imie` po 75 znaków, `telefon` 30 znaków,`adres` 150 znaków) oraz znaków dodanych (jedna spacja i dwa znaki nowej lini - `CHAR(13)`) - jest to najdłuższy możliwy wynik tej funkcji.

```sql
CREATE FUNCTION opoznieni (@liczba_dni INT)
  RETURNS TABLE
  AS
    RETURN (
      SELECT n.id najem_id,
             o.nazwa nazwa_obiektu,
             DATEDIFF(DAY, n.data_rozpoczecia, getdate()) dni_od_wynajmu,
             dbo.koszt_najmu_obiektu(o.id, DATEDIFF(DAY, n.data_rozpoczecia, getdate()) + 1)
    szacunkowy_koszt_najmu,
             dbo.adresowka(u.id) etykieta
        FROM najmy n
        JOIN obiekty o on n.obiekt_id = o.id
        JOIN uzytkownicy u on n.uzytkownik_id = u.id
        WHERE DATEDIFF(DAY, n.data_rozpoczecia, getdate()) >= @liczba_dni
              AND n.data_zakonczenia IS NULL
    )
```

Listing 13: Skrypt tworzący funkcję tabelarną `opoznieni`

5.6 Triggery

Zostały stworzone następujące triggery:

- `wylicz_koszt_najmu` - Trigger zapisujący koszt najmu
- `aktualizuj_stan_obiektu` - Trigger zapisujący koszt najmu

5.6.1 `wylicz_koszt_najmu` - Trigger zapisujący koszt najmu

Trigger ten jest uruchamiany w momencie dodania nowego wiersza do tabeli `najmu` lub aktualizacji już istniejącego. Klauzulą WHERE ograniczamy obliczenia tylko do wierszy nowych lub wierszy gdzie kolumna `data_zakonczenia` została zaktualizowana. Dzięki wykorzystaniu wcześniej przygotowanej funkcji skalarnej `koszt_najmu` (listing 11), kod tego triggera jest stosunkowo prosty i przejrzysty. Trigger został zaprojektowany w taki sposób aby możliwe było wykonywanie zbiorowych operacji - dzięki temu możemy dodawać/aktualizować wiele wierszy a i tak trigger będzie działać prawidłowo. Przykładowy wynik działania widzimy na rysunku 5 ze strony 11 - po dodaniu losowych danych testowych, koszty najmów zostały automatycznie obliczone.

```
CREATE TRIGGER wylicz_koszt_najmu
  ON najmy
  AFTER INSERT, UPDATE
  AS
  BEGIN
    UPDATE najmy
      SET koszt = dbo.koszt_najmu(n.id)
      FROM najmy n
      JOIN inserted i ON n.id = i.id
      WHERE UPDATE (data_zakonczenia) OR NOT EXISTS(SELECT 1 FROM DELETED)
  END
```

Listing 14: Skrypt tworzący trigger `wylicz_koszt_najmu`

5.6.2 `aktualizuj_stan_obiektu` - Trigger zapisujący koszt najmu

Kolejny przygotowany trigger dba o aktualizację stanu obiektu (`obecnie_wynajete`) w tabeli `obiekty`.

```
CREATE TRIGGER aktualizuj_stan_obiektu
  ON najmy
  AFTER INSERT, UPDATE
  AS
  BEGIN
    UPDATE obiekty
      SET obecnie_wynajete = IIF(i.data_zakonczenia IS NULL, 'T', 'N')
      FROM obiekty o
      JOIN najmy n ON o.id = n.obiekt_id
      JOIN inserted i ON n.id = i.id
  END
```

Listing 15: Skrypt tworzący trigger `aktualizuj_stan_obiektu`

5.7 Procedury Składowane

Zostały stworzone następujące procedury:

- `utworz_uzytkownika` - Procedura do tworzenia użytkowników
- `wynajmij_obiekt` - Procedura do wynajmowania obiektów przez użytkowników

5.7.1 `utworz_uzytkownika` - Procedura do tworzenia użytkowników

Procedura ta służy do tworzenia użytkowników bazy SQL Server i dodawania ich do tabeli `uzytkownicy`. Procedura przyjmuje 8 parametrów:

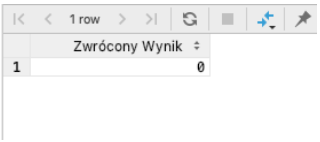
- `@login VARCHAR(75)`
- `@nazwisko VARCHAR(75)`
- `@imie VARCHAR(75)`
- `@wiek INT`
- `@adres VARCHAR(150)`
- `@telefon VARCHAR(30)`
- `@plec CHAR(1)`
- `@haslo VARCHAR(30)`

Procedura zwraca następujące wartości:

- 0 - Użytkownik dodany pomyślnie
- 1 - Wystąpił nieznany błąd
- 2 - Taki użytkownik już istnieje

```
CREATE PROCEDURE utworz_uzytkownika
  @login VARCHAR(75),
  @nazwisko VARCHAR(75),
  @imie VARCHAR(75),
  @wiek INT,
  @adres VARCHAR(150),
  @telefon VARCHAR(30),
  @plec CHAR(1),
  @haslo VARCHAR(30)
  AS
    IF EXISTS(SELECT * FROM uzytkownicy WHERE login = @login)
      RETURN 2;

    EXEC sp_addlogin @login, @haslo;
    EXEC sp_adduser @login;
    EXEC sp_addrolemember 'uzytkownicy_systemu', @login;

    BEGIN TRANSACTION;

    INSERT INTO uzytkownicy (login, nazwisko, imie, wiek, adres, telefon, plec)
      VALUES (@login, @nazwisko, @imie, @wiek, @adres, @telefon, @plec);

    IF @@ERROR<>0
      GOTO BLAD;

    COMMIT TRANSACTION
    RETURN 0;

    BLAD:
      ROLLBACK TRANSACTION
      RETURN 1;
```

Listing 16: Skrypt tworzący procedurę składowaną `utworz_uzytkownika`



Rysunek 8: Wynik prawidłowego uruchomienia przykładu użycia procedury `utworz_uzytkownika`

```
DECLARE @wynik INT;
EXEC @wynik = utworz_uzytkownika
   @login = 'j_kowalski',
   @nazwisko = 'Jan',
   @imie = 'Kowalski',
   @wiek = 25,
   @adres = 'Ul. Kowalska 23, 02-001, Warszawa',
   @telefon = '+48 625 548 874',
   @plec = 'M',
   @haslo = 'yourStrong(!)Password';
SELECT 'Zwr cony Wynik' = @wynik;
```

Listing 17: Przykład użycia procedury `utworz_uzytkownika`

5.7.2 `wynajmij_obiekt` - Procedura do wynajmowania obiektów przez użytkowników

Procedura ta służy do wynajmowania (tworzenia odpowiedniego wpisu w tabeli `najmy`) obiektów przez użytkowników.
Procedura przyjmuje 1 parametr:

- `@obiekt_id INT`

Procedura zwraca następujące wartości:

- 0 - Użytkownik dodany pomyślnie
- 1 - Wystąpił nieznany błąd
- 2 - Obiekt o podanym identyfikatorze nie istnieje
- 3 - Obiekt o podanym identyfikatorze jest obecnie zajęty

```
CREATE PROCEDURE wynajmij_obiekt
   @obiekt_id INT
   WITH EXECUTE AS OWNER
   AS
      IF NOT EXISTS(SELECT * FROM obiekty WHERE id = @obiekt_id)
        RETURN 2;

      IF EXISTS(SELECT * FROM obiekty WHERE id = @obiekt_id AND obecnie_wynajete = 't')
        RETURN 3;

      EXECUTE AS CALLER;
      DECLARE @login VARCHAR(75);
      SELECT @login = CURRENT_USER;
      REVERT;

      BEGIN TRANSACTION;
      INSERT INTO najmy (uzytkownik_id, obiekt_id, data_rozpoczecia, data_zakonczenia, koszt)
        VALUES ((SELECT u.id FROM dbo.uzytkownicy u WHERE u.login COLLATE
      SQL_Latin1_General_CP1_CS_AS = @login COLLATE SQL_Latin1_General_CP1_CS_AS), @obiekt_id,
      DEFAULT, NULL, NULL);

      IF @@ERROR<>0
        GOTO BLAD;

      COMMIT TRANSACTION;
      RETURN 0;

      BLAD:
        ROLLBACK TRANSACTION;
```

```
28          RETURN 1;
```

Listing 18: Skrypt tworzący procedurę składowaną `wynajmij_obiekt`

```
1 DECLARE @wynik INT;
2 EXEC @wynik = wynajmij_obiekt @obiekt_id = 17;
3 SELECT 'Zwr cony Wynik' = @wynik;
```

Listing 19: Przykład użycia procedury `wynajmij_obiekt`

5.8 Użytkownicy, role i uprawnienia

Przewidziane zostały 3 role dla użytkowników:

- `admini_systemu` - Administratorzy - zarządzają obiektami, kategoriami i miastami
- `operatorzy_systemu` - Operatorzy - zarządzają najmami (wynajem i zwroty)
- `uzytkownicy_systemu` - Użytkownicy - zarządzają swoimi danymi oraz najmami

Do tych ról przypisane zostały uprawnienia zgodnie z tabelą 5.8.
Dodatkowo fabrycznie zostali utworzeni dwaj użytkownicy (listing 20 i listing 21):

- `admin` z hasłem `yourStrong(!)Password` przypisany do roli `admini_systemu`
- `operator` z hasłem `yourStrong(!)Password` przypisany do roli `operatorzy_systemu`

Kolejnych użytkowników roli `uzytkownicy_systemu`, można tworzyć za pomocą procedury składowanej z listingu 17 - `utworz_uzytkownika` (strona 17).

	Administratorzy	**Operatorzy**	**Użytkownicy**
Tabele			
`kategorie`	S, I, U, D	S	S
`miasta`	S, I, U, D	S	S
`dzielnice`	S, I, U, D	S	S
`obiekty`	S, I, U, D	S	S
`uzytkownicy`	S	S	S[3], U[4]
`najmy`	S, I, U	S, I, U	S[5], U[6]
Widoki			
`lista_najmow`	S	S	-
`lista_niepopularnych_obiektow`	S	-	-
`lista_popularnosci_obiektow`	S	-	-
Procedury			
`utworz_uzytkownika`	tylko `dbo`		
`wynajmij_obiekt`	-	-	X
S - SELECT, I - INSERT, U - UPDATE, X - EXECUTE			

Tabela 1: Role i ich uprawnienia

```
1  CREATE ROLE admini_systemu;
2  GO;
3
4  GRANT SELECT, INSERT, UPDATE, DELETE ON kategorie TO admini_systemu;
5  GO;
6
7  GRANT SELECT, INSERT, UPDATE, DELETE ON miasta TO admini_systemu;
8  GO;
9
10 GRANT SELECT, INSERT, UPDATE, DELETE ON dzielnice TO admini_systemu;
11 GO;
12
13 GRANT SELECT, INSERT, UPDATE, DELETE ON obiekty TO admini_systemu;
```

[3] wybieranie i edycja ograniczone tylko do swoich danych
[4] tylko pola nazwisko, imie, wiek, adres, telefon, plec
[5] wybieranie i edycja ograniczone tylko do swoich danych
[6] tylko pole data_zakonczenia

```
GO;

GRANT SELECT, UPDATE, DELETE ON uzytkownicy TO admini_systemu;
GO;

GRANT SELECT, INSERT, UPDATE ON najmy TO admini_systemu;
GO;

GRANT SELECT ON lista_najmow TO admini_systemu;
GO;

GRANT SELECT ON lista_niepopularnych_obiektow TO admini_systemu;
GO;

GRANT SELECT ON lista_popularnosci_obiektow TO admini_systemu;
GO;

CREATE USER admin WITH PASSWORD ='yourStrong(!)Password';
GO;

EXEC sp_addrolemember 'admini_systemu', 'admin';
GO;
```

Listing 20: Skrypt tworzący rolę i użytkownika administracyjnego

```
CREATE ROLE operatorzy_systemu;
GO;

GRANT SELECT ON kategorie TO operatorzy_systemu;
GO;

GRANT SELECT ON miasta TO operatorzy_systemu;
GO;

GRANT SELECT ON dzielnice TO operatorzy_systemu;
GO;

GRANT SELECT ON obiekty TO operatorzy_systemu;
GO;

GRANT SELECT ON uzytkownicy TO operatorzy_systemu;
GO;

GRANT SELECT, INSERT, UPDATE ON najmy TO operatorzy_systemu;
GO;

GRANT SELECT ON lista_najmow TO operatorzy_systemu;
GO;

CREATE USER operator WITH PASSWORD ='yourStrong(!)Password';
GO;

EXEC sp_addrolemember 'operatorzy_systemu', 'operator';
GO;
```

Listing 21: Skrypt tworzący rolę i użytkownika operatorskiego

5.8.1 Ograniczenie uprawnień na poziomie wiersza

Uwaga: Funkcjonalność Row Level Security (RLS) została wprowadzona w SQL Server 2016. Próba uruchomienia tych skryptów na wersji niższej niż 2016 skończy się niepowodzeniem.
W tabelach `uzytkownicy` oraz `najmy` została ograniczona widoczność grupie `uzytkownicy_systemu` tylko do rekordów powiązanych z zalogowanym użytkownikiem. Jest to możliwe dzięki funkcjonalności Row Level Security. Sposób utworzenia ograniczenia RLS można zobaczyć na listingu 22 oraz listingu 23.

```
CREATE FUNCTION fn_securitypredicate_uzytkownicy (@login_field SYSNAME)
  RETURNS TABLE
  WITH SCHEMABINDING
  AS
    RETURN SELECT 1 AS fn_securitypredicate_uzytkownicy_result
      FROM dbo.uzytkownicy
      WHERE (
```

```
        @login_field = user_name()
        OR IS_ROLEMEMBER('admini_systemu', user_name()) = 1
        OR IS_ROLEMEMBER('operatorzy_systemu', user_name()) = 1
        OR user_name() = 'dbo'
      )
GO;

CREATE SECURITY POLICY fn_security_uzytkownicy
  ADD FILTER PREDICATE
  dbo.fn_securitypredicate_uzytkownicy(login)
  ON dbo.uzytkownicy
GO;

ALTER SECURITY POLICY fn_security_uzytkownicy WITH (state = on);
GO;
```

Listing 22: Skrypt ustawiający ograniczenie RLS na tabeli `uzytkownicy`

```
CREATE FUNCTION fn_securitypredicate_najmy (@login_field SYSNAME)
  RETURNS TABLE
  WITH SCHEMABINDING
  AS
    RETURN SELECT 1 AS fn_securitypredicate_najmy_result
      FROM dbo.najmy n
      WHERE (
        @login_field = (SELECT u.id FROM dbo.uzytkownicy u WHERE u.login COLLATE
    SQL_Latin1_General_CP1_CS_AS = CURRENT_USER COLLATE SQL_Latin1_General_CP1_CS_AS)
        OR IS_ROLEMEMBER('admini_systemu', user_name()) = 1
        OR IS_ROLEMEMBER('operatorzy_systemu', user_name()) = 1
        OR user_name() = 'dbo'
      )
GO;

CREATE SECURITY POLICY fn_security_najmy
  ADD FILTER PREDICATE
  dbo.fn_securitypredicate_najmy(uzytkownik_id)
  ON dbo.najmy
GO;

ALTER SECURITY POLICY fn_security_najmy WITH (state = on);
GO;
```

Listing 23: Skrypt ustawiający ograniczenie RLS na tabeli `najmy`

5.8.2 Test uprawnień użytkowników

W pierwszy teście sprawdzimy czy do widoku `lista_najmow`, uprawnienia mają tylko użytkownicy grup `operatorzy_systemu` oraz `admini_systemu`.

```
PRINT 'Uruchamiam jako: admin'
EXECUTE AS USER = 'admin';
SELECT TOP 5 nazwisko, nazwa_obiektu, data_rozpoczecia FROM lista_najmow;
REVERT;

PRINT 'Uruchamiam jako: operator'
EXECUTE AS USER = 'operator';
SELECT TOP 5 nazwisko, nazwa_obiektu, data_rozpoczecia FROM lista_najmow;
REVERT;

PRINT 'Uruchamiam jako: login_1'
EXECUTE AS USER = 'login_1';
SELECT TOP 5 nazwisko, nazwa_obiektu, data_rozpoczecia FROM lista_najmow;
REVERT;
```

Listing 24: Skrypt testujący uprawnienia do wyświetlania widoku `lista_najmow`

```
Uruchamiam jako: admin
nazwisko                          |nazwa_obiektu                      |data_rozpoczecia
----------------------------------|-----------------------------------|---------------
Jan                               |Nazwa 3                            |      2018-12-02
Jan                               |Nazwa 2                            |      2018-12-09
```

```
Jan                              |Nazwa 4                          |       2018-12-13
Jan                              |Nazwa 2                          |       2018-12-28
Jan                              |Nazwa 2                          |       2018-12-01

(5 rows affected)
Uruchamiam jako: operator
nazwisko                         |nazwa_obiektu                    |data_rozpoczecia
---------------------------------|---------------------------------|---------------
Jan                              |Nazwa 3                          |       2018-12-02
Jan                              |Nazwa 2                          |       2018-12-09
Jan                              |Nazwa 4                          |       2018-12-13
Jan                              |Nazwa 2                          |       2018-12-28
Jan                              |Nazwa 2                          |       2018-12-01

(5 rows affected)
Uruchamiam jako: login_1
Msg 229, Level 14, State 5, Server 7c9f95a221a5, Line 13
The SELECT permission was denied on the obj
ect 'lista_najmow', database 'projekt', schema 'dbo'.
```

Listing 25: Wynik uruchomienia skryptu testującego uprawnienia do wyświetlania widoku `lista_najmow`

W drugim teście sprawdzimy czy `uzytkownicy_systemu` widzą tylko swoje dane w tabeli `uzytkownicy`, a `operatorzy_systemu` oraz `admini_systemu` widzą wszystkich 4 użytkowników.

```
PRINT 'Uruchamiam jako: admin'
EXECUTE AS USER = 'admin';
SELECT id, login, nazwisko FROM uzytkownicy;
REVERT;

PRINT 'Uruchamiam jako: operator'
EXECUTE AS USER = 'operator';
SELECT id, login, nazwisko FROM uzytkownicy;
REVERT;

PRINT 'Uruchamiam jako: login_1'
EXECUTE AS USER = 'login_1';
SELECT id, login, nazwisko FROM uzytkownicy;
REVERT;

PRINT 'Uruchamiam jako: login_2'
EXECUTE AS USER = 'login_2';
SELECT id, login, nazwisko FROM uzytkownicy;
REVERT;

PRINT 'Uruchamiam jako: login_3'
EXECUTE AS USER = 'login_3';
SELECT id, login, nazwisko FROM uzytkownicy;
REVERT;

PRINT 'Uruchamiam jako: login_4'
EXECUTE AS USER = 'login_4';
SELECT id, login, nazwisko FROM uzytkownicy;
REVERT;
```

Listing 26: Skrypt testujący uprawnienia do wyświetlania tabeli `uzytkownicy`

```
Uruchamiam jako: admin
id          |login                           |nazwisko
------------|--------------------------------|--------------------------------
          1|login_1                         |Jan
          2|login_2                         |Ewa
          3|login_3                         |Ignacy
          4|login_4                         |Julia

(4 rows affected)
Uruchamiam jako: operator
id          |login                           |nazwisko
------------|--------------------------------|--------------------------------
          1|login_1                         |Jan
          2|login_2                         |Ewa
          3|login_3                         |Ignacy
          4|login_4                         |Julia
```

```
(4 rows affected)
Uruchamiam jako: login_1
id          |login                           |nazwisko
------------+--------------------------------+--------------------------------
           1|login_1                         |Jan

(1 rows affected)
Uruchamiam jako: login_2
id          |login                           |nazwisko
------------+--------------------------------+--------------------------------
           2|login_2                         |Ewa

(1 rows affected)
Uruchamiam jako: login_3
id          |login                           |nazwisko
------------+--------------------------------+--------------------------------
           3|login_3                         |Ignacy

(1 rows affected)
Uruchamiam jako: login_4
id          |login                           |nazwisko
------------+--------------------------------+--------------------------------
           4|login_4                         |Julia

(1 rows affected)
```

Listing 27: Wynik uruchomienia skryptu testującego uprawnienia do wyświetlania tabeli `uzytkownicy`

6 Skrypt usuwający obiekty z bazy danych

Ponieważ do każdej operacji tworzącej lub modyfikującej obiekty w bazie danych została napisana również operacja odwrotna, to możliwe jest wygenerowanie skryptu `skrypt_usuwajacy_obiekty_z_bazy.sql`.

6.1 Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym

Jak widać na listingu 28, skrypt podaje bardzo dokładne informacje na temat aktualnie wykonywanej operacji. W przypadku tego skryptu, operacje są wykonywane w odwrotnej kolejności niż w skrypcie tworzącym z listingu 3.

```
Wersja 36: 'Utworzenie triggeru aktualizujacego stan obiekty'

(1 rows affected)
Wersja 36: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 35
Wersja 35: 'Nadanie uprawnien wiersza w tabeli najmy'

(1 rows affected)
Wersja 35: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 34
Wersja 34: 'Nadanie uprawnien wiersza w tabeli uzytkownicy'

(1 rows affected)
Wersja 34: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 33
Wersja 33: 'Nadanie uprawnien roli uzytkownika systemu'

(1 rows affected)
Wersja 33: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 32
Wersja 32: 'Utworzenie procedury skladowanej do tworzenia uzytkownikow'

(1 rows affected)
Wersja 32: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 31
Wersja 31: 'Utworzenie procedury skladowanej wynajmowania obiektow'

(1 rows affected)
Wersja 31: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 30
Wersja 30: 'Utworzenie roli uzytkownika systemu'

(1 rows affected)
Wersja 30: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 29
Wersja 29: 'Przypisanie uzytkownikowi operatora roli operatora systemu'

(1 rows affected)
Wersja 29: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 28
Wersja 28: 'Utworzenie uzytkownika operatora'

(1 rows affected)
Wersja 28: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 27
Wersja 27: 'Nadanie uprawnien roli operatora systemu'

(1 rows affected)
Wersja 27: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 26
Wersja 26: 'Utworzenie roli operatora systemu'

(1 rows affected)
Wersja 26: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 25
Wersja 25: 'Przypisanie uzytkownikowi administracyjnemu roli administratora systemu'

(1 rows affected)
Wersja 25: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 24
Wersja 24: 'Utworzenie uzytkownika administracyjnego'

(1 rows affected)
Wersja 24: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 23
Wersja 23: 'Nadanie uprawnien roli administratora systemu'

(1 rows affected)
Wersja 23: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 22
Wersja 22: 'Utworzenie roli administratora systemu'

(1 rows affected)
Wersja 22: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 21
Wersja 21: 'Utworzenie funkcji wyswietlajacej spozniajacych sie uzytkownikow'
```

```
(1 rows affected)
Wersja 21: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 20
Wersja 20: 'Utworzenie funkcji wyswietlajacej adresowke uzytkownika'

(1 rows affected)
Wersja 20: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 19
Wersja 19: 'Utworzenie triggeru aktualizujacego koszt najmu'

(1 rows affected)
Wersja 19: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 18
Wersja 18: 'Utworzenie funkcji wyliczajacej koszt konkretnego najmu'

(1 rows affected)
Wersja 18: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 17
Wersja 17: 'Utworzenie funkcji wyliczajacej koszt najmu obiektu'

(1 rows affected)
Wersja 17: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 16
Wersja 16: 'Utworzenie widoku z lista niewynajmowanych obiektow'

(1 rows affected)
Wersja 16: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 15
Wersja 15: 'Utworzenie widoku z lista popularnosci obiektow'

(1 rows affected)
Wersja 15: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 14
Wersja 14: 'Utworzenie widoku z lista wszystkich najmow'

(1 rows affected)
Wersja 14: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 13
Wersja 13: 'Utworzenie indeksu unikatowego w tabeli z uzytkownikami'

(1 rows affected)
Wersja 13: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 12
Wersja 12: 'Utworzenie relacji pomiedzy obiektami a najmami'

(1 rows affected)
Wersja 12: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 11
Wersja 11: 'Utworzenie relacji pomiedzy uzytkownikami a najmami'

(1 rows affected)
Wersja 11: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 10
Wersja 10: 'Utworzenie indeksu unikatowego w tabeli z najemcami'

(1 rows affected)
Wersja 10: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 9
Wersja 9: 'Utworzenie tabeli z najmami'

(1 rows affected)
Wersja 9: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 8
Wersja 8: 'Utworzenie tabeli z uzytkownikami'

(1 rows affected)
Wersja 8: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 7
Wersja 7: 'Utworzenie relacji pomiedzy kategoriami a obiektami'

(1 rows affected)
Wersja 7: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 6
Wersja 6: 'Utworzenie relacji pomiedzy dzielnicami a obiektami'

(1 rows affected)
Wersja 6: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 5
Wersja 5: 'Utworzenie tabeli z obiektami'

(1 rows affected)
Wersja 5: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 4
Wersja 4: 'Utworzenie tabeli z kategoriami'

(1 rows affected)
Wersja 4: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 3
Wersja 3: 'Utworzenie relacji pomiedzy miastami a dzielnicami'

(1 rows affected)
Wersja 3: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 2
```

```
137 Wersja 2: 'Utworzenie tabeli z dzielnicami'
138
139 (1 rows affected)
140 Wersja 2: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 1
141 Wersja 1: 'Utworzenie tabeli z miastami'
142
143 (1 rows affected)
144 Wersja 1: Migracja zostala odinstalowana pomyslnie - teraz baza jest w wersji 0
145 Tabela wersjonowania zostala skasowana
```

Listing 28: Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym

7 Generator bilingów (skrypt oparty na kursorach)

Został przygotowany jeden skrypt służący do wygenerowania i wydrukowania bilingu dla każdego użytkownika. Ponieważ plikiem wynikowym jest czysty plik tekstowy, to można od razu go wysłać do wydrukowania na drukarkę igłową. Pomiędzy poszczególnymi bilingami zostały dodane znaczniki Form Feed (FF), dzięki czemu każdy biling będzie na oddzielnej stronie.

```
DECLARE @liczba INT;

SELECT @liczba = COUNT(*)
  FROM uzytkownicy

PRINT '+' + REPLICATE('-', 28 + DATALENGTH(CONVERT(VARCHAR, @liczba))) + '+'
PRINT '| W bazie jest ' + CONVERT(VARCHAR, @liczba) + ' uzytkownikow |'
PRINT '+' + REPLICATE('-', 28 + DATALENGTH(CONVERT(VARCHAR, @liczba))) + '+'



DECLARE @etykieta VARCHAR(MAX);
DECLARE @id INT;
DECLARE @wiersz VARCHAR(MAX);
DECLARE @suma DECIMAL(15,2);

DECLARE uzytkownicy_kursor CURSOR LOCAL FOR
  SELECT u.id, dbo.adresowka(u.id) etykieta
  FROM uzytkownicy u;

OPEN uzytkownicy_kursor;
FETCH uzytkownicy_kursor INTO @id, @etykieta;
WHILE @@FETCH_STATUS = 0
BEGIN
  PRINT CHAR(12) + 'Klient: ' + CHAR(13) + @etykieta;

  DECLARE najmy_kursor CURSOR LOCAL FOR
    SELECT CONCAT(
     '| ',
      CAST(o.nazwa AS CHAR(30)),
      ' | ',
      CAST(n.data_rozpoczecia AS CHAR(20)),
      ' | ',
      CAST(n.data_zakonczenia AS CHAR(20)),
      ' | ',
      REPLICATE(' ', 10 - DATALENGTH(CAST(n.koszt AS VARCHAR))),
      CAST(n.koszt AS VARCHAR),
      ' |'
    )
    FROM najmy n
    JOIN obiekty o on n.obiekt_id = o.id
    WHERE uzytkownik_id = @id
          AND data_zakonczenia IS NOT NULL
    ORDER BY data_rozpoczecia;

  PRINT '+' + REPLICATE('-', 32) + '+' + REPLICATE('-', 22) + '+' + REPLICATE('-', 22) + '+'
    + REPLICATE('-', 12) + '+'
  PRINT '| Nazwa Obiektu' + REPLICATE(' ', 30-DATALENGTH('Nazwa Obiektu')) + ' | Od' +
    REPLICATE(' ', 20-DATALENGTH('Od')) + ' | Do' + REPLICATE(' ', 20-DATALENGTH('Do')) + ' |
     Koszt' + REPLICATE(' ', 10-DATALENGTH('Koszt')) + ' |'
  PRINT '+' + REPLICATE('-', 32) + '+' + REPLICATE('-', 22) + '+' + REPLICATE('-', 22) + '+'
    + REPLICATE('-', 12) + '+'
  OPEN najmy_kursor;
  FETCH najmy_kursor INTO @wiersz;
  WHILE @@FETCH_STATUS = 0
  BEGIN
    PRINT @wiersz;
    PRINT '+' + REPLICATE('-', 32) + '+' + REPLICATE('-', 22) + '+' + REPLICATE('-', 22) + '+
    ' + REPLICATE('-', 12) + '+'
    FETCH najmy_kursor INTO @wiersz;
  END
  CLOSE najmy_kursor;
  DEALLOCATE najmy_kursor;

  SELECT @suma = SUM(koszt)
    FROM najmy n
    WHERE uzytkownik_id = @id
```

```
            AND data_zakonczenia IS NOT NULL

    PRINT '|' + REPLICATE(' ', 32) + ' ' + REPLICATE(' ', 22) + '|' + REPLICATE(' ', 22-
      DATALENGTH('Suma: ')) + 'Suma: | ' + REPLICATE(' ', 10 - DATALENGTH(CAST(@suma AS VARCHAR
      ))) + CAST(@suma AS VARCHAR) + ' |'
    PRINT '|' + REPLICATE(' ', 32) + ' ' + REPLICATE(' ', 22) + '+' + REPLICATE('-', 22) + '+'
      + REPLICATE('-', 12) + '+'

    FETCH uzytkownicy_kursor INTO @id, @etykieta;
END
CLOSE uzytkownicy_kursor;
```

Listing 29: Oparty na kursorach skrypt do generowania bilingów

```
+-------------------------------+
| W bazie jest 4 uzytkownik w |
+-------------------------------+
```

```
Klient:
Jan Kowalski
+48 625 548 874
Ul. Kowalska 23, 02-001, Warszawa
```

Nazwa Obiektu	Od	Do	Koszt
Nazwa 2	2018-11-29	2018-12-25	1729.35
Nazwa 2	2018-11-30	2018-12-24	3000.00
Nazwa 2	2018-12-01	2018-12-27	4049.73
Nazwa 3	2018-12-02	2018-12-22	2520.00
Nazwa 1	2018-12-05	2018-12-24	2400.00
Nazwa 4	2018-12-05	2018-12-25	532.35
Nazwa 1	2018-12-06	2018-12-24	1216.95
Nazwa 1	2018-12-06	2018-12-25	2999.80
Nazwa 2	2018-12-09	2018-12-27	481.65
Nazwa 4	2018-12-13	2018-12-20	960.00
Nazwa 4	2018-12-13	2018-12-26	1680.00
		Suma:	21569.83

```
Klient:
Ewa Nowak
+48 222 111 333
Ul. Warszawska 23, 11-111, Krak w
```

Nazwa Obiektu	Od	Do	Koszt
Nazwa 1	2018-11-29	2018-12-27	3480.00
Nazwa 4	2018-11-30	2018-12-20	1345.05
Nazwa 1	2018-11-30	2018-12-24	633.75
Nazwa 3	2018-12-06	2018-12-27	2640.00
Nazwa 2	2018-12-09	2018-12-20	768.60
		Suma:	8867.40

```
Klient:
Ignacy Grzeszczak
+48 864 645 328
Ul. Krakowska 23, 33-666, Kielce
```

```
+-----------------+------------+------------+----------+
| Nazwa Obiektu   | Od         | Do         | Koszt    |
+-----------------+------------+------------+----------+
| Nazwa 4         | 2018-11-29 | 2018-12-27 |  1857.45 |
+-----------------+------------+------------+----------+
| Nazwa 4         | 2018-11-30 | 2018-12-27 |  3360.00 |
+-----------------+------------+------------+----------+
| Nazwa 3         | 2018-12-06 | 2018-12-22 |  2040.00 |
+-----------------+------------+------------+----------+
| Nazwa 2         | 2018-12-07 | 2018-12-26 |  1281.00 |
+-----------------+------------+------------+----------+
| Nazwa 4         | 2018-12-08 | 2018-12-22 |   960.75 |
+-----------------+------------+------------+----------+
| Nazwa 1         | 2018-12-11 | 2018-12-21 |  1320.00 |
+-----------------+------------+------------+----------+
| Nazwa 3         | 2018-12-11 | 2018-12-22 |  1799.88 |
+-----------------+------------+------------+----------+
| Nazwa 2         | 2018-12-12 | 2018-12-26 |   380.25 |
+-----------------+------------+------------+----------+
|                 |            |      Suma: | 12999.33 |
|                 |            +------------+----------+
```

```
Klient:
Julia Paderewska
+48 852 147 993
Ul. Ostatnia 89, 67-125, Grudziadz
+-----------------+------------+------------+----------+
| Nazwa Obiektu   | Od         | Do         | Koszt    |
+-----------------+------------+------------+----------+
| Nazwa 3         | 2018-11-29 | 2018-12-21 |  3449.77 |
+-----------------+------------+------------+----------+
| Nazwa 2         | 2018-11-29 | 2018-12-22 |   608.40 |
+-----------------+------------+------------+----------+
| Nazwa 4         | 2018-12-01 | 2018-12-26 |  3120.00 |
+-----------------+------------+------------+----------+
| Nazwa 1         | 2018-12-02 | 2018-12-22 |   532.35 |
+-----------------+------------+------------+----------+
| Nazwa 2         | 2018-12-03 | 2018-12-21 |  2849.81 |
+-----------------+------------+------------+----------+
| Nazwa 3         | 2018-12-04 | 2018-12-27 |  1537.20 |
+-----------------+------------+------------+----------+
| Nazwa 3         | 2018-12-08 | 2018-12-25 |   456.30 |
+-----------------+------------+------------+----------+
| Nazwa 4         | 2018-12-13 | 2018-12-23 |   704.55 |
+-----------------+------------+------------+----------+
|                 |            |      Suma: | 13258.38 |
|                 |            +------------+----------+
```

Listing 30: Wynik uruchomienia skryptu do generowania bilingów

8 Dane Testowe

8.1 Skrypt tworzący dane testowe

W skrypcie tworzącym, w pierwszej kolejności definiujemy zbiór danych testowych (na przykład tak jak na listingu 31), a w dalszej części tworzymy i dodajemy do bazy danych, dane testowe (listing 32 lub listing 33). Część tabel, jak na przykład tabela `uzytkownicy`, jest wypełniana w sposób losowy[7] (Listing 33). Oznacza to że wraz z każdym uruchomieniem skryptu dane dodane do bazy danych będą inne.

```
-- Przygotowanie danych testowych
DECLARE @MIASTA TABLE (nazwa VARCHAR(20));
INSERT INTO @MIASTA VALUES ('Miasto 1'),('Miasto 2'),('Miasto 3'),('Miasto 4');
```

Listing 31: Fragment deklaracji danych testowych

```
INSERT INTO dzielnice (miasto_id, nazwa)
  SELECT m.id miasto_id, d.nazwa
  FROM miasta m
  CROSS JOIN @DZIELNICE d;

SELECT @liczba_wierszy = @@ROWCOUNT;
PRINT 'Do tabeli dzielnice dodano '+CAST(@liczba_wierszy AS VARCHAR)+' wiersz(y).';
```

Listing 32: Fragment prostego tworzenia danych testowych

```
INSERT INTO obiekty (dzielnica_id, kategoria_id, nazwa, adres, dzienna_stawka_najmu,
    obecnie_wynajete)
  SELECT d.id dzielnica_id,
    k.id kategoria_id,
    (SELECT TOP 1 nazwa from @NAZWY WHERE d.id IS NOT NULL AND k.id IS NOT NULL ORDER BY
    NewID()) nazwa,
    (SELECT TOP 1 adres from @ADRESY WHERE d.id IS NOT NULL AND k.id IS NOT NULL ORDER BY
    NewID()) adres,
    (SELECT TOP 1 stawka from @STAWKI WHERE d.id IS NOT NULL AND k.id IS NOT NULL ORDER BY
    NewID()) dzienna_stawka_najmu,
    'N' obecnie_wynajete
  FROM dzielnice d
  CROSS JOIN kategorie k;

SELECT @liczba_wierszy = @@ROWCOUNT;
PRINT 'Do tabeli obiekty dodano '+CAST(@liczba_wierszy AS VARCHAR)+' wiersz(y).';
```

Listing 33: Fragment losowego tworzenia danych testowych

8.1.1 Wynik uruchomienia skryptu tworzącego dane testowe w trybie wsadowym

W momencie wykonywania skryptu, podaje on liczbę dodanych wierszy do każdej z tabel, w formacie takim jak na listingu 34.

```
Do tabeli miasta dodano 4 wiersz(y).
Do tabeli dzielnice dodano 16 wiersz(y).
Do tabeli kategorie dodano 4 wiersz(y).
Do tabeli uzytkownicy dodano 4 wiersz(y).
Do tabeli obiekty dodano 64 wiersz(y).
Do tabeli najmy dodano 48 wiersz(y).
```

Listing 34: Wynik uruchomienia całego skryptu tworzącego dane testowe w trybie wsadowym

[7] Zastanawiający może być warunek `WHERE d.id IS NOT NULL AND k.id IS NOT NULL` - jest to pewnego rodzaju "obejście" procesu optymalizacji SQL Server'a. Gdybyśmy nie zastosowali takiej struktury to dane były by wylosowane tylko za pierwszym razem - czyli wszystkie wiersze miały by taką samą wartość danego pola, a w tym przypadku takie zachowanie jest zachowaniem niepożądanym.

8.2 Skrypt usuwający dane testowe

Ponieważ nasze tabele posiadają kolumny autonumerowanie (typu `IDENTITY`), to nie możemy skasować danych przy pomocy funkcji `TRUNCATE`. Aby uzyskać podobny wynik (opróżnienie tabeli) wykorzystuję dyrektywę `DBCC CHECKIDENT(<tabela>, RESEED, 0)` która powoduje zresetowanie autonumerowania w tabeli.

```
1 DELETE FROM najmy WHERE 1=1 DBCC CHECKIDENT (najmy, RESEED, 0);
2 PRINT 'Tabela ''najmy'' zosta a opr  niona';
```

Listing 35: Fragment skryptu usuwającego dane testowe

8.2.1 Wynik uruchomienia skryptu usuwającego dane testowe w trybie wsadowym

```
1
2  (48 rows affected)
3  Checking identity information: current identity value '48'.
4  DBCC execution completed. If DBCC printed error messages, contact your system administrator.
5  Tabela 'najmy' zostala opr zniona
6
7  (4 rows affected)
8  Checking identity information: current identity value '4'.
9  DBCC execution completed. If DBCC printed error messages, contact your system administrator.
10 Tabela 'uzytkownicy' zostala opr zniona
11
12 (64 rows affected)
13 Checking identity information: current identity value '64'.
14 DBCC execution completed. If DBCC printed error messages, contact your system administrator.
15 Tabela 'obiekty' zostala opr zniona
16
17 (4 rows affected)
18 Checking identity information: current identity value '4'.
19 DBCC execution completed. If DBCC printed error messages, contact your system administrator.
20 Tabela 'kategorie' zostala opr zniona
21
22 (16 rows affected)
23 Checking identity information: current identity value '16'.
24 DBCC execution completed. If DBCC printed error messages, contact your system administrator.
25 Tabela 'dzielnice' zostala opr zniona
26
27 (4 rows affected)
28 Checking identity information: current identity value '4'.
29 DBCC execution completed. If DBCC printed error messages, contact your system administrator.
30 Tabela 'miasta' zostala opr zniona
```

Listing 36: Wynik uruchomienia całego skryptu usuwającego dane testowe w trybie wsadowym

Spis listingów

Spis rysunków

Spis tabel