

Projekt Zaliczeniowy
Komputerowy System Wspomagania Wynajmu Krótkoterminowego

Krystian Duma - Grupa Z501 - Nr. Albumu 7763

Grudzień 2018

Spis treści

Spis treści	2
1 Krótki opis słowny projektu	3
2 Założenia do projektu	3
3 Środowisko Projektowe	3
4 Model fizyczny bazy danych	4
5 Skrypt tworzący obiekty w bazie danych	5
5.1 Model wersjonowania bazy danych	5
5.2 Wynik uruchomienia całego skryptu tworzącego obiekty w trybie wsadowym	5
5.3 Tabele	7
5.3.1 Utworzenie tabeli z obiektami	8
5.3.2 Utworzenie tabeli z najmami i utworzenie indeksu unikatowego w tabeli z najmami	9
5.4 Utworzenie relacji pomiędzy użytkownikami a najmami	9
5.5 Widoki	10
5.5.1 lista_najmow - Lista wszystkich najmów	10
5.5.2 lista_niepopularnych_obiektow - Lista obiektów które nie zostały nigdy wynajęte	11
5.6 Funkcje Skalarne i Tabelarne	11
5.6.1 koszt_najmu_obiektu - Funkcja obliczająca koszt najmu wskazanego obiektu	12
5.6.2 koszt_najmu - Funkcja obliczająca koszt danego najmu	12
5.6.3 adresowka - Funkcja generująca etykietę adresową dla użytkownika	12
5.6.4 opoznieni - Funkcja generująca etykiety dla użytkowników którzy coś wynajęli ale jeszcze nie oddali	13
5.7 Triggery	13
5.7.1 wylicz_koszt_najmu - Trigger zapisujący koszt najmu	13
5.8 Procedury Składowane	14
5.8.1 utworz_uzytkownika - Procedura do tworzenia użytkowników	14
5.9 Skrypty w oparciu o kursory	15
5.10 Inne poznane obiekty, własności bazy danych	15
5.11 Użytkownicy, role i uprawnienia	15
6 Skrypt usuwający obiekty z bazy danych	16
6.1 Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym	16
7 Dane Testowe	19
7.1 Skrypt tworzący dane testowe	19
7.1.1 Wynik uruchomienia skryptu tworzącego dane testowe w trybie wsadowym	19
7.2 Skrypt usuwający dane testowe	19
7.2.1 Wynik uruchomienia skryptu usuwającego dane testowe w trybie wsadowym	20
Spis listingów	21
Spis rysunków	21
Spis tabel	21

1 Krótki opis słowny projektu

Projekt zawiera założenia do bazy danych przechowującej podstawowe informacje o wybranych funkcjach systemu informatycznego wspierającego 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 związana z wynajęciem **Obiektu** przez **Użytkownika**.

3. Szczegóły opisu

- **Użytkownik** - potrzeba przechowania informacji: nazwisko klienta, imię klienta, wiek klienta, adres zamieszkania klienta, telefon klienta, płeć 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-najemcy, 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 ([L^AT_EX](#)) oraz skryptów tworzących i usuwających obiekty z bazy ([PHP](#)). Dodatkowo na serwerze została skonfigurowana opcja `contained database authentication` dzięki której możliwe jest tworzenie i autoryzacja użytkowników w bazie SQL.

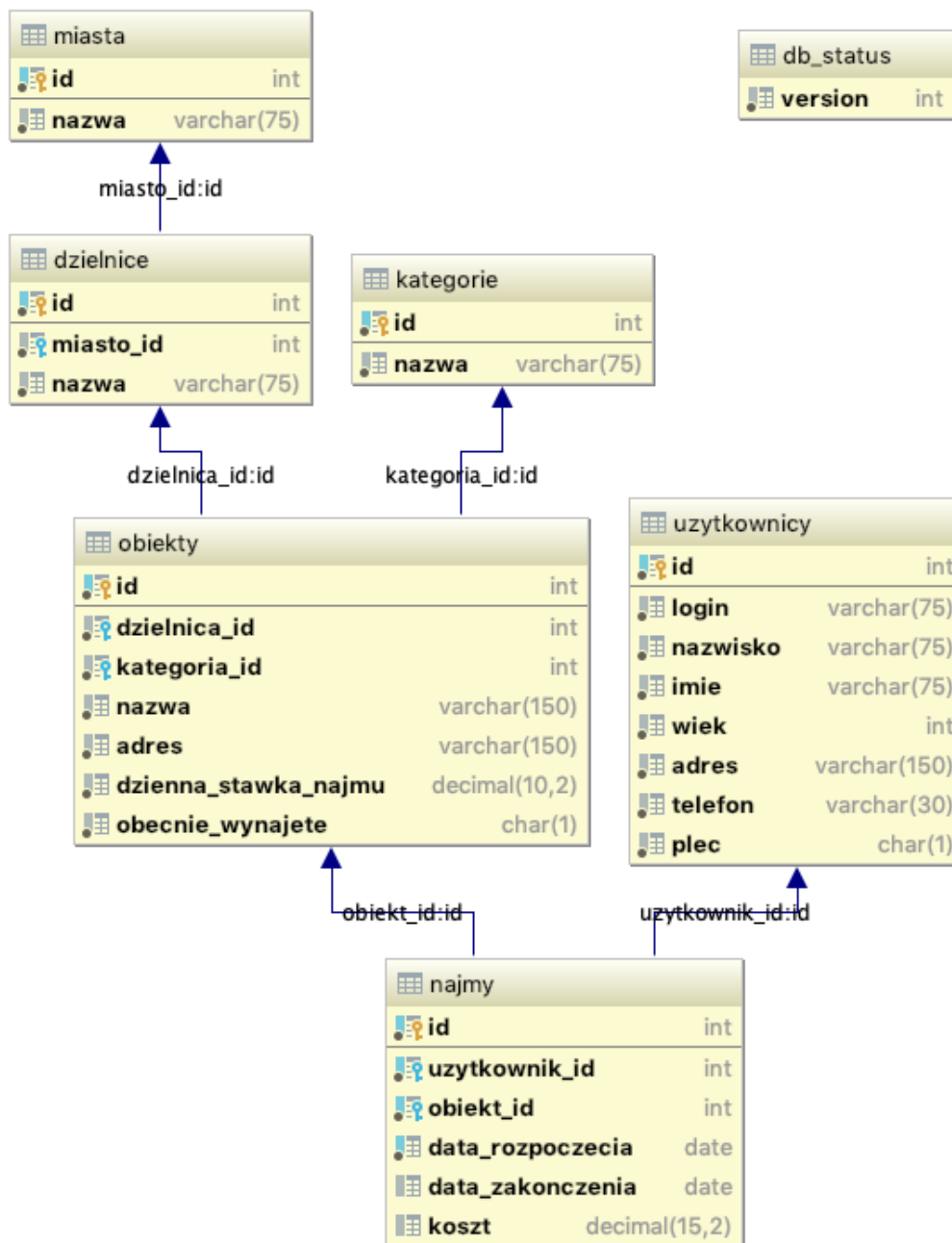
Jako aplikację służącą do łączenia się i wykonywania poleceń wykorzystane zostały aplikacje:

- 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



Powered by yFiles

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.

```
1 PRINT 'Wersja X: ' '<<< OPIS OPERACJI >>>' ''
2 IF EXISTS(SELECT * FROM sys.tables WHERE name = N'db_status')
3 BEGIN
4     IF EXISTS(SELECT * FROM db_status WHERE version = X)
5         BEGIN
6
7             <<< MIEJSCE NA KOD >>>
8
9             UPDATE db_status SET version = 1 WHERE version = X;
10            PRINT 'Wersja X: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji
11            X';
12        END
13    ELSE
14        BEGIN
15            IF EXISTS(SELECT * FROM db_status WHERE version < X)
16                BEGIN
17                    RAISERROR ('Wersja X: Baza danych jest w za niskiej wersji (wymagana jest wersja
18                    X) aby zainstalować migracje', 11, 2);
19                END
20            ELSE
21                BEGIN
22                    PRINT 'Wersja X: Migracja już została zainstalowana wcześniej';
23                END
24        END
25    BEGIN
26        RAISERROR ('Wersja X: Nie znaleziono tabeli wersjonowania bazy danych', 11, 1);
27    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 działanie¹ i przechodził od razu do bloku `CATCH`.

```
1 BEGIN CATCH
2
3     SELECT
4         ERROR_NUMBER() AS ErrorNumber,
5         ERROR_SEVERITY() AS ErrorSeverity,
6         ERROR_STATE() AS ErrorState,
7         ERROR_PROCEDURE() AS ErrorProcedure,
8         ERROR_LINE() AS ErrorLine,
9         ERROR_MESSAGE() AS ErrorMessage;
10
11 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`.

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

```

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

```

```

75 Wersja 18: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 18
76 Wersja 19: 'Utworzenie triggeru aktualizującego koszt najmu'
77
78 (1 rows affected)
79 Wersja 19: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 19
80 Wersja 20: 'Utworzenie funkcji wyświetlającej adresówkę użytkownika'
81
82 (1 rows affected)
83 Wersja 20: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 20
84 Wersja 21: 'Utworzenie funkcji wyświetlającej spóźniających się użytkowników'
85
86 (1 rows affected)
87 Wersja 21: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 21
88 Wersja 22: 'Utworzenie roli administratora systemu'
89
90 (1 rows affected)
91 Wersja 22: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 22
92 Wersja 23: 'Nadanie uprawnień roli administratora systemu'
93
94 (1 rows affected)
95 Wersja 23: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 23
96 Wersja 24: 'Utworzenie użytkownika administracyjnego'
97
98 (1 rows affected)
99 Wersja 24: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 24
100 Wersja 25: 'Przypisanie użytkownikowi administracyjnemu roli administratora systemu'
101
102 (1 rows affected)
103 Wersja 25: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 25
104 Wersja 26: 'Utworzenie roli operatora systemu'
105
106 (1 rows affected)
107 Wersja 26: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 26
108 Wersja 27: 'Nadanie uprawnień roli operatora systemu'
109
110 (1 rows affected)
111 Wersja 27: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 27
112 Wersja 28: 'Utworzenie użytkownika operatora'
113
114 (1 rows affected)
115 Wersja 28: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 28
116 Wersja 29: 'Przypisanie użytkownikowi operatora roli operatora systemu'
117
118 (1 rows affected)
119 Wersja 29: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 29
120 Wersja 30: 'Utworzenie roli użytkownika systemu'
121
122 (1 rows affected)
123 Wersja 30: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 30
124 Wersja 31: 'Nadanie uprawnień roli użytkownika systemu'
125
126 (1 rows affected)
127 Wersja 31: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 31
128 Wersja 32: 'Utworzenie procedury składowanej do tworzenia użytkowników'
129
130 (1 rows affected)
131 Wersja 32: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 32
132 Wersja 33: 'Nadanie uprawnień roli administratora i operatora systemu'
133
134 (1 rows affected)
135 Wersja 33: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 33

```

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 pomiędzy miastami a dzielnicami

- Utworzenie tabeli z kategoriami
- Utworzenie tabeli z obiektami
- Utworzenie relacji pomiędzy dzielnicami a obiektami
- Utworzenie relacji pomiędzy kategoriami a obiektami
- Utworzenie tabeli z uzytkownikami
- Utworzenie tabeli z najmami
- Utworzenie indeksu unikatowego w tabeli z najmami
- Utworzenie relacji pomiędzy uzytkownikami a najmami
- Utworzenie relacji pomiędzy 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".

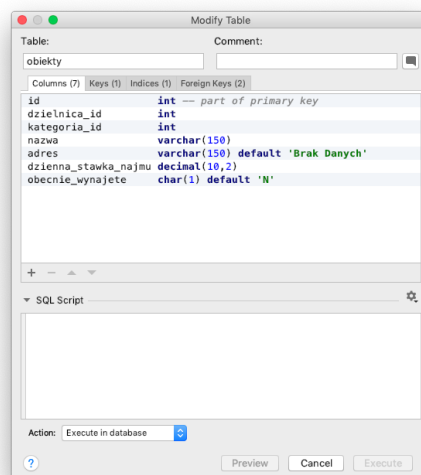
Ze względu na to aby nie zajmować zbyt dużo miejsca, poniżej zostaną przedstawione tylko najważniejsze z powyższych skryptów.

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 obiektu wynajętego oraz wolnego.



Rysunek 2: Tabela `obiekty` wyświetlona w programie `DataGrip`

```

1 CREATE TABLE obiekty (
2   id INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
3
4   dzielnica_id INT NOT NULL,
5   kategoria_id INT NOT NULL,
6
7   nazwa VARCHAR(150) NOT NULL,
8   adres VARCHAR(150) NOT NULL DEFAULT 'Brak Danych',
9   dzienna_stawka_najmu DECIMAL(10, 2) NOT NULL CHECK (dzienna_stawka_najmu > 0),
10

```



```

11     obecnie_wynajete CHAR(1) NOT NULL DEFAULT 'N' CHECK (obecnie_wynajete IN ('T', 'N')),
12 );

```

Listing 4: Skrypt tworzący tabelę obiekty

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

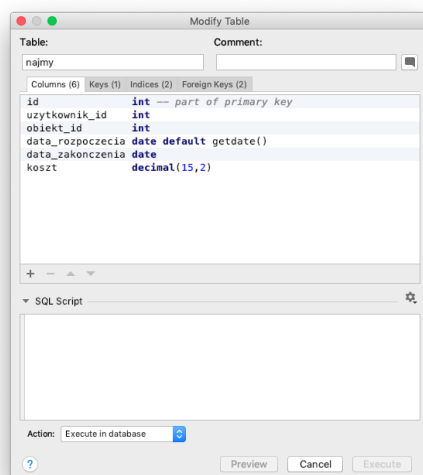
Przyjmujemy że domyślną datą rozpoczęcia najmu jest data jego dodania do bazy.

```

1 CREATE TABLE najmy (
2     id INT PRIMARY KEY NOT NULL IDENTITY (1, 1),
3
4     uzytkownik_id INT NOT NULL,
5     obiekt_id INT NOT NULL,
6
7     data_roz poczenia DATE NOT NULL DEFAULT getdate(),
8     data_zakonczenia DATE NULL,
9     koszt DECIMAL(15, 2) NULL,
10 );

```

Listing 5: Skrypt tworzący tabelę najmy



Rysunek 3: Tabela najmy wyświetlona w programie DataGrip

Ponieważ w założeniach projektowych przyjęliśmy 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_roz poczenia oraz data_zakonczenia wartość będzie ta sama. Dzięki temu możemy również utworzyć indeks unikatowy na pola uzytkownik_id, obiekt_id oraz data_roz poczenia.

```

1 CREATE UNIQUE INDEX najemcy_ui
2     ON najmy (uzytkownik_id, obiekt_id, data_roz poczenia);

```

Listing 6: Skrypt tworzący indeks unikatowy w tabeli obiekty

5.4 Utworzenie relacji pomiędzy użytkownikami a najmami

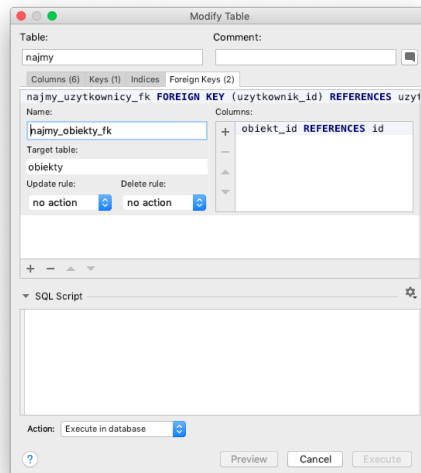
Wszystkie tabele zostały połączone relacją (CONSTRAINT) typu klucz obcy (FOREIGN KEY) tak jak na przykładzie z listingu 7.

```

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

```

Listing 7: Skrypt tworzący relację najmy-obiekty_fk



Rysunek 4: Relacja najmy_obiekty_fk wyświetlona w programie [DataGrip](#)

5.5 Widoki

Zostało stworzone kilka widoków:

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

5.5.1 lista_najmow - Lista wszystkich najmów

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

- nazwisko
- imie
- datę rozpoczęcia najmu
- datę zakończenia najmu
- nazwę wynajmowanego obiektu
- całkowity koszt najmu

	nazwisko	imie	data_roz poczenia	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**

```

1 CREATE VIEW lista_najmow
2 AS
3     SELECT nazwisko, imie, data_roz poczeczcia, data_zakonczenia, nazwa nazwa_obiektu, koszt
4         calkowity_koszt
5     FROM najmy n
6     JOIN uzytkownicy u ON n.uzytkownik_id = u.id
7     JOIN obiekty o ON n.obiekt_id = o.id;

```

Listing 8: Skrypt tworzący widok lista_najmow

5.5.2 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.

	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

```

1 CREATE VIEW lista_niepopularnych_obiektow
2 AS
3     SELECT id, nazwa, adres
4     FROM lista_popularnosci_obiektow
5     GROUP BY id, nazwa, adres
6     HAVING SUM(liczba_najmow) = 0;

```

Listing 9: Skrypt tworzący widok lista_niepopularnych_obiektow

5.6 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

5.6.1 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 13).

```
1 CREATE FUNCTION koszt_najmu_obiektu (@obiekt_id INT, @liczba_dni INT)
2 RETURNS DECIMAL(15, 2)
3 AS
4 BEGIN
5     DECLARE @dzienna_stawka DECIMAL(10,2);
6
7     SELECT @dzienna_stawka = dzienna_stawka_najmu
8     FROM obiekty o
9     WHERE id = @obiekt_id;
10
11     RETURN @liczba_dni*@dzienna_stawka;
12 END
```

Listing 10: Skrypt tworzący funkcję skalarną koszt_najmu_obiektu

5.6.2 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 13).

```
1 CREATE FUNCTION koszt_najmu (@najem_id INT)
2 RETURNS DECIMAL(15, 2)
3 AS
4 BEGIN
5     DECLARE @koszt DECIMAL(15, 2);
6     DECLARE @liczba_dni INT;
7     DECLARE @obiekt_id INT;
8     DECLARE @data_rozpoczecia DATE;
9     DECLARE @data_zakonczenia DATE;
10
11     SELECT @data_rozpoczecia = n.data_rozpoczecia,
12            @data_zakonczenia = n.data_zakonczenia,
13            @obiekt_id = o.id
14     FROM najmy n
15     JOIN obiekty o ON n.obiekt_id = o.id
16     WHERE n.id = @najem_id;
17
18     IF @data_zakonczenia IS NULL
19         RETURN NULL;
20
21     SELECT @liczba_dni = (1+DATEDIFF(DAY, @data_rozpoczecia, @data_zakonczenia));
22
23     SELECT @koszt = dbo.koszt_najmu_obiektu(@obiekt_id, @liczba_dni);
24
25     RETURN @koszt;
26 END
```

Listing 11: Skrypt tworzący funkcję skalarną koszt_najmu

5.6.3 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.

```
1 CREATE FUNCTION adresowka (@uzytkownik_id INT)
2 RETURNS VARCHAR(333) — 75+75+150+30+3 = 333 — suma d u g o c i czonych p l
```

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

```

3 AS
4 BEGIN
5     DECLARE @adresowka VARCHAR(330);
6
7     SELECT @adresowka = nazwisko + ' ' + imie + CHAR(13) + telefon + CHAR(13) + adres
8     FROM uzytkownicy
9     WHERE id = @uzytkownik_id;
10
11     RETURN @adresowka;
12 END

```

Listing 12: Skrypt tworzący funkcję skalarną `adresowka`

5.6.4 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

```

1 CREATE FUNCTION opoznieni (@liczba_dni INT)
2 RETURNS TABLE
3 AS
4 RETURN (
5     SELECT n.id najem_id,
6            o.nazwa nazwa_obiektu,
7            DATEDIFF(DAY, n.data_roz poczenia, getdate()) dni_od_wynajmu,
8            dbo.koszt_najmu_obiektu(o.id, DATEDIFF(DAY, n.data_roz poczenia, getdate()) + 1)
9            szacunkowy_koszt_najmu,
10           dbo.adresowka(u.id) etykieta
11     FROM najmy n
12     JOIN obiekty o ON n.obiekt_id = o.id
13     JOIN uzytkownicy u ON n.uzytkownik_id = u.id
14     WHERE DATEDIFF(DAY, n.data_roz poczenia, getdate()) >= @liczba_dni
15           AND n.data_zakonczenia IS NULL
16 )

```

Listing 13: Skrypt tworzący funkcję tabelarną `opoznieni`

5.7 Triggery

Zostały stworzone następujące triggery:

- `wylicz_koszt_najmu` - Trigger zapisujący koszt najmu

5.7.1 `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 10 - po dodaniu losowych danych testowych, koszty najmów zostały automatycznie obliczone.

```

1 CREATE TRIGGER wylicz_koszt_najmu
2 ON najmy
3 AFTER INSERT, UPDATE

```

```

4 AS
5 BEGIN
6     UPDATE najmy
7         SET koszt = dbo.koszt_najmu(n.id)
8         FROM najmy n
9         JOIN inserted i ON n.id = i.id
10        WHERE UPDATE (data_zakonczenia) OR NOT EXISTS(SELECT 1 FROM DELETED)
11 END

```

Listing 14: Skrypt tworzący trigger wylicz_koszt_najmu

5.8 Procedury Składowane

Zostały stworzone następujące procedury:

- **utworz_uzytkownika** - Procedura do tworzenia użytkowników

5.8.1 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

```

1 CREATE PROCEDURE utworz_uzytkownika
2     @login VARCHAR(75),
3     @nazwisko VARCHAR(75),
4     @imie VARCHAR(75),
5     @wiek INT,
6     @adres VARCHAR(150),
7     @telefon VARCHAR(30),
8     @plec CHAR(1),
9     @haslo VARCHAR(30)
10 AS
11     IF EXISTS(SELECT * FROM uzytkownicy WHERE login = @login)
12         RETURN 2;
13
14     EXEC sp_addlogin @login, @haslo;
15     EXEC sp_adduser @login;
16     EXEC sp_addrolemember 'uzytkownicy_systemu', @login;
17
18     BEGIN TRANSACTION;
19
20     INSERT INTO uzytkownicy (login, nazwisko, imie, wiek, adres, telefon, plec)
21         VALUES (@login, @nazwisko, @imie, @wiek, @adres, @telefon, @plec);
22
23     IF @@ERROR <> 0
24         GOTO BLAD;
25

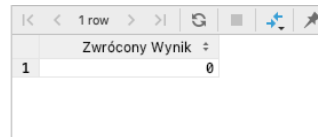
```

```

26 COMMIT TRANSACTION
27 RETURN 0;
28
29 BLAD:
30 ROLLBACK TRANSACTION
31 RETURN 1;

```

Listing 15: Skrypt tworzący procedurę składowaną `utworz_uzytkownika`



Zwrócony Wynik	
1	0

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

```

1 DECLARE @wynik INT;
2 EXEC @wynik = utworz_uzytkownika
3     @login = 'j_kowalski',
4     @nazwisko = 'Jan',
5     @imie = 'Kowalski',
6     @wiek = 25,
7     @adres = 'Ul. Kowalska 23, 02-001, Warszawa',
8     @telefon = '+48 625 548 874',
9     @plec = 'M',
10    @haslo = 'yourStrong(!)Password';
11 SELECT 'Zwrócony Wynik' = @wynik;

```

Listing 16: Przykład użycia procedury `utworz_uzytkownika`

5.9 Skrypty w oparciu o kursory

5.10 Inne poznane obiekty, własności bazy danych

5.11 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 5.11.

Dodatkowo fabrycznie zostali utworzeni dwaj użytkownicy:

- `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 16 - `utworz_uzytkownika`.

³wybieranie i edycja ograniczone tylko do swoich danych

⁴tylko pola nazwisko, imie, wiek, adres, telefon, plec

⁵wybieranie i edycja ograniczone tylko do swoich danych

⁶tylko pole `data_zakonczenia`

	Administratorzy	Operatorzy	Użytkownicy
Tabele			
katégorie	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 , I, U ⁶
Widoki			
lista_najmow	S	S	-
lista_niepopularnych_obiektow	S	-	-
lista_popularnosci_obiektow	S	-	-
Procedury			
utworz_uzytkownika	X	X	-
S - SELECT, I - INSERT, U - UPDATE, X - EXECUTE			

Tabela 1: Role i ich uprawnienia

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 17, 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.

```

1 Wersja 33: 'Nadanie uprawnień roli administratora i operatora systemu'
2
3 (1 rows affected)
4 Wersja 33: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 32
5 Wersja 32: 'Utworzenie procedury składowanej do tworzenia użytkowników'
6
7 (1 rows affected)
8 Wersja 32: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 31
9 Wersja 31: 'Nadanie uprawnień roli użytkownika systemu'
10
11 (1 rows affected)
12 Wersja 31: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 30
13 Wersja 30: 'Utworzenie roli użytkownika systemu'
14
15 (1 rows affected)
16 Wersja 30: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 29
17 Wersja 29: 'Przypisanie użytkownikowi operatora roli operatora systemu'
18
19 (1 rows affected)
20 Wersja 29: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 28
21 Wersja 28: 'Utworzenie użytkownika operatora'
22
23 (1 rows affected)
24 Wersja 28: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 27
25 Wersja 27: 'Nadanie uprawnień roli operatora systemu'
26
27 (1 rows affected)

```


28 Wersja 27: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 26
 29 Wersja 26: 'Utworzenie roli operatora systemu'
 30
 31 (1 rows affected)
 32 Wersja 26: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 25
 33 Wersja 25: 'Przypisanie użytkownikowi administracyjnemu roli administratora systemu'
 34
 35 (1 rows affected)
 36 Wersja 25: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 24
 37 Wersja 24: 'Utworzenie użytkownika administracyjnego'
 38
 39 (1 rows affected)
 40 Wersja 24: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 23
 41 Wersja 23: 'Nadanie uprawnień roli administratora systemu'
 42
 43 (1 rows affected)
 44 Wersja 23: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 22
 45 Wersja 22: 'Utworzenie roli administratora systemu'
 46
 47 (1 rows affected)
 48 Wersja 22: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 21
 49 Wersja 21: 'Utworzenie funkcji wyświetlającej spóźniających się użytkowników'
 50
 51 (1 rows affected)
 52 Wersja 21: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 20
 53 Wersja 20: 'Utworzenie funkcji wyświetlającej adresówkę użytkownika'
 54
 55 (1 rows affected)
 56 Wersja 20: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 19
 57 Wersja 19: 'Utworzenie triggeru aktualizującego koszt najmu'
 58
 59 (1 rows affected)
 60 Wersja 19: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 18
 61 Wersja 18: 'Utworzenie funkcji wyliczającej koszt konkretnego najmu'
 62
 63 (1 rows affected)
 64 Wersja 18: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 17
 65 Wersja 17: 'Utworzenie funkcji wyliczającej koszt najmu obiektu'
 66
 67 (1 rows affected)
 68 Wersja 17: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 16
 69 Wersja 16: 'Utworzenie widoku z lista niewynajmowanych obiektów'
 70
 71 (1 rows affected)
 72 Wersja 16: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 15
 73 Wersja 15: 'Utworzenie widoku z lista popularności obiektów'
 74
 75 (1 rows affected)
 76 Wersja 15: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 14
 77 Wersja 14: 'Utworzenie widoku z lista wszystkich najmów'
 78
 79 (1 rows affected)
 80 Wersja 14: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 13
 81 Wersja 13: 'Utworzenie indeksu unikatowego w tabeli z użytkownikami'
 82
 83 (1 rows affected)
 84 Wersja 13: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 12
 85 Wersja 12: 'Utworzenie relacji pomiędzy obiektami a najmami'
 86
 87 (1 rows affected)
 88 Wersja 12: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 11
 89 Wersja 11: 'Utworzenie relacji pomiędzy użytkownikami a najmami'
 90
 91 (1 rows affected)
 92 Wersja 11: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 10
 93 Wersja 10: 'Utworzenie indeksu unikatowego w tabeli z najemcami'
 94
 95 (1 rows affected)
 96 Wersja 10: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 9
 97 Wersja 9: 'Utworzenie tabeli z najmami'
 98
 99 (1 rows affected)
 100 Wersja 9: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 8
 101 Wersja 8: 'Utworzenie tabeli z użytkownikami'
 102

```

103 (1 rows affected)
104 Wersja 8: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 7
105 Wersja 7: 'Utworzenie relacji pomiędzy kategoriami a obiektami'
106
107 (1 rows affected)
108 Wersja 7: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 6
109 Wersja 6: 'Utworzenie relacji pomiędzy dzielnicami a obiektami'
110
111 (1 rows affected)
112 Wersja 6: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 5
113 Wersja 5: 'Utworzenie tabeli z obiektami'
114
115 (1 rows affected)
116 Wersja 5: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 4
117 Wersja 4: 'Utworzenie tabeli z kategoriami'
118
119 (1 rows affected)
120 Wersja 4: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 3
121 Wersja 3: 'Utworzenie relacji pomiędzy miastami a dzielnicami'
122
123 (1 rows affected)
124 Wersja 3: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 2
125 Wersja 2: 'Utworzenie tabeli z dzielnicami'
126
127 (1 rows affected)
128 Wersja 2: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 1
129 Wersja 1: 'Utworzenie tabeli z miastami'
130
131 (1 rows affected)
132 Wersja 1: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 0
133 Tabela wersjonowania została skasowana

```

Listing 17: Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym

7 Dane Testowe

7.1 Skrypt tworzący dane testowe

W skrypcie tworzącym, w pierwszej kolejności definiujemy zbiór danych testowych (na przykład tak jak na listingu 18), a w dalszej części tworzymy i dodajemy do bazy danych, dane testowe (listing 19 lub listing 20). Część tabel, jak na przykład tabela `uzytkownicy`, jest wypełniana w sposób losowy⁷ (Listing 20). Oznacza to że wraz z każdym uruchomieniem skryptu dane dodane do bazy danych będą inne.

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

Listing 18: Fragment deklaracji danych testowych

```
1 INSERT INTO dzielnice (miasto_id, nazwa)
2     SELECT m.id miasto_id, d.nazwa
3     FROM miasta m
4     CROSS JOIN @DZIELNICE d;
5
6 SELECT @liczba_wierszy = @@ROWCOUNT;
7 PRINT 'Do tabeli dzielnice dodano ' + CAST(@liczba_wierszy AS VARCHAR) + ' wiersz(y).';
```

Listing 19: Fragment prostego tworzenia danych testowych

```
1 INSERT INTO obiekty (dzielnica_id, kategoria_id, nazwa, adres, dzienna_stawka_najmu,
2     obecnie_wynajete)
3     SELECT d.id dzielnica_id,
4     k.id kategoria_id,
5     (SELECT TOP 1 nazwa from @NAZWY WHERE d.id IS NOT NULL AND k.id IS NOT NULL ORDER BY
6     NewID()) nazwa,
7     (SELECT TOP 1 adres from @ADRESY WHERE d.id IS NOT NULL AND k.id IS NOT NULL ORDER BY
8     NewID()) adres,
9     (SELECT TOP 1 stawka from @STAWKI WHERE d.id IS NOT NULL AND k.id IS NOT NULL ORDER BY
10    NewID()) dzienna_stawka_najmu,
11    'N' obecnie_wynajete
12 FROM dzielnice d
13 CROSS JOIN kategorie k;
14
15 SELECT @liczba_wierszy = @@ROWCOUNT;
16 PRINT 'Do tabeli obiekty dodano ' + CAST(@liczba_wierszy AS VARCHAR) + ' wiersz(y).';
```

Listing 20: Fragment losowego tworzenia danych testowych

7.1.1 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 21.

```
1 Do tabeli miasta dodano 4 wiersz(y).
2 Do tabeli dzielnice dodano 16 wiersz(y).
3 Do tabeli kategorie dodano 4 wiersz(y).
4 Do tabeli uzytkownicy dodano 4 wiersz(y).
5 Do tabeli obiekty dodano 64 wiersz(y).
6 Do tabeli najmy dodano 48 wiersz(y).
```

Listing 21: Wynik uruchomienia całego skryptu tworzącego dane testowe w trybie wsadowym

7.2 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) wykorzystujemy dyrektywę `DBCC CHECKIDENT(<tabela>, RESEED, 0)` która powoduje zresetowanie autonumerowania w tabeli.

⁷Zastanawiający 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.

```

1 DELETE FROM najmy WHERE 1=1 DBCC CHECKIDENT (najmy, RESEED, 0);
2 PRINT 'Tabela ''najmy'' zostala oprazona';

```

Listing 22: Fragment skryptu usuwającego dane testowe

7.2.1 Wynik uruchomienia skryptu usuwającego dane testowe w trybie wsadowym

```

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

```

Listing 23: Wynik uruchomienia całego skryptu usuwającego dane testowe w trybie wsadowym

Spis listingów

1	Szablon kodu wersjonowanego	5
2	Blok CATCH w skrypcie tworzącym	5
3	Wynik uruchomienia całego skryptu tworzącego obiekty w trybie wsadowym	6
4	Skrypt tworzący tabelę obiekty	8
5	Skrypt tworzący tabelę najmy	9
6	Skrypt tworzący indeks unikatowy w tabeli obiekty	9
7	Skrypt tworzący relację najmy_obiekty_fk	9
8	Skrypt tworzący widok lista_najmow	11
9	Skrypt tworzący widok lista_niepopularnych_obiektow	11
10	Skrypt tworzący funkcję skalarną koszt_najmu_obiektu	12
11	Skrypt tworzący funkcję skalarną koszt_najmu	12
12	Skrypt tworzący funkcję skalarną adresowka	12
13	Skrypt tworzący funkcję tabelarną opoznieni	13
14	Skrypt tworzący trigger wylicz_koszt_najmu	13
15	Skrypt tworzący procedurę składowaną utworz_uzytkownika	14
16	Przykład użycia procedury utworz_uzytkownika	15
17	Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym	16
18	Fragment deklaracji danych testowych	19
19	Fragment prostego tworzenia danych testowych	19
20	Fragment losowego tworzenia danych testowych	19
21	Wynik uruchomienia całego skryptu tworzącego dane testowe w trybie wsadowym	19
22	Fragment skryptu usuwającego dane testowe	20
23	Wynik uruchomienia całego skryptu usuwającego dane testowe w trybie wsadowym	20

Spis rysunków

1	Diagram tabel wygenerowanej bazy danych	4
2	Tabela obiekty wyświetlona w programie DataGrip	8
3	Tabela najmy wyświetlona w programie DataGrip	9
4	Relacja najmy_obiekty_fk wyświetlona w programie DataGrip	10
5	Wyświetlony widok lista_najmow	10
6	Wyświetlony widok lista_niepopularnych_obiektow	11
7	Uruchomiona funkcja opoznieni z parametrem @liczba_dni równym 3	13
8	Wynik prawidłowego uruchomienia przykładu użycia procedury utworz_uzytkownika	15

Spis tabel

1	Role i ich uprawnienia	16
---	----------------------------------	----