

Projekt Zaliczeniowy

Komputerowy System Wspomagania Wynajmu Krótkoterminowego

Krystian Duma - Grupa Z501 - Nr. Albumu 7763

Grudzień 2018

Spis treści

Spis treści	1
1 Krótki opis słowny projektu	2
2 Założenia do projektu	2
3 Środowisko Projektowe	2
4 Model fizyczny bazy danych	3
5 Skrypt tworzący obiekty w bazie danych	4
5.1 Model wersjonowania bazy danych	4
5.2 Wynik uruchomienia całego skryptu tworzącego obiekty w trybie wsadowym	4
5.3 Tabele	5
5.3.1 Utworzenie tabeli z obiektami	6
5.3.2 Utworzenie tabeli z najmami i utworzenie indeksu unikatowego w tabeli z najmami . . .	7
5.4 Utworzenie relacji pomiędzy użytkownikami a najmami	7
5.5 Widoki	8
5.5.1 lista_najmow - Lista wszystkich najmów	8
5.5.2 lista_niepopularnych_obiektow - Lista obiektów które nie zostały nigdy wynajęte . .	9
5.6 Funkcje Skalarne	9
5.7 Funkcje Tabelarne	9
5.8 Triggery	9
5.9 Procedury Składowane	9
5.10 Skrypty w oparciu o kursory	9
5.11 Inne poznane obiekty, własności bazy danych	9
5.12 Skrypt tworzący użytkowników i nadający uprawnienia	9
6 Skrypt usuwający obiekty z bazy danych	11
6.1 Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym	11
7 Dane Testowe	13
7.1 Skrypt tworzący dane testowe	13
7.1.1 Wynik uruchomienia skryptu tworzącego dane testowe w trybie wsadowym	13
7.2 Skrypt usuwający dane testowe	13
7.2.1 Wynik uruchomienia skryptu usuwającego dane testowe w trybie wsadowym	14
Spis listingów	15
Spis rysunków	15

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](#)).

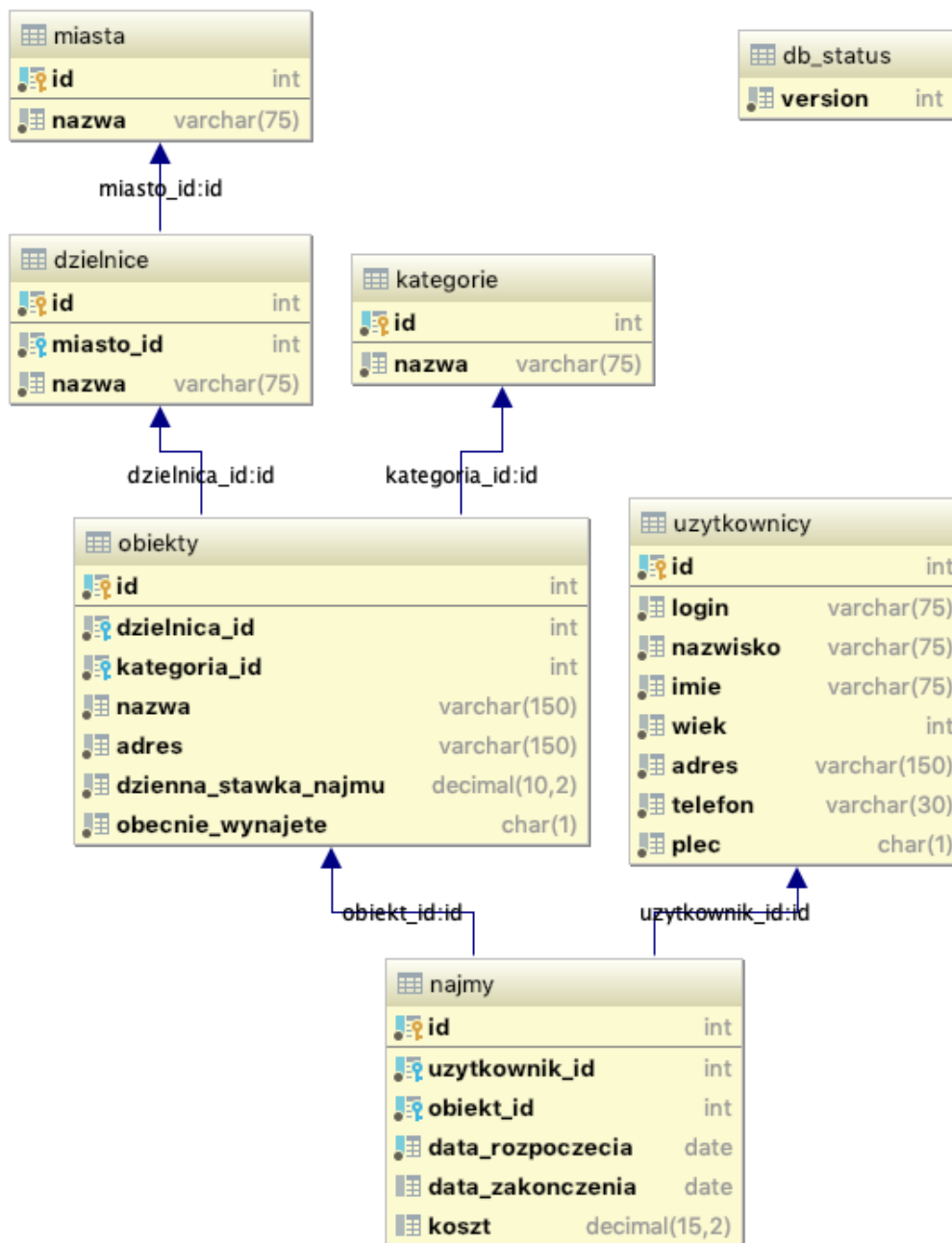
Jako aplikację służącą do łączenia się i wykonywania poleceń wykorzystane zostały aplikacje:

- 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        END
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`.

```
1
2 (1 rows affected)
```

¹Aby 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.

```

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 najmów'
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 popularności obiektów'
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 obiektów'
65
66 (1 rows affected)
67 Wersja 16: Migracja została zainstalowana pomyślnie – teraz baza jest w wersji 16

```

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

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

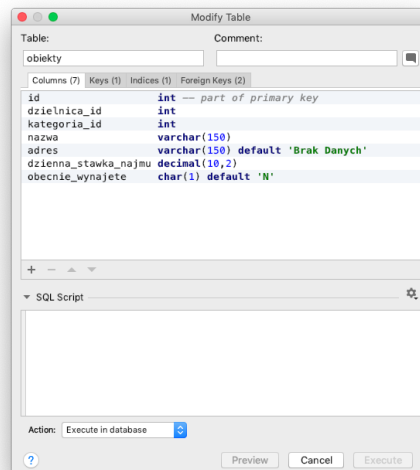
Ze względu na to aby nie zajmować zbyt dużo miejsca, poniżej zostaną przedstawione tylko najważniejsze z powyższych skryptów.

5.3.1 Tworzenie 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

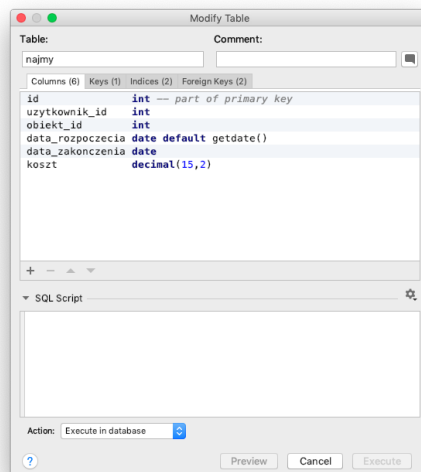
Przyjmujemy że domyślną datą rozpoczęcia najmu jest data jego dodania do bazy.

```

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 przyjeśliśmy 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

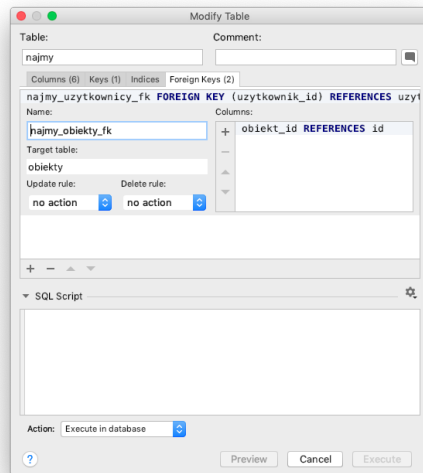
Wszystkie tabele zostały połączone relacją (CONSTRAINT) typu klucz obcy (FOREIGN KEY) tak jak na przykładzie z listingu 7.

```

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`
- `data_roz poczenia`
- `data_zakonczenia`
- `nazwa_obiektu`
- `calkowity_koszt`

	<code>nazwisko</code>	<code>imie</code>	<code>data_roz poczenia</code>	<code>data_zakonczenia</code>	<code>nazwa_obiektu</code>	<code>calkowity_koszt</code>
25	Nazwisko 4	Imie 4	2018-11-30	2018-12-23	Nazwa 4	<null>
26	Nazwisko 1	Imie 3	2018-12-08	2018-12-25	Nazwa 1	<null>
27	Nazwisko 4	Imie 1	2018-12-02	2018-12-19	Nazwa 4	<null>
28	Nazwisko 2	Imie 1	2018-12-09	2018-12-20	Nazwa 3	<null>
29	Nazwisko 1	Imie 3	2018-11-27	2018-12-23	Nazwa 3	<null>
30	Nazwisko 1	Imie 3	2018-12-10	2018-12-24	Nazwa 3	<null>
31	Nazwisko 4	Imie 4	2018-12-05	2018-12-19	Nazwa 4	<null>
32	Nazwisko 4	Imie 1	2018-12-02	2018-12-22	Nazwa 3	<null>
33	Nazwisko 4	Imie 1	2018-12-25	<null>	Nazwa 2	<null>
34	Nazwisko 2	Imie 1	2018-12-23	<null>	Nazwa 3	<null>
35	Nazwisko 4	Imie 1	2018-12-26	<null>	Nazwa 4	<null>
36	Nazwisko 4	Imie 4	2018-12-24	<null>	Nazwa 3	<null>
37	Nazwisko 4	Imie 1	2018-12-24	<null>	Nazwa 2	<null>
38	Nazwisko 4	Imie 1	2018-12-24	<null>	Nazwa 1	<null>
39	Nazwisko 2	Imie 1	2018-12-26	<null>	Nazwa 3	<null>
40	Nazwisko 1	Imie 3	2018-12-24	<null>	Nazwa 2	<null>
41	Nazwisko 4	Imie 1	2018-12-23	<null>	Nazwa 2	<null>
42	Nazwisko 2	Imie 1	2018-12-23	<null>	Nazwa 2	<null>
43	Nazwisko 4	Imie 4	2018-12-23	<null>	Nazwa 3	<null>
44	Nazwisko 4	Imie 1	2018-12-26	<null>	Nazwa 3	<null>
45	Nazwisko 1	Imie 3	2018-12-26	<null>	Nazwa 3	<null>

Rysunek 5: Wyświetlony widok `lista_najmow`


```

1 CREATE VIEW lista_najmow
2 AS
3 SELECT nazwisko, imie, data_roz poczeczia, 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

5.7 Funkcje Tabelarne

5.8 Triggery

5.9 Procedury Składowane

5.10 Skrypty w oparciu o kursory

5.11 Inne poznane obiekty, własności bazy danych

5.12 Skrypt tworzący użytkowników i nadający uprawnienia

8. SKRYPT TWORZACY UZYTKOWNIKOW BAZY DANYCH iKONTA, UPRAWNIENIA itp. W FORMIE: a. OPIS UZYTKOWNIKOW iICH ZADANIAi b. SKŁADNIA SKRYPTU c. ZRZUTY EKRANU Z:

UTWORZENIA UZYTKOWNIKOW, UPRAWNIENIA, TEST MOZLIWOSCI/NIE MOZLIWOSCI KAZ-
DEGO Z NICH

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 10, 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 16: 'Utworzenie widoku z lista niewynajmowanych obiektow'
2
3 (1 rows affected)
4 Wersja 16: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 15
5 Wersja 15: 'Utworzenie widoku z lista popularnosci obiektow'
6
7 (1 rows affected)
8 Wersja 15: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 14
9 Wersja 14: 'Utworzenie widoku z lista wszystkich najmow'
10
11 (1 rows affected)
12 Wersja 14: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 13
13 Wersja 13: 'Utworzenie indeksu unikatowego w tabeli z uzytkownikami'
14
15 (1 rows affected)
16 Wersja 13: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 12
17 Wersja 12: 'Utworzenie relacji pomiedzy obiektami a najmami'
18
19 (1 rows affected)
20 Wersja 12: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 11
21 Wersja 11: 'Utworzenie relacji pomiedzy uzytkownikami a najmami'
22
23 (1 rows affected)
24 Wersja 11: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 10
25 Wersja 10: 'Utworzenie indeksu unikatowego w tabeli z najemcami'
26
27 (1 rows affected)
28 Wersja 10: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 9
29 Wersja 9: 'Utworzenie tabeli z najmami'
30
31 (1 rows affected)
32 Wersja 9: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 8
33 Wersja 8: 'Utworzenie tabeli z uzytkownikami'
34
35 (1 rows affected)
36 Wersja 8: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 7
37 Wersja 7: 'Utworzenie relacji pomiedzy kategoriami a obiektami'
38
39 (1 rows affected)
40 Wersja 7: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 6
41 Wersja 6: 'Utworzenie relacji pomiedzy dzielnicami a obiektami'
42
43 (1 rows affected)
44 Wersja 6: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 5
45 Wersja 5: 'Utworzenie tabeli z obiektami'
46
47 (1 rows affected)
48 Wersja 5: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 4
49 Wersja 4: 'Utworzenie tabeli z kategoriami'
50
51 (1 rows affected)
52 Wersja 4: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 3
53 Wersja 3: 'Utworzenie relacji pomiedzy miastami a dzielnicami'
54
55 (1 rows affected)
56 Wersja 3: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 2
57 Wersja 2: 'Utworzenie tabeli z dzielnicami'
58
59 (1 rows affected)
60 Wersja 2: Migracja zostala odinstalowana pomyslnie – teraz baza jest w wersji 1
61 Wersja 1: 'Utworzenie tabeli z miastami'
62
```

```
63 (1 rows affected)
64 Wersja 1: Migracja została odinstalowana pomyślnie – teraz baza jest w wersji 0
65 Tabela wersjonowania została skasowana
```

Listing 10: 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 11), a w dalszej części tworzymy i dodajemy do bazy danych, dane testowe (listing 12 lub listing 13).

Część tabel, jak na przykład tabela `uzytkownicy`, jest wypełniana w sposób losowy² (Listing 13). 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 11: Fragment deklaracji danych testowych

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

Listing 12: Fragment prostego tworzenia danych testowych

```
1 — Wstawianie danych do tabeli 'uzytkownicy'
2 INSERT INTO uzytkownicy (login, nazwisko, imie, wiek, adres, telefon, plec)
3     SELECT l.login login,
4         (SELECT TOP 1 nazwisko from @NAZWISKA WHERE l.login IS NOT NULL ORDER BY NewID())
5         nazwisko,
6         (SELECT TOP 1 imie from @IMIONA WHERE l.login IS NOT NULL ORDER BY NewID()) imie,
7         (SELECT TOP 1 wiek from @WIEKI WHERE l.login IS NOT NULL ORDER BY NewID()) wiek,
8         (SELECT TOP 1 adres from @ADRESY WHERE l.login IS NOT NULL ORDER BY NewID()) adres,
9         (SELECT TOP 1 telefon from @TELEFONY WHERE l.login IS NOT NULL ORDER BY NewID()) telefon,
10        (SELECT TOP 1 plec from @PLCI WHERE l.login IS NOT NULL ORDER BY NewID()) plec
11    FROM @LOGINY l;
12
13 SELECT @liczba_wierszy = @@ROWCOUNT;
14 PRINT 'Do tabeli uzytkownicy dodano '+CAST(@liczba_wierszy AS VARCHAR)+' wiersz(y).';
```

Listing 13: 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 14.

```
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 14: 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.

```
1 DELETE FROM najmy WHERE 1=1 DBCC CHECKIDENT (najmy, RESEED, 0);
2 PRINT 'Tabela 'najmy' została opróżniona';
3
4 DELETE FROM uzytkownicy WHERE 1=1 DBCC CHECKIDENT (uzytkownicy, RESEED, 0);
5 PRINT 'Tabela 'uzytkownicy' została opróżniona';
```

²Zastanawiający może być warunek `WHERE l.login 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.

```

6
7 DELETE FROM obiekty WHERE 1=1 DBCC CHECKIDENT (obiekty, RESEED, 0);
8 PRINT 'Tabela ''obiekty'' zostala oprazona';
9
10 DELETE FROM kategorie WHERE 1=1 DBCC CHECKIDENT (kategorie, RESEED, 0);
11 PRINT 'Tabela ''kategorie'' zostala oprazona';
12
13 DELETE FROM dzielnice WHERE 1=1 DBCC CHECKIDENT (dzielnice, RESEED, 0);
14 PRINT 'Tabela ''dzielnice'' zostala oprazona';
15
16 DELETE FROM miasta WHERE 1=1 DBCC CHECKIDENT (miasta, RESEED, 0);
17 PRINT 'Tabela ''miasta'' zostala oprazona';

```

Listing 15: Skrypt usuwający dane testowe

7.2.1 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 oprazona
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 oprazona
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 oprazona
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 oprazona
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 oprazona
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 oprazona

```

Listing 16: Wynik uruchomienia całego skryptu usuwającego dane testowe w trybie wsadowym

Spis listingów

1	Szablon kodu wersjonowanego	4
2	Blok CATCH w skrypcie tworzącym	4
3	Wynik uruchomienia całego skryptu tworzącego obiekty w trybie wsadowym	4
4	Skrypt tworzący tabelę obiekty	6
5	Skrypt tworzący tabelę najmy	7
6	Skrypt tworzący indeks unikatowy w tabeli obiekty	7
7	Skrypt tworzący relację najmy_obiekty_fk	7
8	Skrypt tworzący widok lista_najmow	9
9	Skrypt tworzący widok lista_niepopularnych_obiektow	9
10	Wynik uruchomienia całego skryptu usuwającego obiekty w trybie wsadowym	11
11	Fragment deklaracji danych testowych	13
12	Fragment prostego tworzenia danych testowych	13
13	Fragment losowego tworzenia danych testowych	13
14	Wynik uruchomienia całego skryptu tworzącego dane testowe w trybie wsadowym	13
15	Skrypt usuwający dane testowe	13
16	Wynik uruchomienia całego skryptu usuwającego dane testowe w trybie wsadowym	14

Spis rysunków

1	Diagram tabel wygenerowanej bazy danych	3
2	Tabela obiekty wyświetlona w programie DataGrip	6
3	Tabela najmy wyświetlona w programie DataGrip	7
4	Relacja najmy_obiekty_fk wyświetlona w programie DataGrip	8
5	Wyświetlony widok lista_najmow	8
6	Wyświetlony widok lista_niepopularnych_obiektow	9