Bazy Danych 1 – Projekt

Prowadzący:

Dr inż. Dariusz Jankowski

Autorzy:

Emilia Augustyn, 241248

Mateusz Śliwka, 241375

Joanna Komorniczak, 241245

# Wstęp teoretyczny

## Podstawy relacyjnych baz danych

### Relacyjna baza danych

Baza danych jest zorganizowanym zbiorem danych (informacji), który przechowywany jest zazwyczaj w formie elektronicznej. Organizacja danych polega na ich odpowiednim podzieleniu i pogrupowaniu według poszczególnych pól, rekordów a także i plików, co ułatwia późniejsze pozyskiwanie, przetwarzanie i wprowadzanie informacji.

Istnieje wiele rodzajów baz danych. Obecnie najbardziej popularnym rodzajem jest **relacyjna baza danych**. Struktura tej bazy oparta jest na obiektach zwanych tabelami (relacjami), pomiędzy którymi definiuje się różne związki. Dane zapisywane są w postaci krotek. Krotki mają swoje atrybuty, a każda krotka zapisana jest w relacji.

### Postulaty Codd’a

Podstawą tego modelu stała się praca opublikowana w 1970 r. przez E.F Codda - *A Relational Model of Data for Large Shared Data Banks*. Zauważył on, że zastosowanie struktur i procesów matematycznych w zarządzaniu danymi mogłoby rozwiązać wiele problemów trapiących współczesne modele. Relacyjny model bazy danych oparty jest o algebrę relacji.

Postulaty Codd’a:

* **postulat informacyjny** - dane są reprezentowane jedynie przez wartości atrybutów w wierszach tabel (w krotkach)
* **postulat dostępu** - każda wartość w bazie danych jest dostępna poprzez podanie nazwy tabeli, atrybutu i wartości klucza podstawowego (głównego)
*  **postulat dotyczący wartości NULL** - dostępna jest specjalna wartość NULL dla reprezentacji zarówno wartości nieokreślonej, jak i nieadekwatnej, inna od wszystkich i podlegająca przetwarzaniu
* **postulat dotyczący katalogu** - wymaga się, aby system obsługiwał wbudowany katalog relacyjny z bieżącym dostępem dla uprawnionych użytkowników używających języka zapytań
* **postulat języka danych** - system musi dostarczać pełny język przetwarzania danych, który może być używany zarówno w trybie interaktywnym, jak i w obrębie programów, obsługuje operacje definiowania danych, operacje manipulowania danymi, ograniczenia związane z bezpieczeństwem i integralnością oraz operacje zarządzania transakcji
* **postulat modyfikowalności perspektyw** - system musi umożliwiać modyfikowanie perspektyw, o ile jest ono semantycznie realizowalne.
* **postulat modyfikowalności danych** - system musi umożliwiać operacje modyfikacji danych, musi obsługiwać operacje INSERT, UPDATE oraz DELETE
* **postulat fizycznej niezależności danych** - zmiany fizycznej reprezentacji danych i organizacji dostępu nie wpływają na aplikacje
* **postulat logicznej niezależności danych** - zmiany wartości w tabelach nie wpływają na aplikacje
* **postulat niezależności więzów spójności** - więzy spójności są definiowane w bazie i nie zależą od aplikacji
* **postulat niezależności dystrybucyjnej** - działanie aplikacji nie zależy od modyfikacji  i dystrybucji bazy
* **postulat bezpieczeństwa względem operacji niskiego poziomu** - operacje niskiego poziomu nie mogą naruszać modelu relacyjnego i więzów spójności

### Podstawowe pojęcia dotyczące baz danych

* Encja (relacja/tabela) - zbiór podobnych obiektów opisanych w jednolity sposób.
* Krotka (obiekt/rekord) – obiekt opisany wszystkimi atrybutami danej relacji.
* Związek (relacja/więź) – zależność występująca pomiędzy dwiema poszczególnymi tabelami.
* Atrybut – pojedyncza dana wchodząca w skład krotki określająca ją.
* Klucz główny – Taki zbiór atrybutów relacji, których kombinacje wartości jednoznacznie identyfikują każdą krotkę tej relacji a żaden podzbiór tego zbioru nie posiada tej własności.

Jeśli zbiór ten jest jednoelementowy mówimy o kluczu prostym, jeśli jest wieloelementowy – o kluczu złożonym .

* Klucz obcy - atrybut lub zbiór atrybutów, wskazujący na klucz główny w innej relacji (tabeli). Klucz obcy to nic innego jak związek, relacja między dwoma tabelami. Definicja klucza obcego, pilnuje aby w tabeli powiązanej, w określonych atrybutach, znaleźć się mogły tylko takie wartości które istnieją w tabeli docelowej jako klucz główny. Klucz obcy może dotyczyć również tej samej tabeli.

### Powiązania pomiędzy tabelami

Można mówić o trzech fundamentalnych związkach między relacjami . Dzięki nim można zapewnić integralność referencyjną danych i zamodelować pewną logikę struktury danych.

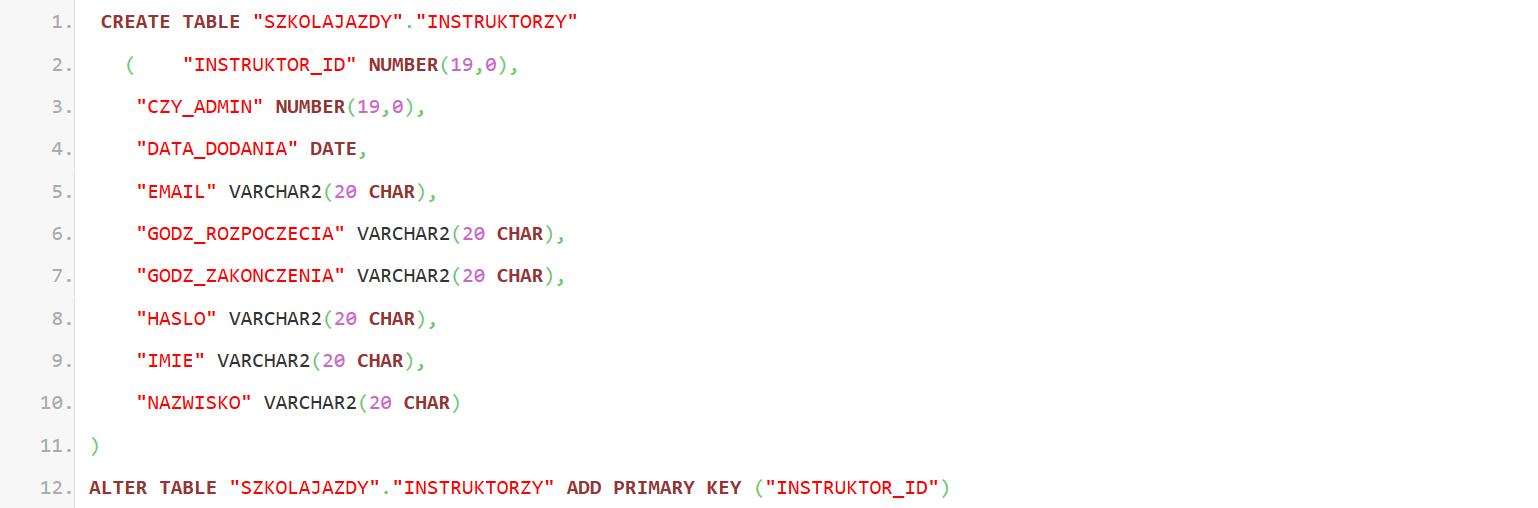
* Relacja jeden do jednego (1:1) – Każdy wiersz z tabeli A może mieć tylko jednego odpowiednika w tabeli B. Relacja 1:1 jest tworzona, jeżeli obie kolumny pokrewne są kluczami podstawowymi lub podlegają unikatowym ograniczeniom. Stosowana jest np. wtedy, gdy zbiór dodatkowych atrybutów jest określony tylko dla wąskiego podzbioru wierszy w tabeli podstawowej.
* Relacja jeden do wielu (1:N) – Każdy element z tabeli A może być powiązany z wieloma elementami z tabeli B. Pojedynczemu rekordowi z tabeli B odpowiada tylko jeden rekord z tabeli A.
* Relacja wiele do wielu (N:M) – Realizowana jest jako dwie relacje 1:N. Aby zamodelować relację N:M pomiędzy dwoma tabelami, potrzebna jest trzecia tabela zwana łącznikową. Brak wprowadzenia tabeli łącznikowej może skutkować redundancją danych.

## Normalizacja

**Normalizacja bazy danych to bezstratny proces organizowania danych w tabelach mający na celu zmniejszenie ilości danych składowanych w bazie oraz wyeliminowanie potencjalnych anomalii. Postać normalna - postać relacji w bazie danych, w której nie występuje redundancja (nadmiarowość), czyli powtarzanie się tych samych informacji. Doprowadzenia relacji do postaci normalnej nazywa się normalizacją (lub dekompozycją) bazy danych. Poniżej przedstawiony jest przykład normalizacji na podstawie realizowanej przez nas bazy danych.**

### Pierwsza postać normalna 1NF

Pierwsza postać normalna mówi o atomowości danych. Każde pole przechowuje jedną informację, dzięki czemu można dokonywać efektywnych zapytań. Wprowadza także istnienie klucza głównego. Kolejność wierszy może być dowolna, znaczenie danych nie zależy od kolejności wierszy. Tabela jest w pierwszej postaci normalnej jeśli: każdy wiersz przechowuje informację o pojedynczym obiekcie, nie zawiera kolekcji (powtarzających się grup informacji), wartości atrybutów są elementarne, posiada klucz główny. Typy przechowywanych danych powinny być najmniejsze z możliwych.

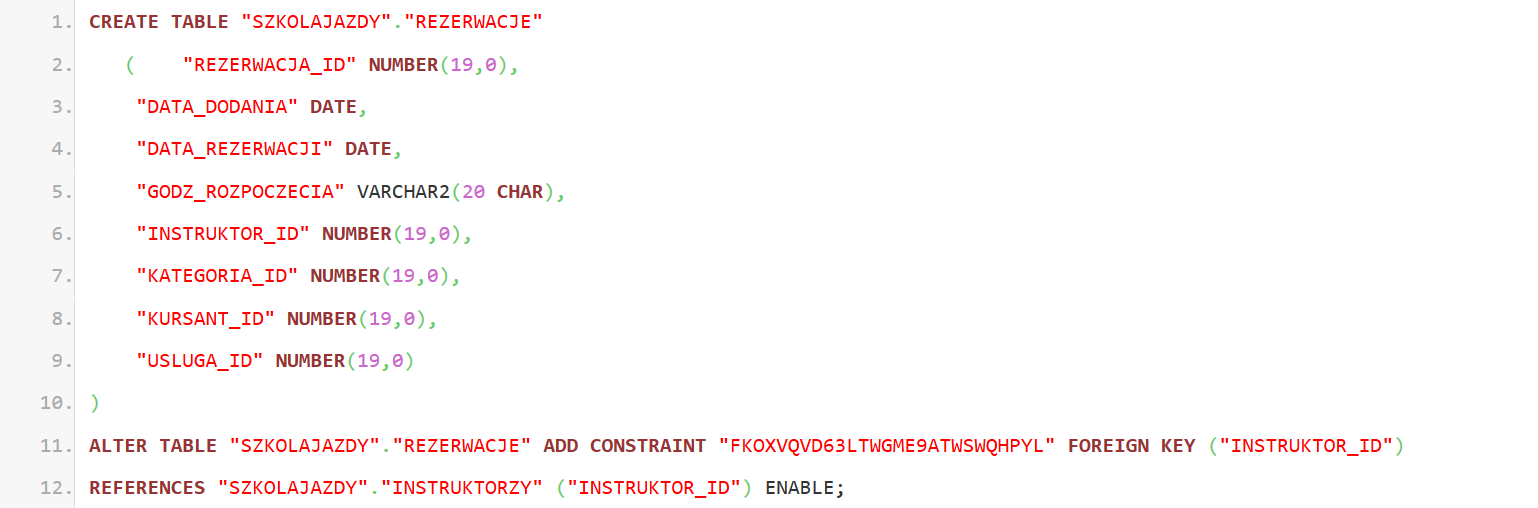


Rysunek 1. Tabela INSTRUKTORZY spełniająca zasadę 1NF

### Druga postać normalna 2NF

Druga postać normalna mówi o tym, że **każda tabela powinna przechowywać dane dotyczące tylko konkretnej klasy obiektów.** Należy wydzielić należy zbiór atrybutów, który jest zależny tylko od klucza głównego. **Wszystkie atrybuty informacyjne (nie należące do klucza), muszą zawierać informacje o elementach tej konkretnej klasy** (encji), a nie żadnej innej. Kolumny opisujące inne obiekty, powinny trafić do właściwych encji, w których te obiekty będziemy przechowywać.

Przykładem 2NF w naszej bazie danych może być to, że tabela instruktorzy nie przechowuje informacji o rezerwacjach, które są wykonane na konkretnego instruktora. Obiekt instruktor i rezerwacja znajdują się w różnych encjach powiązanych ze sobą poprzez klucz obcy (instruktor\_id) wskazujący na klucz główny tabeli INSTRUKTORZY (instruktor\_id).



Rysunek 2. Tabela REZERWACJE przechowująca informację o odpowiadającym jej id instruktora

### Trzecia postać normalna 3NF

Trzecia postać normalne mówi o tym, że **kolumna informacyjna nie należąca do klucza nie zależy też od innej kolumny informacyjnej,** nie należącej do klucza. Czyli każdy atrybut niekluczowy nie zależy funkcyjnie od innego atrybutu niekluczowego.

Przykładem 3NF w naszej bazie danych może być to, że w tabeli REZERWACJE pod usluga\_id kryją się nazwa i cena usługi. Cena usługi zależy bezpośrednio od jej nazwy, zatem gdybyśmy włączyli do tabeli te dwa dodatkowe atrybuty zasada 3NF nie byłaby spełniona. Rozwiązaniem problemu jest utworzenie osobnej tabeli USLUGI, która przechowuje informację o cenie i nazwie pojedynczej usługi.



Rysunek 3. Tabela USŁUGI

# Część praktyczna projektu

## Przedstawienie problemu

Postanowiliśmy zrealizować prosty system **szkoły jazdy**. Projekt będzie realizowany w następujący sposób:

* Połączenie z bazą będzie realizowane przy pomocy aplikacji desktopowej. Dostępne w niej będą dwie formy początkowego logowania – jedna dla użytkownika i druga dla instruktora oraz jedna forma rejestracji – zakładanie nowego konta użytkownika.

### Interfejs użytkownika

* Użytkownik może przeglądać historię rezerwacji (wraz z filtrowaniem rekordów), wyświetlać ich szczegóły (datę, czas trwania, w przypadku kursu – ilość zrealizowanych godzin, ilość brakujących godzin).
* Użytkownik może dodawać nowe rezerwacje do swojego konta.
* Użytkownik może sprawdzać saldo swojego konta.

### Interfejs instruktora

* Instruktor może wyświetlać listę nadchodzących usług jakie zostały u niego zarezerwowane.
* Instruktor ma podgląd na swoich kursantów i ich obecne rezerwacje.
* Instruktor może wygenerować podsumowania swojej pracy za dany okres czasu tj. wygenerować plik ze swoimi danymi oraz rezerwacjami przypisanymi do swojego konta.

### Interfejs administratora

* Administrator może wyświetlić listę rezerwacji i je w pełni modyfikować.
* Administrator może wyświetlić listę instruktorów, dodawać oraz edytować ich konta i nadawać prawo do kategorii.
* Administrator może wyświetlać i edytować konta użytkownika.
* Administrator może wyświetlać raport danego kursanta lub instruktora za dany okres czasu tj. wygenerować plik pokazujący dane osobowe oraz rezerwacje przypisane do danego konta.
* Administrator może wyświetlać listę usług i je modyfikować.
* Administrator może dodawać i edytować płatności za daną rezerwację, które odciążają saldo kursanta.

## Wymagania systemu

* IntelliJ IDEA Ultimate 2018 – do uruchomienia aplikacji
* Oracle SQL Developer 11.2.0
* Oracle Database XE 11.2

## Model danych ERD

Model ER (Entity Relationship Model) – opis teoretyczny relacyjnej bazy danych, który ma na celu opisanie jej za pomocą związków encji.

Diagram ERD (Entity Relationship Diagram) – graficzny odpowiednik modelu ER. Pozwala na zrozumienie struktury danych, przygotowania późniejszej strategii optymalizacji bazy, oraz stanowi podstawową dokumentację systemu przechowywania informacji. Przedstawia on obiekty, o których informacje są istotne w realizacji celów do których tworzona jest baza danych, ich atrybuty oraz związki pomiędzy nimi.

### Identyfikacja zbioru encji wraz z ich atrybutami kluczowymi

W naszym systemie wyodrębniono encje oraz atrybuty jednoznacznie je identyfikujące.

Tabela 1. Encje i identyfikatory

|  |  |
| --- | --- |
| **Encja** | **Klucz główny** |
| *INSTRUKTORZY* | *INSTRUKTOR\_ID* |
| *KATEGORIE* | *KATEGORIA\_ID* |
| *KATEGORIE\_INSTRUKTOROW* | *ID\_WPISU* |
| *KURSANCI* | *KURSANT\_ID* |
| *PLATNOSCI* | *PLATNOSC\_ID* |
| *REZERWACJE* | *REZERWACJA\_ID* |
| *USLUGI* | *USLUGA\_ID* |

### Identyfikacja bezpośrednich zależności między encjami

Tabela 2. Tabela krzyżowa - zależności bezpośrednie pomiędzy encjami

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | *INSTRUKTORZY* | *KATEGORIE* | *KATEGORIE\_*  *INSTRUKTOROW* | *KURSANCI* | *PLATNOSCI* | *REZERWACJE* | *USLUGI* |
| *INSTRUKTORZY* |  |  |  |  |  |  |  |
| *KATEGORIE* |  |  |  |  |  |  |  |
| *KATEGORIE\_*  *INSTRUKTOROW* | x | x |  |  |  |  |  |
| *KURSANCI* |  |  |  |  |  |  |  |
| *PLATNOSCI* |  |  |  | x |  |  |  |
| *REZERWACJE* | x | x |  | x |  |  | x |
| *USLUGI* |  |  |  |  |  |  |  |

Tabela 3. Opis atrybutów tabeli "INSTRUKTORZY"

|  |  |
| --- | --- |
| ***INSTRUKTORZY*** | |
| **Atrybut** | **Opis** |
| *INSTRUKTOR\_ID* | Uniwersalny identyfikator instruktora |
| *CZY\_ADMIN* | Czy instruktor ma uprawnienia admina |
| *DATA\_DODANIA* | Data dodania instruktora do systemu |
| *EMAIL* | Adres e-mail instruktora |
| *GODZ\_ROZPOCZECIA* | Godzina rozpoczęcia pracy instruktora |
| *GODZ\_ZAKONCZENIA* | Godzina zakończenia pracy instruktora |
| *HASLO* | Hasło logowania do systemu |
| *IMIE* | Imię instruktora |
| *NAZWISKO* | Nazwisko instruktora |

Tabela 4. Opis atrybutów tabeli "KATEGORIE"

|  |  |
| --- | --- |
| ***KATEGORIE*** | |
| **Atrybut** | **Opis** |
| *KATEGORIA\_ID* | Uniwersalny identyfikator kategorii prawa jazdy |
| *SYMBOL* | Symbol kategorii prawa jazdy (A,B itp.) |

Tabela 5. Opis atrybutów tabeli "KATEGORIE\_INSTRUKTORÓW"

|  |  |
| --- | --- |
| ***KATEGORIE\_INSTRUKTORÓW*** | |
| **Atrybut** | **Opis** |
| *ID\_WPISU* | Uniwersalny identyfikator wpisu, który wiąże instruktora z odpowiednią kategorią |
| *INSTRUCTOR\_ID* | Uniwersalny identyfikator instruktora |
| *KATEGORIA\_ID* | Uniwersalny identyfikator kategorii prawa jazdy |

Tabela 6. Opis atrybutów tabeli "KURSANCI"

|  |  |
| --- | --- |
| ***KURSANCI*** | |
| **Atrybut** | **Opis** |
| *KURSANT\_ID* | Uniwersalny identyfikator kursanta |
| *DATA\_REJESTRACJI* | Data rejestracji kursanta w systemie |
| *EMAIL* | Adres e-mail kursanta |
| *HASLO* | Hasło logowania do systemu |
| *IMIE* | Imię kursanta |
| *NAZWISKO* | Nazwisko kursanta |
| *PESEL* | Numer PESEL kursanta |
| *PKK* | Numer PKK kursanta |

Tabela 7. Opis atrybutów tabeli "PLATNOSCI"

|  |  |
| --- | --- |
| ***PLATNOSCI*** | |
| **Atrybut** | **Opis** |
| *PLATNOSC\_ID* | Uniwersalny identyfikator płatności |
| *DATA\_PLATNOSCI* | Data wykonania płatności |
| *KURSANT\_ID* | Uniwersalny identyfikator kursanta dokonującego płatności |
| *KWOTA* | Kwota płatności |

Tabela 8. Opis atrybutów tabeli "REZERWACJE"

|  |  |
| --- | --- |
| ***REZERWACJE*** | |
| **Atrybut** | **Opis** |
| *REZERWACJA\_ID* | Uniwersalny identyfikator rezerwacji |
| *DATA\_DODANIA* | Data dodania rezerwacji do systemu |
| *DATA\_REZERWACJI* | Data na którą rezerwacja ma być wykonana |
| *GODZ\_ROZPOCZECIA* | Godzina rozpoczęcia rezerwacji |
| *GODZ\_ZAKONCZENIA* | Godzina zakończenia rezerwacji |
| *INSTRUKTOR\_ID* | Uniwersalny identyfikator instruktora obsługującego rezerwację |
| *KATEGORIA\_ID* | Kategoria prawa jazdy, na którą jest zrobiona rezerwacja |
| *KURSANT\_ID* | Uniwersalny identyfikator kursanta dokonującego rezerwacji |
| *USLUGA\_ID* | Uniwersalny identyfikator usługi, która jest rezerwowana |

Tabela 9. Opis atrybutów tabeli "USLUGI"

|  |  |
| --- | --- |
| ***USLUGI*** | |
| **Atrybut** | **Opis** |
| *USLUGA\_ID* | Uniwersalny identyfikator usługi |
| *CENA* | Cena usługi |
| *NAZWA* | Nazwa usługi |

## Schemat diagramu ERD

### Opis aplikacji w której modelowano schemat

Schemat modelowano w aplikacji Oracle SQL Developer 11.2.0. SQL Developer przeznaczony jest pracy z bazą Oracle w wersji 9.2.0.1 oraz późniejszych. Ma on możliwość podłączenia do innych nie-Oraclowych baz danych, jednak dla ułatwienia bazę szkoły jazdy utworzono w Oracle Application Express.

Edytor programisty wyposażony jest we wsparcie pisania kodu, polegające zarówno na podpowiedziach nazw obiektów, kolumn tabel itp., jak i na możliwości automatycznego wklejania szablonów struktur programowych, takich jak pętle, kursory itp.

SQL Developer umożliwia jednoczesne połączenie do wielu baz danych i pracy z nimi. Połączenia znajdują się w okienku *Connections* po lewej stronie. Umieszczając bazę w okienku połączeń mamy pogląd na strukturę bazy danych i łatwy dostęp do jednostki edytowanej w danej chwili. Okienko *Worksheet* przeznaczone jest do uruchamiania poleceń SQL. Po uruchomieniu napisanego przez nas skryptu otwiera się okienko Script Output w którym widoczny jest wynik naszego polecenia SQL. W menu *Run* znajdują się opcje krokowego uruchamiania programu oraz ustawiania pułapek.

SQL Developer jest darmowy, gwarantuje łatwość instalacji, uruchomienia i przenośność.

### Prezentacja schematu ERD bazy danych

Schemat wygenerowano automatycznie, uprzednio implementując bazę w SQL Developerze.

Obraz zawierający zrzut ekranu

Opis wygenerowany przy wysokim poziomie pewności

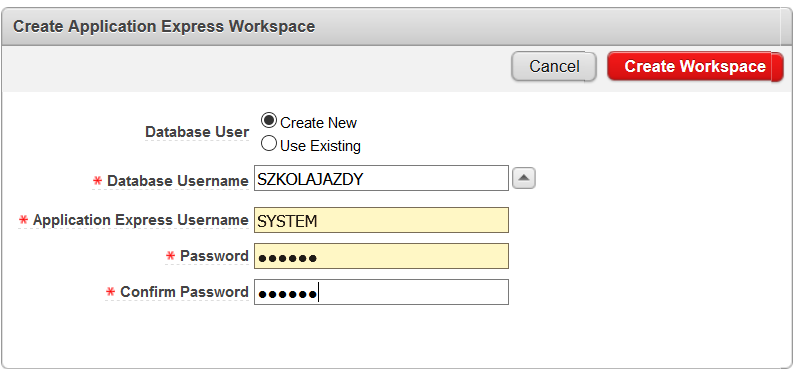
Rysunek 4. Diagram ERD bazy "SZKOLAJAZDY"

## Rozwiązanie problemu

### System bazodanowy

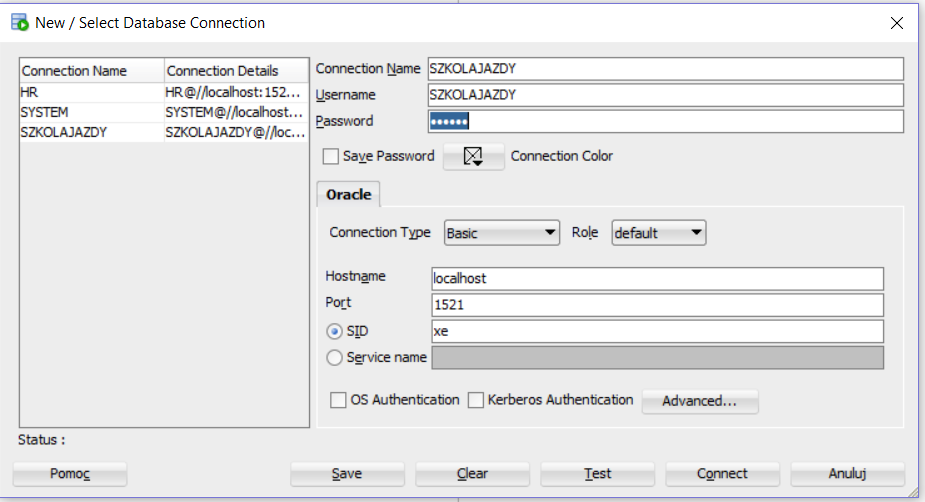
#### Utworzenie bazy danych

Baza została utworzona przy pomocy Oracle Application Express. Po zalogowaniu się na ekranie widoczne jest okienko *Create Application Express Workspace*.



Rysunek 5. Tworzenie nowej bazy danych w Oracle Application Express

Następnie uruchomiono SQL Developer i utworzono nowe połączenie z bazą SZKOLAJAZDY, aby móc rozpocząć pracę na niej. W celu nawiązania połączenia należy kliknąć przycisk *Connect.*

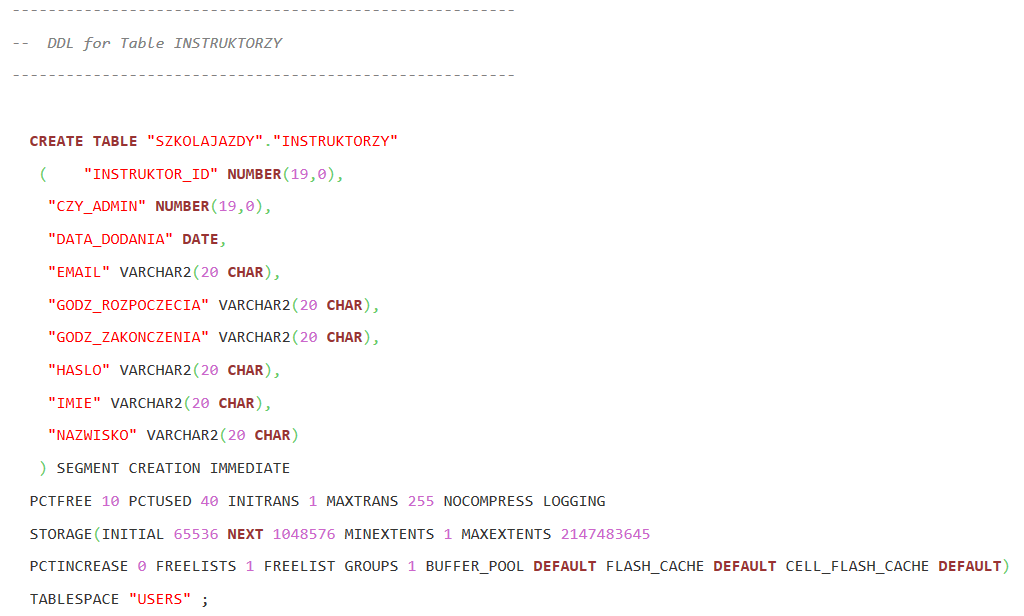


Rysunek 6. Nawiązywanie połączenia z bazą danych w SQL Developer

Po wykonaniu tej operacji w okienku *Connections* po lewej stronie widzimy nowe połączenie z bazą SZKOLAJAZDY.

#### Implementacja obiektów

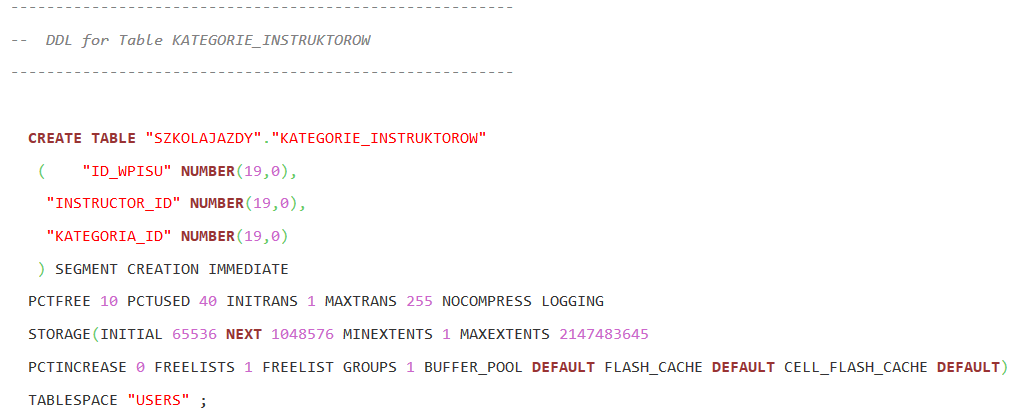
Podstawowymi obiektami w bazie danych są tabele. Do utworzenia tabeli służy zapytanie **CREATE TABLE**. Składnia zapytania: **CREATE TABLE** nazwa tabeli ( kolumna 1 typ\_danych, kolumna 2 typ\_danych, itp.), gdzie „kolumna…” definiuje nazwę kolumny (atrybutu), a typ\_danych definiuje typ danych w danej kolumnie. Na typ danych możemy narzucić ograniczenia używając słowa kluczowego **CHECK** (..) i w nawiasie podając ograniczenia.



Rysunek 7. Tworzenie tabeli "INSTRUKTORZY"



Rysunek 8. Tworzenie tabeli "KATEGORIE"



Rysunek 9. Tworzenie tabeli "KATEGORIE\_INSTRUKTOROW"



Rysunek 10. Tworzenie tabeli "KURSANCI"



Rysunek 11. Tworzenie tabeli "PLATNOSCI"



Rysunek 12. Tworzenie tabeli "REZERWACJE"

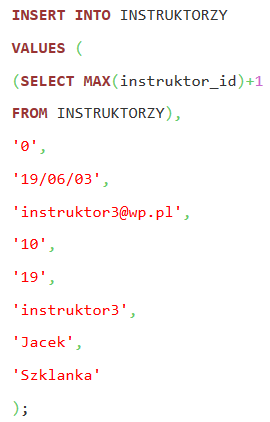
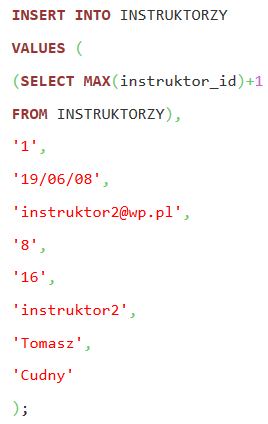
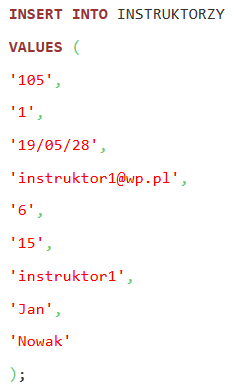


Rysunek 13. Tworzenie tabeli "USLUGI"

#### Wprowadzenie danych

Zapytanie **INSERT** służy do dodania do tabeli nowych rekordów zawierających w odpowiednich kolumnach podane wartości. Składnia zapytania: ***INSERT INTO*** *<nazwa tabeli>* ***VALUES*** *(‘wartość1’,’wartość2’, itp.)*, gdzie **INSERT INTO** oznacza początek operacji wstawiania, w nawiasie za słowem kluczowym **VALUES** definiujemy odpowiednie wartości w takiej kolejności, w jakiej występują one w tabeli.

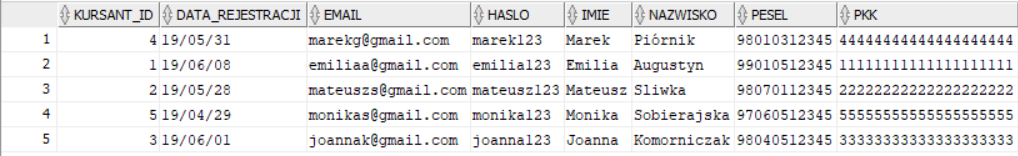
##### Wprowadzenie danych instruktorów



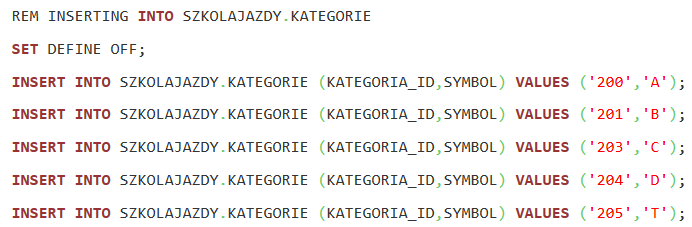
##### Wprowadzenie danych kursantów



W podobny sposób wprowadzono jeszcze trzech kursantów aby tabela **KURSANCI** wyglądała tak:

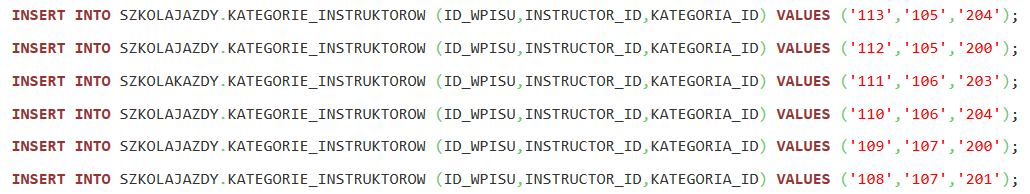


##### Wprowadzanie danych kategorii

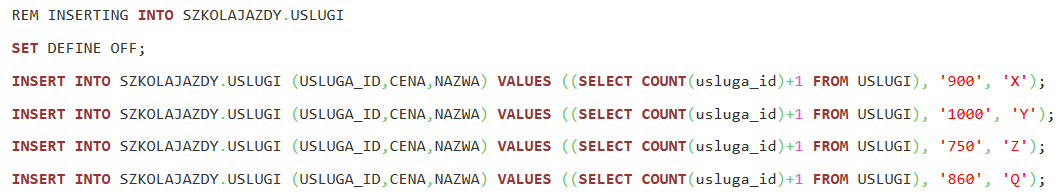


##### Przypisywanie odpowiednich kategorii, które obsługuje instruktor

W bazie każdy instruktor obsługuje dwie różne kategorie praw jazdy:



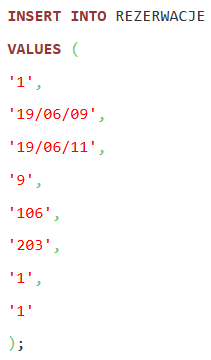
##### Wprowadzenie danych usług



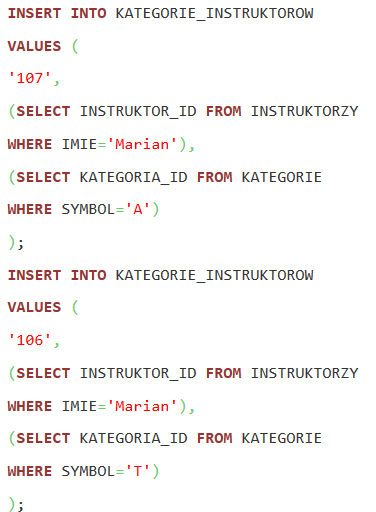
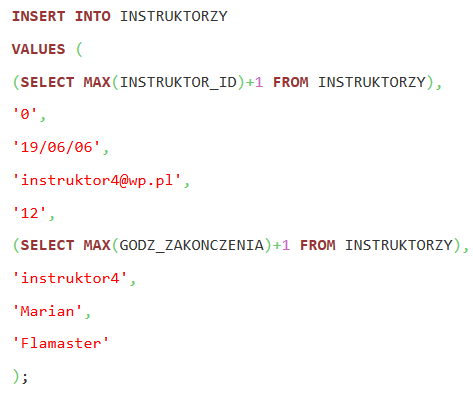
#### Zdefiniowanie w języku SQL poleceń dla realizacji typowych operacji

##### Wstawianie – INSERT

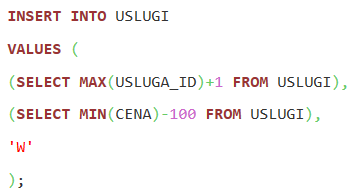
* Wstawienie nowej rezerwacji dla kursanta o identyfikatorze równym 1



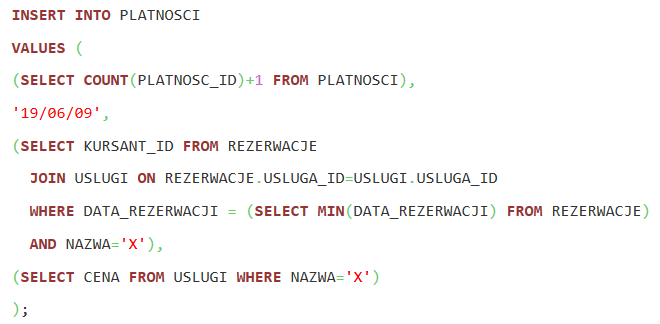
* Wstawianie nowego instruktora, który będzie obsługiwał kategorię prawa jazdy A i T oraz pracował o godzinę dłużej niż najpóźniej pracujący instruktor



* Wstawienie nowej usługi ‘W’, która będzie najtańszą spośród wszystkich usług

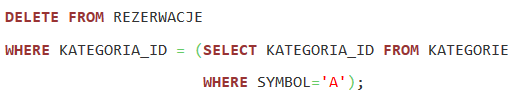


* Wstawienie nowej płatności za kursanta, który ostatnio zarezerwował usługę ‘X’



##### Usuwanie – DELETE

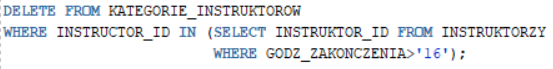
* Usunięcie wszystkich rezerwacji na kategorię ‘A’



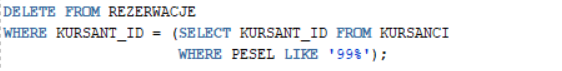
* Usunięcie najpóźniej dodanej płatności



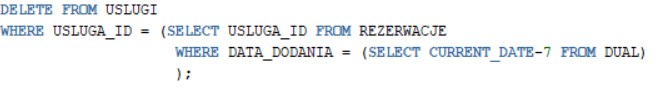
* Usunięcie obsługiwania kategorii ‘A’ i ‘B’ przez instruktorów, którzy pracują po godzinie 16:00



* Usunięcie rezerwacji kursantów, którzy urodzili się w 99’ roku



* Usunięcie usługi, która była zarezerwowana tydzień temu



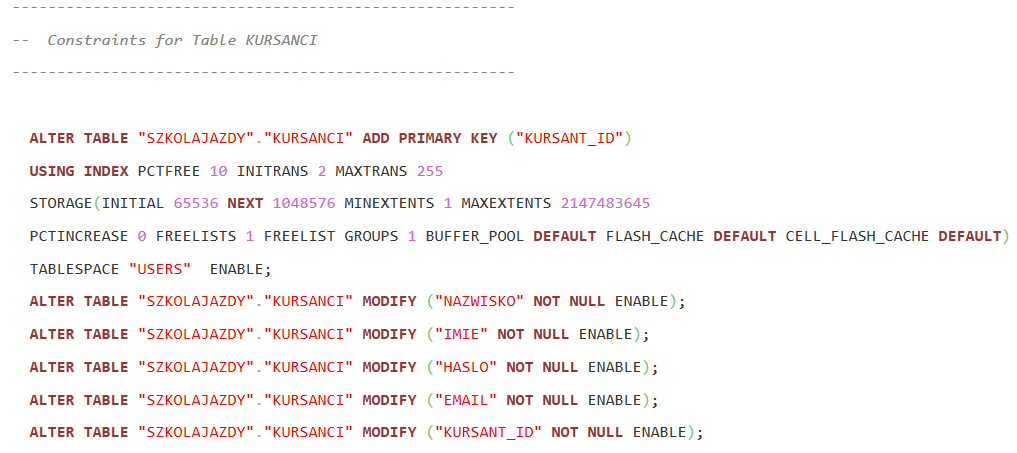
##### Modyfikacja – ALTER/UPDATE

###### Przypisanie klucza głównego do atrybutu i modyfikacja odpowiednich kolumn, aby nie mogły przyjąć wartośći NULL

* Dla tabeli **INSTRUKTORZY**:



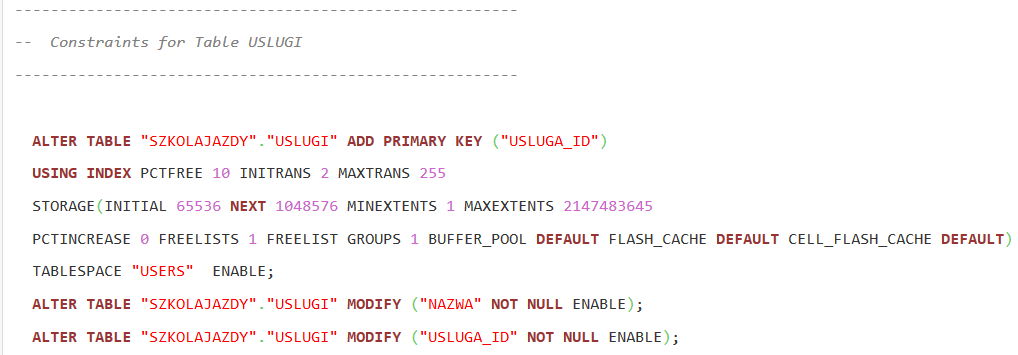
* Dla tabeli **KURSANCI:**



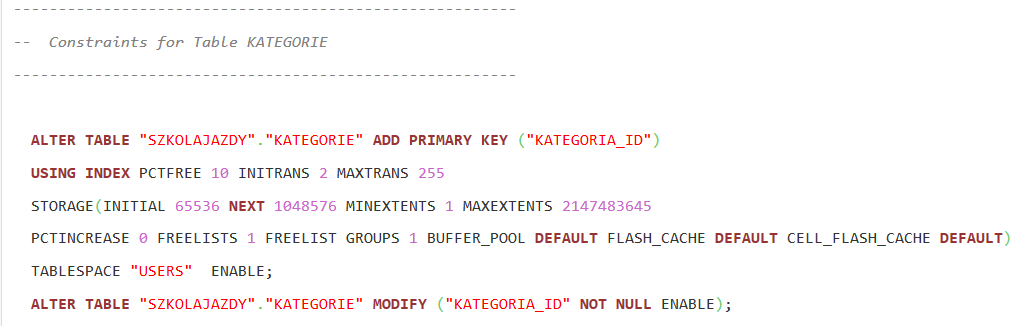
* Dla tabeli **REZERWACJE**:



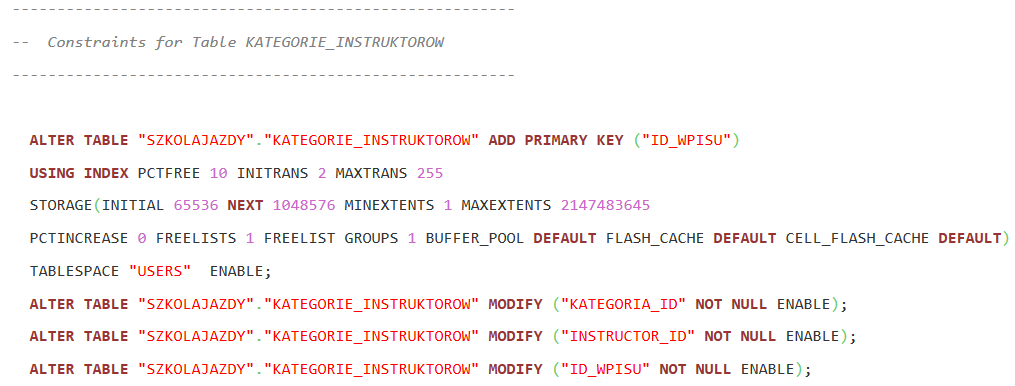
* Dla tabeli **USLUGI**:



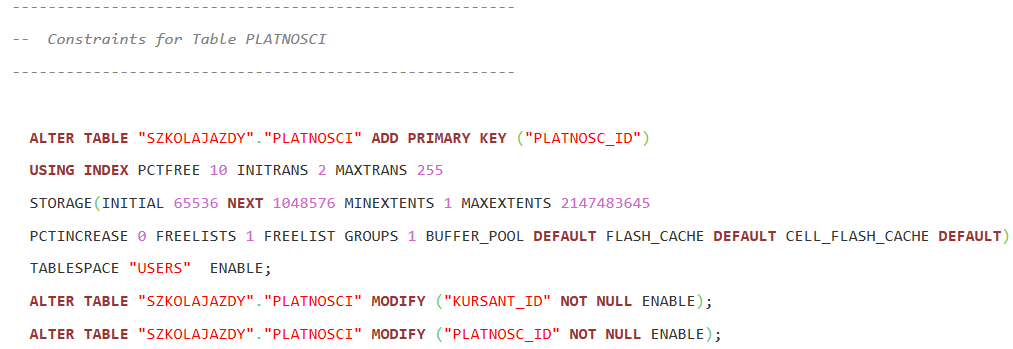
* Dla tabeli **KATEGORIE**:



* Dla tabeli **KATEGORIE\_INSTRUKTOROW**:

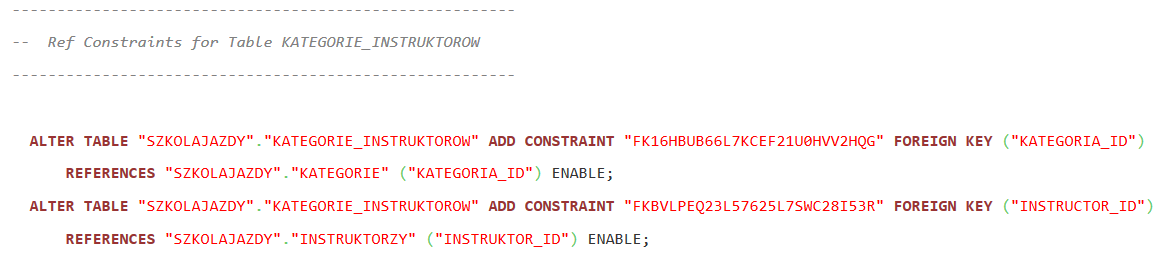


* Dla tabeli **PLATNOSCI**:

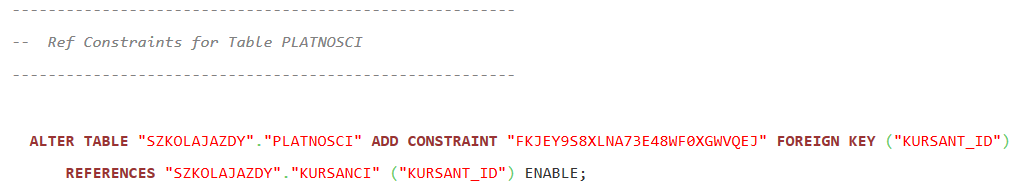


###### Ustawienie kluczy obcych

* W tabeli **KATEGORIE\_INSTRUKTOROW**:



* W tabeli **PLATNOSCI**:

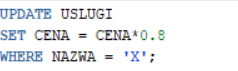


* W tabeli **REZERWACJE**:



###### Modyfikowanie wartości w tabelach

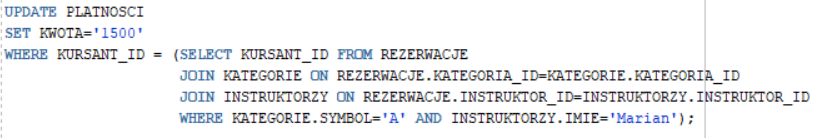
* Przecenienie usługi ‘X’ o 20%



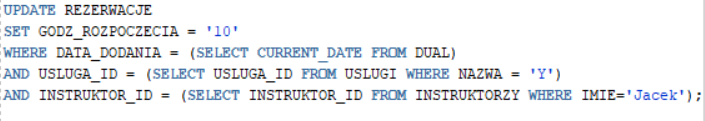
* Zabranie wszystkim instruktorom uprawnień admina



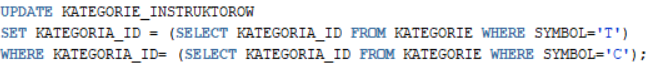
* Zmiana kwoty płatności na 1500,- kursantom, którzy zarezerwowali kurs na kategorię ‘A’ u instruktora Mariana



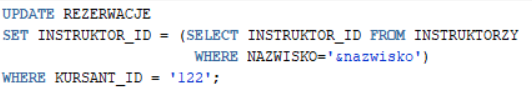
* Zmiana godziny rozpoczęcia rezerwacji na 10:00 kursanta, który zarezerwował dziś usługę Y u instruktora Jacka



* Zmiana kategorii ‘C’ na ‘T’ instruktorom, którzy obsługiwali kategorię ‘C’

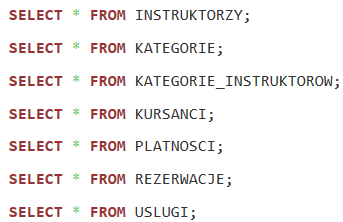


* Zmiana instruktora w rezerwacji kursanta na instruktora o podanym nazwisku

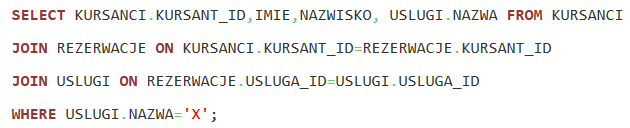


##### Selekcja/prezentacja – SELECT

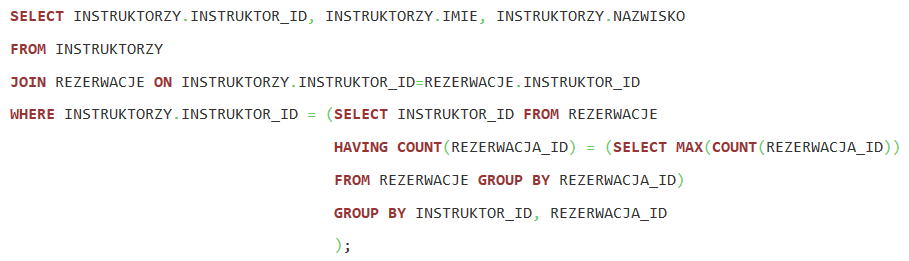
* Wybranie wszystkich rekordów z ich atrybutami z każdej tabeli w bazie, ‘\*’ oznacza wybranie wszystkich:



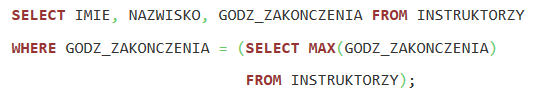
* Wybranie kursantów, którzy zrobili rezerwację na usługę o nazwie ‘X’



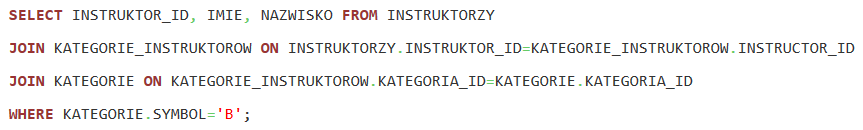
* Wybranie instruktora, na którego zrobione jest najwięcej rezerwacji



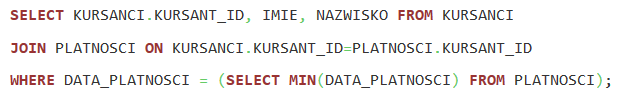
* Wybranie instruktora, który najpóźniej kończy pracę



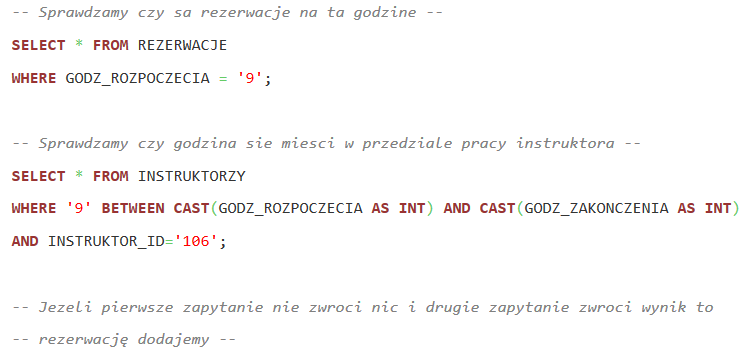
* Wybranie wszystkich instruktorów którzy obsługują kategorie prawa jazdy ‘B’



* Wybranie kursanta, który ostatnio dokonał płatności



* Sprawdzenie, czy można dodać nową rezerwację do instruktora w podanym terminie



### Aplikacja

#### Charakterystyka

Aplikacja została stworzona w języku Java z wykorzystaniem framework’a Hibernate. Hibernate zapewnia translację danych pomiędzy relacyjną bazą danych a światem obiektowym. Opiera się na wykorzystaniu opisu struktury danych za pomocą języka XML, dzięki czemu można rzutować obiekty, stosowane w obiektowych językach programowania, takich jak Java bezpośrednio na istniejące tabele bazy danych.

Aplikacja jest okienkowa, posiada ona trzy interfejsy: użytkownika, instruktora i administratora. W zależności od uprawnień mogą oni wykonywać różne operacje, dokładnie zostało to opisane w podpunkcie *Przedstawienie problemu*. (Ctrl + kliknięcie śledzi łącze).

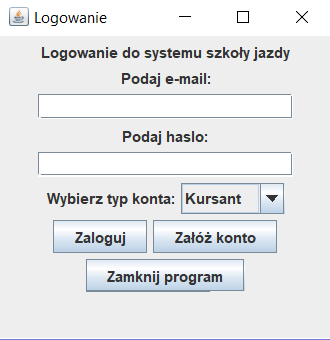
#### Implementacja

Aplikacja została zaimplementowana w środowisku IntelliJ IDEA. Podłączono do niej bazę, którą stworzono wcześniej.

#### Opis formularzy i operacji – najważniejsze wybrane operacje

##### Panel logowania

Na początku po uruchomieniu się aplikacji widzimy okienko logowania. Kluczowa jest tutaj funkcja *actionPerformed*, która za parametr przyjmuje wybraną przez nas opcję: logowania, założenia nowego konta lub zamknięcia programu.



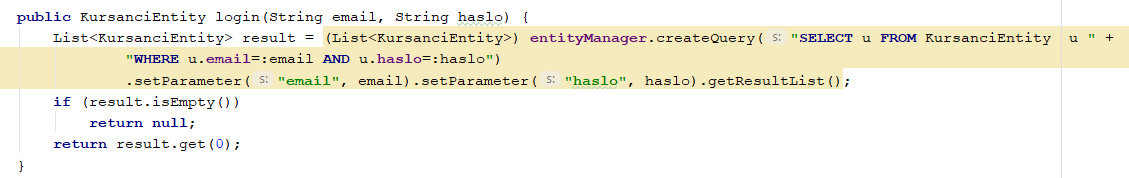
Rysunek 14. Okienko logowania



Rysunek 15. Kod programu

###### Logowanie do panelu kursanta

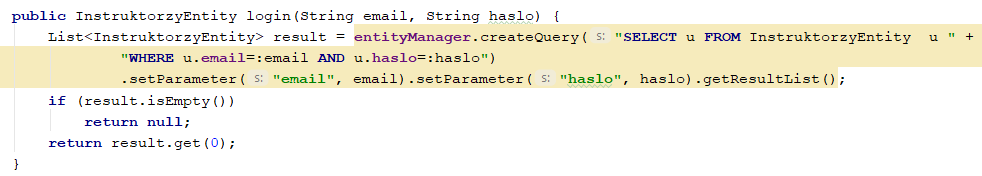
Jeżeli wybraliśmy opcję kursant, wykonuje się funkcja *login:*



Wykonuje ona zapytanie do bazy, które zwraca czy taki kursant znajduje się w bazie. Jeśli nie, funkcja zwraca *null*.

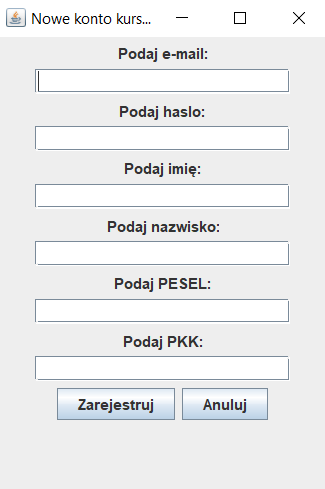
###### Logowanie do panelu instruktora

Podobnie wykonuje się funkcja *login*:

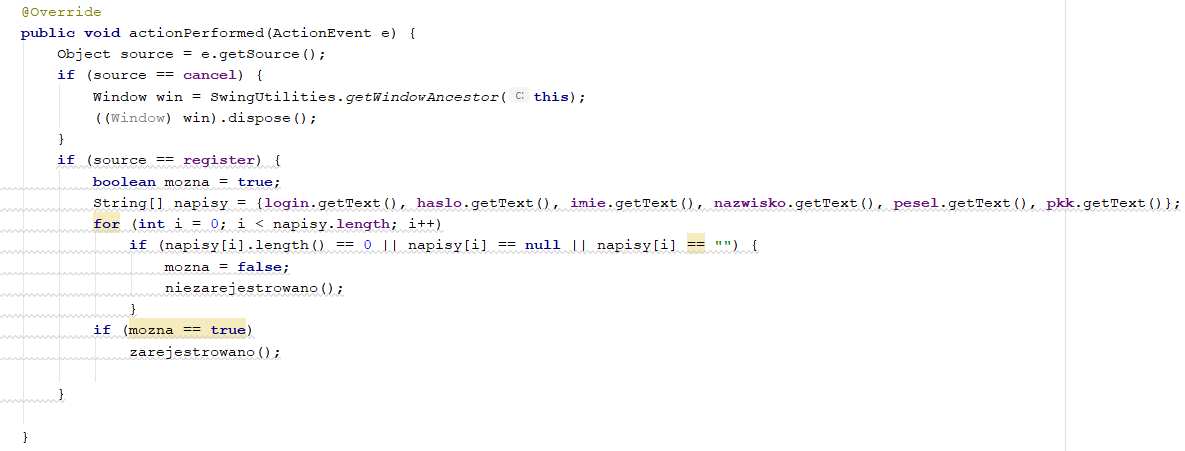


Wykonuje ona analogiczne do powyższego zapytanie do bazy, czy taki instruktor się w niej znajduje.

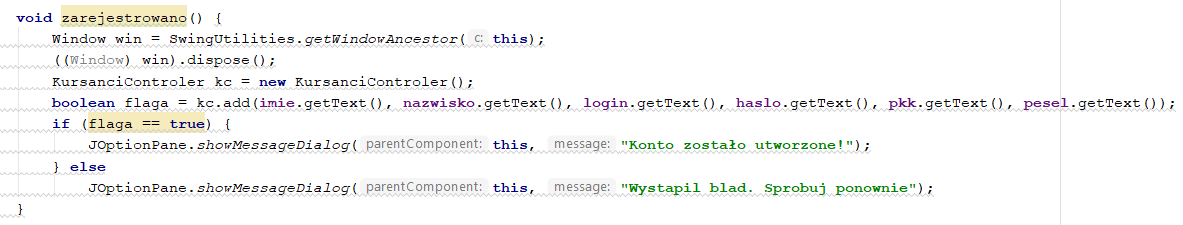
###### Rejestracja



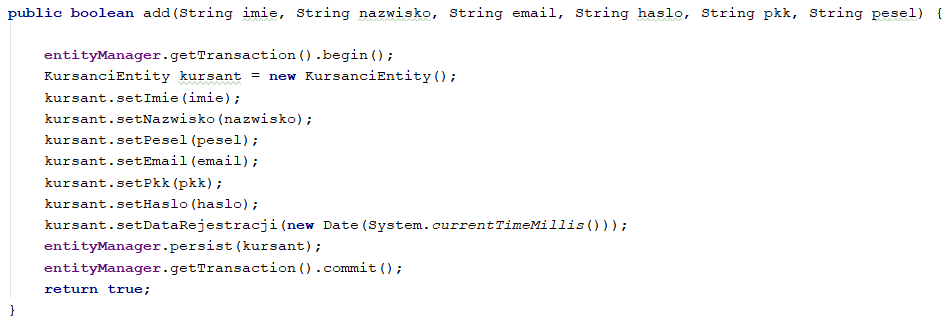
Rysunek 16. Okienko rejestracji

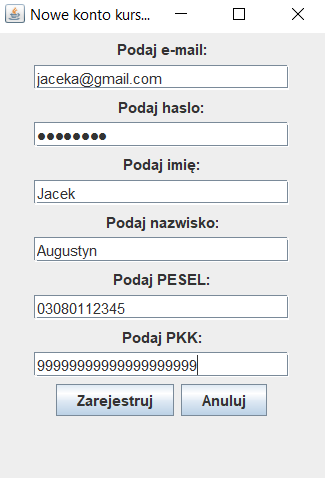


Jeżeli chcemy zarejestrować nowego kursanta, program sczytuje teksty z kolejnych pól, i sprawdza czy wszystkie zostały wypełnione. Jeżeli tak, tworzymy nowego kursanta:

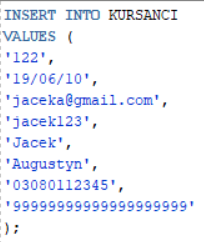


Następnie dodajemy go do bazy poprzez funkcję *add:*



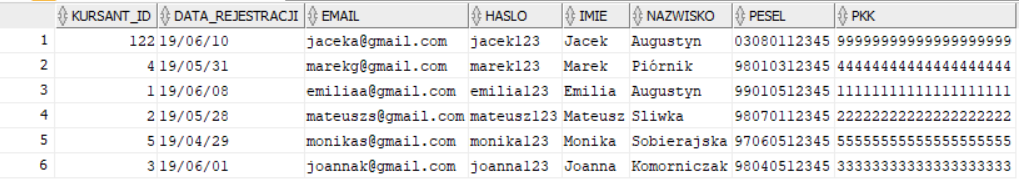


Rysunek 17. Przykładowe dane do rejestracji nowego kursanta

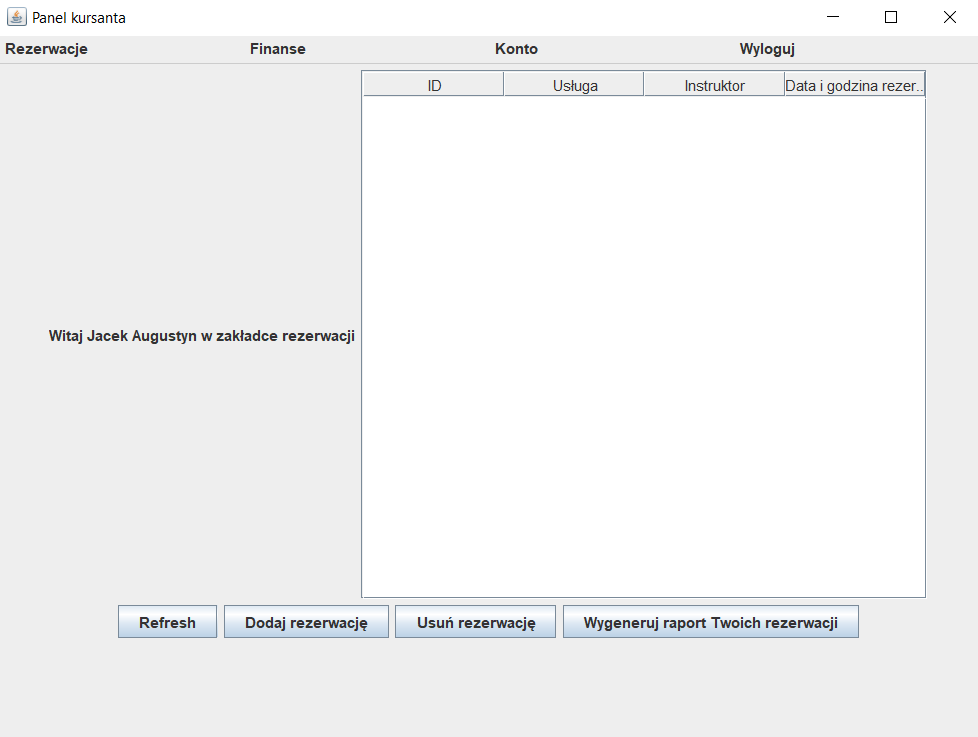


Rysunek 18. Zapytanie SQL odpowiadające funkcji add

Po wykonaniu tej funkcji, tabela **KURSANCI** uzupełnia się o nowego kursanta:

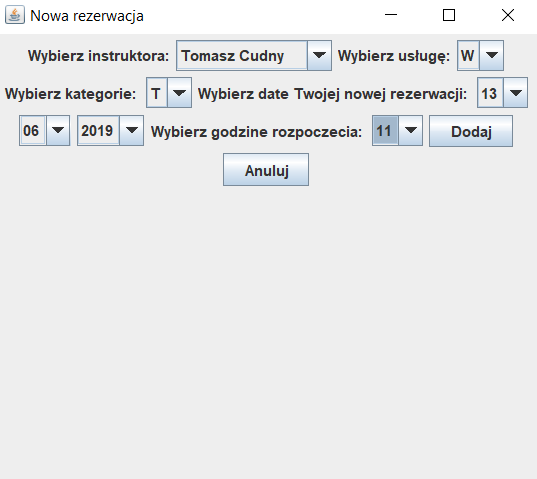


##### Dodawanie rezerwacji przez kursanta



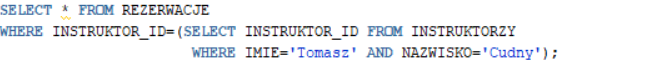
Rysunek 19. Panel obsługi kursanta

Nasz użytkownik nie ma jeszcze żadnej rezerwacji. Aby dodać rezerwację klikamy przycisk *Dodaj rezerwację*



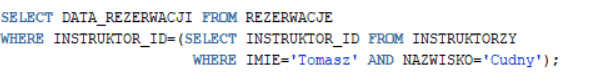
Rysunek 20. Dodawanie rezerwacji

Kiedy uzupełnimy pola odpowiadającymi nam danymi i klikniemy przycisk *Dodaj,* wywołuje się funkcja która sprawdza, czy w danym czasie instruktor ma wolny termin. Wykonuje ona zapytanie do bazy danych:



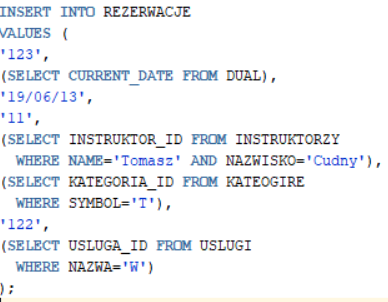
Rysunek 21. Pobranie wszystkich rezerwacji na danego instruktora

Następnie funkcja w pętli dodaje wszystkie daty rezerwacji na instruktora do dat, które są zajęte. W tym celu funkcja musi zapytać bazę o daty wszystkich rezerwacji na instruktora:



Rysunek 22. Pobranie dat rezerwacji na danego instruktora

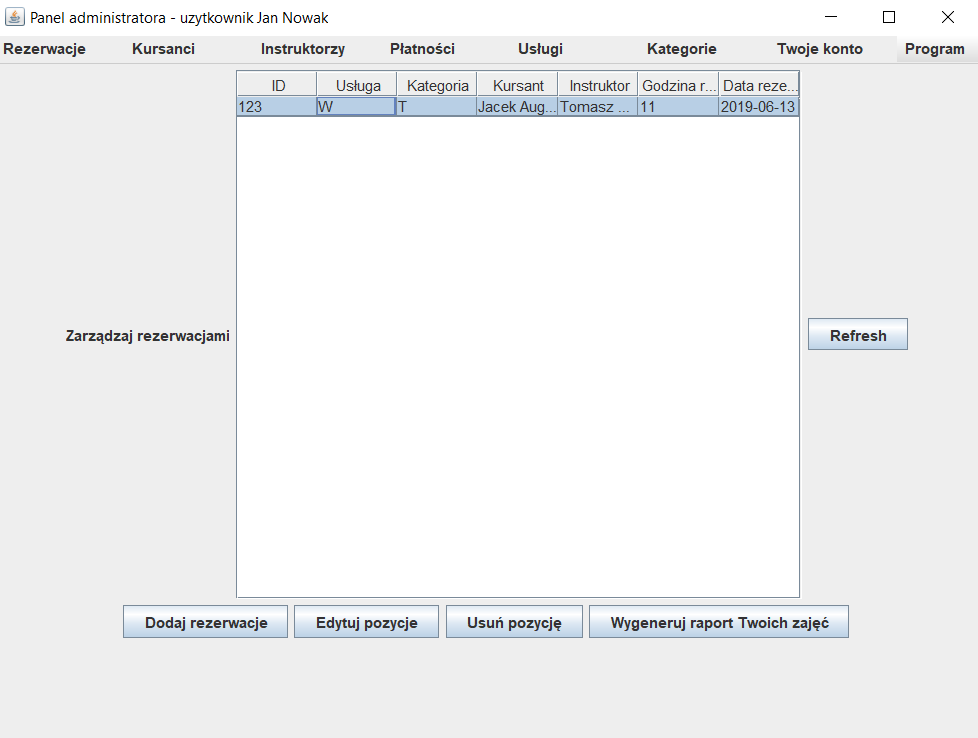
Analogicznie funkcja postępuje z godzinami rezerwacji. Następnie sprawdza, czy podana przez nas data i godzina znajdują się na listach. Jeśli nie, rezerwacja zostaje dodana do bazy danych.



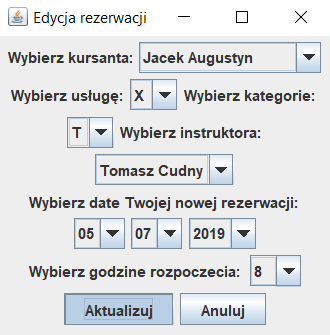
Rysunek 23. Dodawanie rezerwacji do bazy danych

##### Edycja rezerwacji

Logując się jako instruktor, który ma uprawnienia admina, możemy edytować wszystkie rezerwacje jakie są w systemie. W tym celu zaznaczamy pożądaną rezerwację i klikamy przycisk *Edytuj pozycje*:

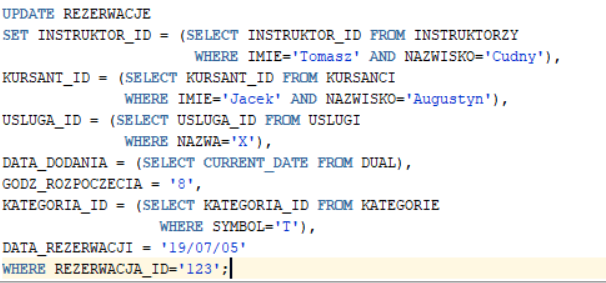


Rysunek 24. Panel obsługi instruktora-admina



Rysunek 25. Okienko edycji rezerwacji

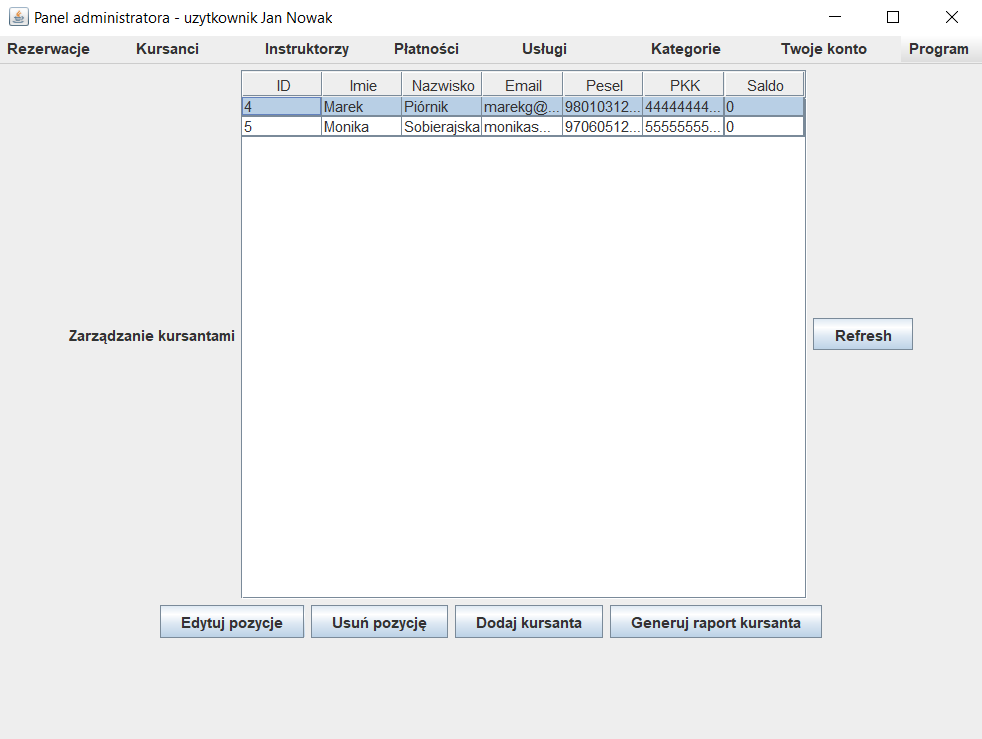
Kiedy wybierzemy nowe dane, funkcja aktualizująca *update* rezerwację wykonuje zapytanie do bazy:



Następnie odświeża listę rezerwacji, pobierając z bazy zaktualizowaną listę rezerwacji dla danego instruktora:

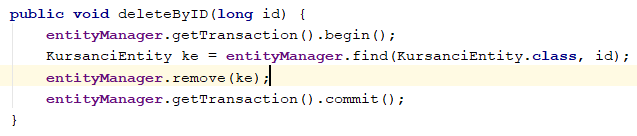


##### Usuwanie kursanta po ID



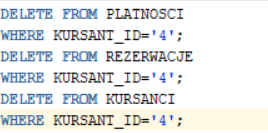
Rysunek 26. Panel obsługi instruktora-admina

Aby usunąć kursanta po ID, zaznaczamy odpowiednią pozycję i klikamy przycisk *Usuń pozycję.* Pobierane jest wtedy ID kursanta i wywołuje się funkcja *deleteByID*, która usuwa kursanta kaskadowo razem ze wszystkimi jego rezerwacjami i płatnościami.



Rysunek 27. Funkcja usuwająca kursanta

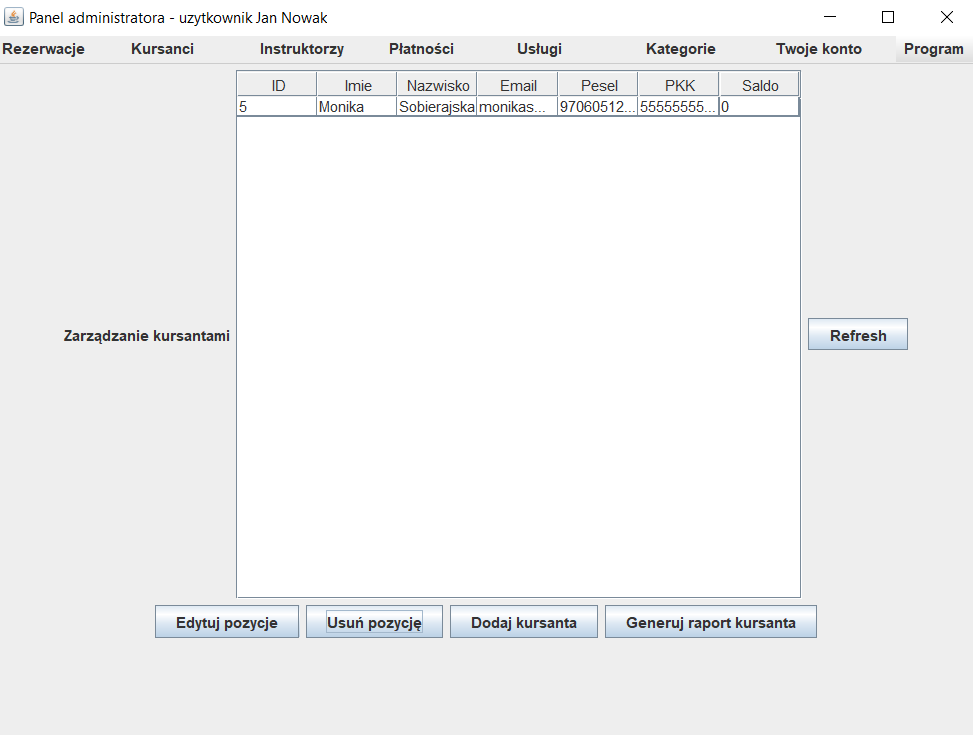
Funkcja *deleteByID* wykonuje poniższe zapytanie do bazy danych:



Lista kursantów odświeża się poprzez funkcję *refreshList,* która wykonuje zapytanie:



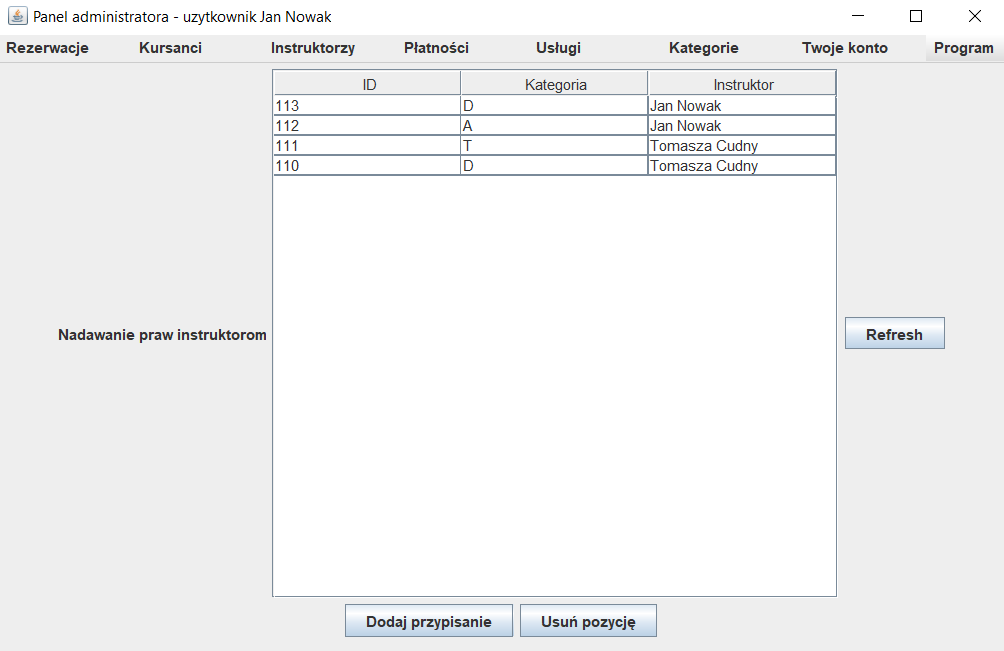
W wyniku tych działań lista kursantów wygląda teraz tak:



Rysunek 28. Lista kursantów po usunięciu kursanta o id = 4

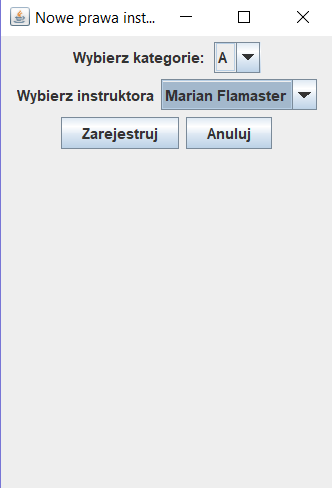
##### Przypisywanie kategorii instruktorom

Logując się jako instruktor przechodzimy do zakładki *Kategorie*. Wyświetla nam się lista instruktorów z przyporządkowanymi im kategoriami.



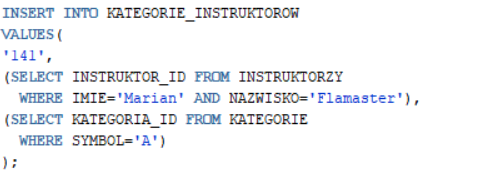
Rysunek 29. Panel obsługi instruktora

Możemy dodać nowe przypisanie klikając przycisk *Dodaj przypisanie*.

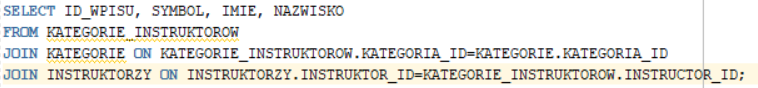


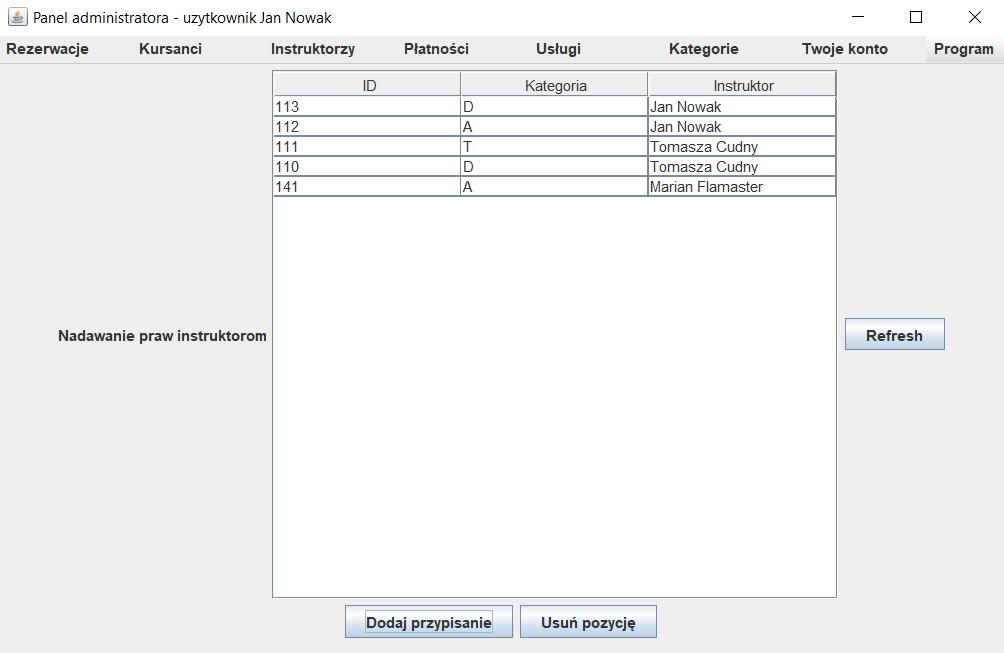
Rysunek 30. Przypisywanie kategorii do instruktora

Kiedy klikniemy przycisk *Zarejestruj,* funkcja *add* dodaje do tabeli **KATEGORIE\_INSTRUKTOROW** podane przez nas „połączenie”. **ID\_WPISU** generuje się automatycznie. Dodanie nowego „połączenia” wykonuje się poprzez poniższe zapytanie:



Po dodaniu nowego połączenia lista odświeża się poprzez funkcję *refreshList,* która wykonuje zapytanie:

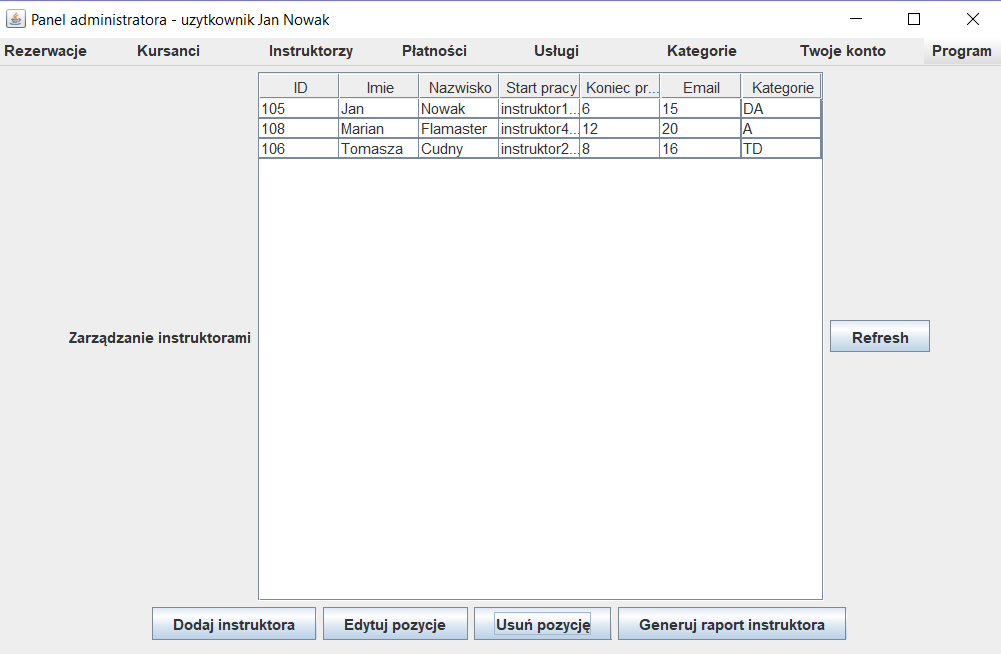




Rysunek 31. Kategorie instruktorów po dodaniu nowego połączenia

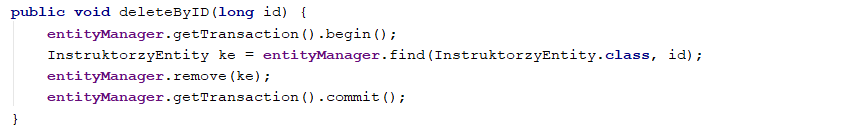
##### Usuwanie instruktora

Aby usunąć instruktora, logujemy się jako instruktor z prawami admina i przechodzimy do zakładki *Instruktorzy*:



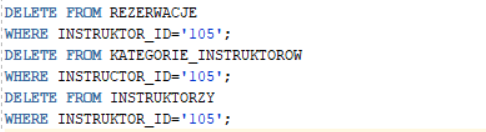
Rysunek 32. Panel obsługi instruktora

Wybieramy instruktora, którego chcemy usunąć i klikamy przycisk *Usuń pozycję*. Wywołuje się wtedy funkcja *deleteByID,* która usuwa instruktora kaskadowo razem ze wszystkimi jego rezerwacjami i kategoriami (podobnie jak przy usuwaniu kursanta).

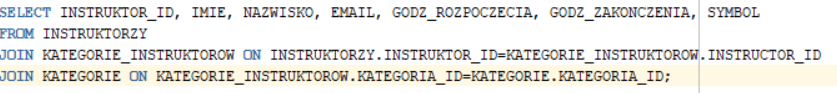


Rysunek 33. Funkcja usuwająca instruktora

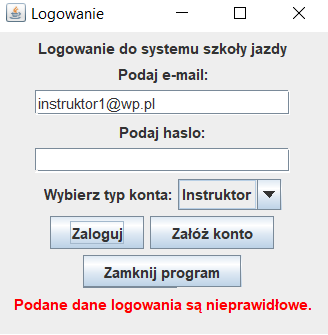
Usuwanie instruktora dokonuje się poprzez wywołanie zapytań:



Następnie lista instruktorów odświeża się poprzez funkcję *refreshList*, która wykonuje zapytanie:



Po usunięciu instruktora możemy łatwo zauważyć, że nie możemy się na niego już zalogować:



Rysunek 34. Błąd logowania - brak instruktora w bazie

## Podsumowanie

### Założenia

Podczas pracy nad produkcją interfejsu zdecydowaliśmy się na małą zmianę w architekturze zależności między użytkownikiem a płatnościami. Zamiast przypisywać płatność do każdej jego rezerwacji z osobna zdecydowaliśmy się na powiązanie płatności bezpośrednio z kontem użytkownika tj. płatność przypisywana jest do ID użytkownika (klucz obcy). Pozwala to na sprawniejsze operowanie interfejsem i szybsze przypisywanie wpłat (nie trzeba szukać konkretnej rezerwacji, wystarczy wskazać użytkownika, który dokonuje wpłaty). Obciążenia natomiast naliczane są automatycznie dzięki powiązaniu rezerwacji z tabelą usług (każda z usług ma swoją cenę). Po przeliczeniu rezerwacji oraz wpłat uzyskujemy saldo kursanta.

Reszta założeń została spełniona wg. opisu problemu w podpunkcie *Przedstawienie problemu* (Ctrl+kliknięcie śledzi łącze)*.*

### Wnioski

Baza danych Oracle w wersji 11 nie wspiera auto inkrementacji ID przy przypisywaniu ich do nowych rekordów. Aby rozwiązać ten problem, generowano w zapytaniach nowe ID poprzez funkcje COUNT(), albo opierano się na mechanizmie Hibernate. Mechanizm ten generuje listę zbiorczych ID dla wszystkich encji, przez co w całej bazie obowiązuje jedna, ogólna lista ID, a nie osobne listy dla każdej encji.

Podczas uzupełniania bazy danych należy uważać na zgodność typów i możliwość kolizji pojedynczych rekordów, np. nakładania się godzin rezerwacji. Rozwiązano to generując szereg zapytań (opisane w punkcie Selekcja/prezentacja – SELECT). Na podstawie tych zapytań napisano funkcję, która podczas dodawania nowych rezerwacji przy pomocy interfejsy sprawdza ich poprawność i dopiero wtedy wykonuje transakcję z bazą danych.

Chęć stworzenia interfejsu zgodnego z zasadą ‘userfriendly’ zmusiła nas do sporządzenia kilku dodatkowych zapytań, które np. zamiast „suchego” ID instruktora zwracało jego imię i nazwisko, czego przykładem jest formularz dodawania rezerwacji w panelu kursanta/administratora.

# Literatura

*[1] https://www.samouczekprogramisty.pl/wstep-do-relacyjnych-baz-danych/#czym-jest-baza-danych*

*[2] http://www.metal.agh.edu.pl/~regulski/bd-podyp/00-wyklady/05\_model\_relacyjny.pdf*

*[3] https://www.sqlpedia.pl/relacyjne-bazy-danych-pojecia-podstawowe/*

*[4] https://www.sqlpedia.pl/projektowanie-i-normalizacja-bazy-danych/*

*[5] https://zeszyt.jedlikowski.com/2014-10-27/sbd/postulaty-codda-dotyczace-relacyjnych-baz-danych-rodzaje-kluczy-zasady-dobierania-kluczy-powiazania-pomiedzy-relacjami/*

*[6] https://www.w3schools.com/sql/*