Politechnika Wrocławska

Wydział Informatyki i Telekomunikacji

Bazy Danych – Projekt

„SZKÓŁKA PŁYWACKA”

Autorzy:

Stanisław Strauchold 259142

Sofiia Hashemi Nezhad 257730

Prowadzący:

Mgr inż. Adam Włodarczyk

1. Wstęp

W poniższej sekcji znajduje się wstęp teoretyczny do aplikacji bazodanowej. Znajduje się w nim opis świata rzeczywistego, opis wymagań funkcjonalnych oraz diagram przypadków użycia wraz z objaśnieniem zastosowanych notacji i symboli.

1.1. Opis słowny systemu – opis „świata rzeczywistego”

Aplikacja bazodanowa obsługuje szkółkę pływacką. Szkółka oferuje różnego rodzaju zajęcia, zarówno dla dzieci, jak i dla dorosłych. W skład oferty wchodzą między innymi: nauka pływania dla dzieci i dorosłych, doskonalenie pływania, zajęcia „baby swim”, czy skoki do wody. Na zajęcia klient może zapisać siebie lub dziecko. Warto zaznaczyć, iż na zajęcia „baby swim” jako uczestnik wpisany jest klient. Lekcje prowadzone są na różnych basenach na terenie Łodzi. Każda lokalizacja charakteryzuje się trochę inną ofertą zajęciową, na przykład zajęcia ze skoków do wody, ze względu na warunki, nie są możliwe do przeprowadzenia na każdym basenie. W skład pracowników firmy wchodzą właściciele oraz instruktorzy. Dzięki zastosowaniu aplikacji bazodanowej, możliwe jest sprawne zarządzanie grupami zajęciowymi, instruktorami, zapisami czy analizą zysków. Do tego baza pozwala na szybkie znalezienie informacji dotyczących konkretnego klienta.

Oprócz zajęć na basenach, szkółka pływacka prowadzi też sklep ze sprzętem sportowym oraz odzieżą do pływania, w którym zakupów mogą dokonywać klienci.

Do tego firma organizuje półkolonie i obozy sportowe w trakcie ferii zimowych i wakacji. Uczestnikami obozów mogą być dzieci klientów.

1.2. Wymagania funkcjonalne

Instruktor:

- przeglądanie danych osób zapisanych do grupy zajęciowej, do której dany instruktor jest przypisany

Właściciel:

- przeglądanie i modyfikacja danych klientów(w tym dodawanie, usuwanie)

- przeglądanie i modyfikacja danych instruktorów(w tym dodawanie, usuwanie)

- przeglądanie i modyfikacja grup zajęciowych(w tym dodanie, usunięcie grupy lub dodanie, usunięcie klienta do/z grupy)

- modyfikacja cennika usług - przeglądanie statystyk finansowych

Administrator:

- przypisanie użytkownikowi roli właściciela

- posiada również funkcjonalności właściciela

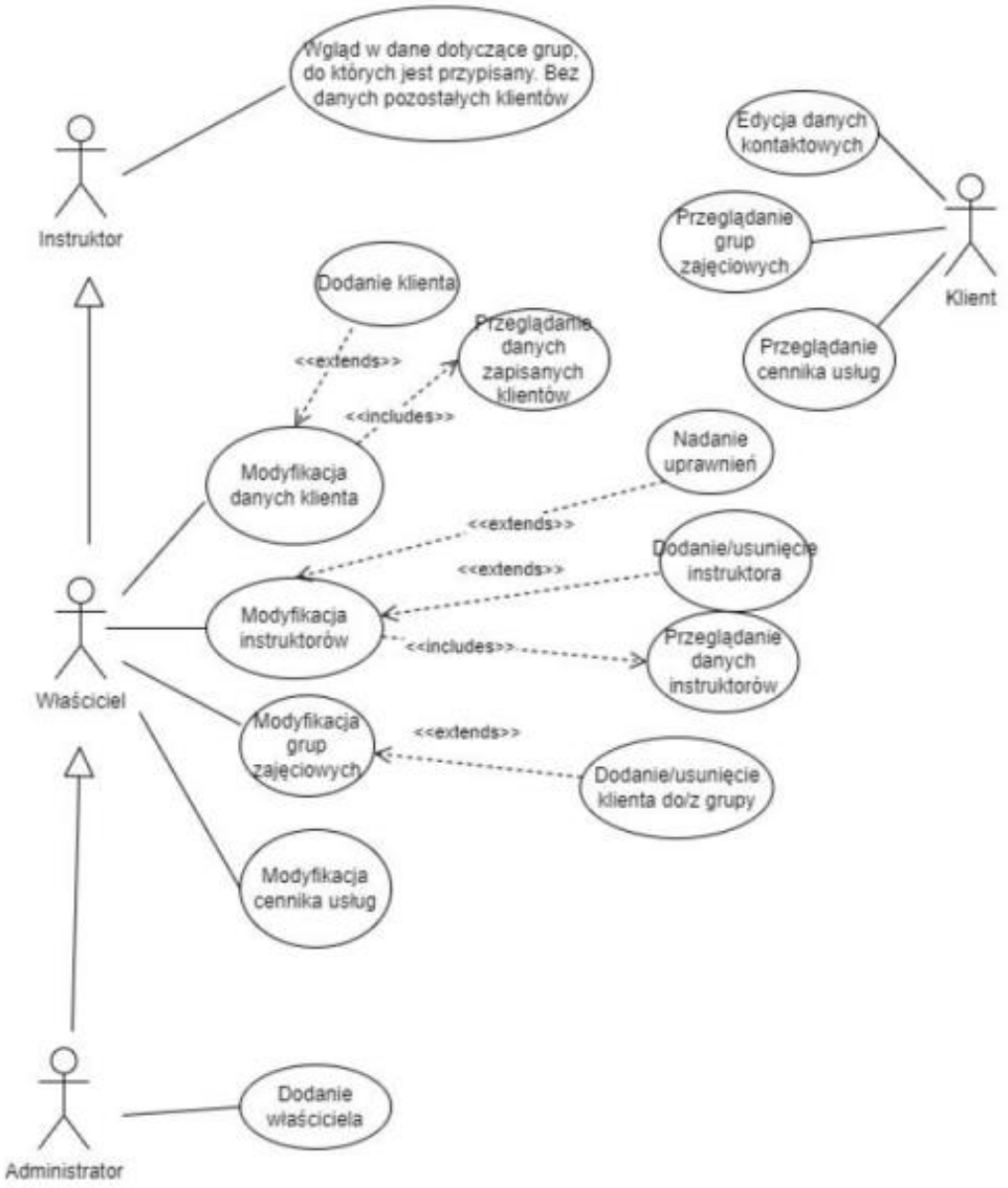
Klient:

- edycja swoich danych kontaktowych

- przeglądanie grup zajęciowych

- przeglądanie cennika usług

1.3. Specyfikacja wymagań funkcjonalnych za pomocą diagramu przypadków użycia



Rys. 1. Diagram przypadków użycia

Opis elementów diagramu:

Aktor – symbolizowany przez rysunek ludzika. Reprezentuje rolę przypisaną użytkownikowi. Każdego aktora charakteryzują inne wymagania funkcjonalne.

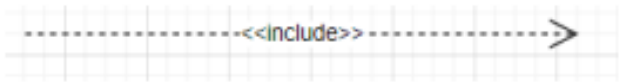
Strzałki łączące aktorów przedstawiają relację dziedziczenia. Dla przykładu Administrator posiada wszystkie funkcjonalności Właściciela, a do tego ma też swoje własne, jak „Dodanie właściciela".

Pojedyncze dymki oznaczają akcje, które może wykonywać aktor. Relacje między tymi akcjami Opisane są dwoma rodzajami przerywanych linii zakończonych strzałkami: extends oraz include.

Strzałka extends:

Strzałka ta oznacza, że dana akcja rozszerza inną akcję. Dla przykładu w naszym diagramie akcja „Dodanie/usunięcie klienta do/z grupy" jest rozszerzeniem akcji „Modyfikacja grup zajęciowych". Grot strzałki skierowany jest w kierunku akcji, która jest rozszerzana.

Strzałka include:



Strzałka ta oznacza, że dana akcja zawiera się w opisywanej funkcjonalności. Dla przykładu w naszym diagramie akcja „Przeglądanie danych instruktorów" zawiera się w akcji „Modyfikacja grup zajęciowych". Grot strzałki skierowany jest w kierunku akcji, która jest zawarta w akcji u źródła strzałki.

2. Projekt bazy danych

W poniższej sekcji znajdują się elementy konieczne do zaprojektowania bazy danych, tj. identyfikacja związków encji, analiza liczby instancji dla encji, modele logiczny i fizyczny.

2.1. Identyfikacja diagramu związków encji na podstawie analizy scenariuszy przypadków użycia

Klienci (id_klienta, imie_klienta, nazwisko_klienta, telefon_klienta, pesel, email_klienta,)

Tabela ta przechowuje dane osobowe i kontaktowe klientów, czyli dorosłych uczestników zajęć, oraz rodziców dzieci uczęszczających na lekcje

Uczestnicy_zajec (id_dziecka, id_klienta, pesel, data_urodzenia, imie_dziecka, nawisko_dziecka)

Tabela ta przechowuje dane osobowe niepełnoletnich uczestników zajęć.

Zajecia (id_zajec, nazwa_zajec, max_ilosc_osob, aktualna_ilosc_osob, id_basenu, cena_zajec, id_instruktora)

Tabela ta przechowuje informacje o wszystkich zajęciach prowadzonych przez szkółkę – godzinę, miejsce, instruktora, cenę oraz aktualny stan zapełnienia grupy

Zapisani_uczestnicy (id_dziecka, id_zajec, id_klienta)

Tabela ta jest łącznikiem pomiędzy tabelami Klienci, Uczesnticy_zajec oraz Zajecia. Jej zadaniem jest stworzenie relacji wiele-do-wielu pomiędzy tabelami Klienci, Uczestnicy_zajec a tabelą Zajecia.

Baseny (id_basenu, nazwa_basenu, cena_toru_h)

Tabela przechowuje informacje o basenach, na których odbywają się zajęcia

Instruktorzy (id_instruktora, imie_instruktora, nazwisko_instruktora, stawka_za_zajecia)

Tabela przechowuje podstawowe dane osobowe instruktorów. Do tego każdy rekord zawiera informacje o stawce wypłacanej każdemu z instruktorów za godzinę pracy.

Turnusy (id_turnusu, data_rozpoczecia, data_zakonczenia, cena_turnusu)

Tabela przechowuje podstawowe informacje na temat turnusów obozów letnich i półkolonii organizowanych przez szkółkę pływacką.

Polkolonie_uczestnicy (id_uczestnika_polkolonii, id_klienta, id_dziecka, id_turnusu)

Tabela przechowuje dane osobowe uczestników zapisanych na półkolonie oraz informację, na który turnus dany uczestnik jest zapisany.

Oboz_uczestnicy (id_uczestnik_obozu, id_klienta, id_dziecka, id_turnusu)

Tabela przechowuje dane osobowe uczestników zapisanych na półkolonie oraz informację, na który turnus dany uczestnik jest zapisany.

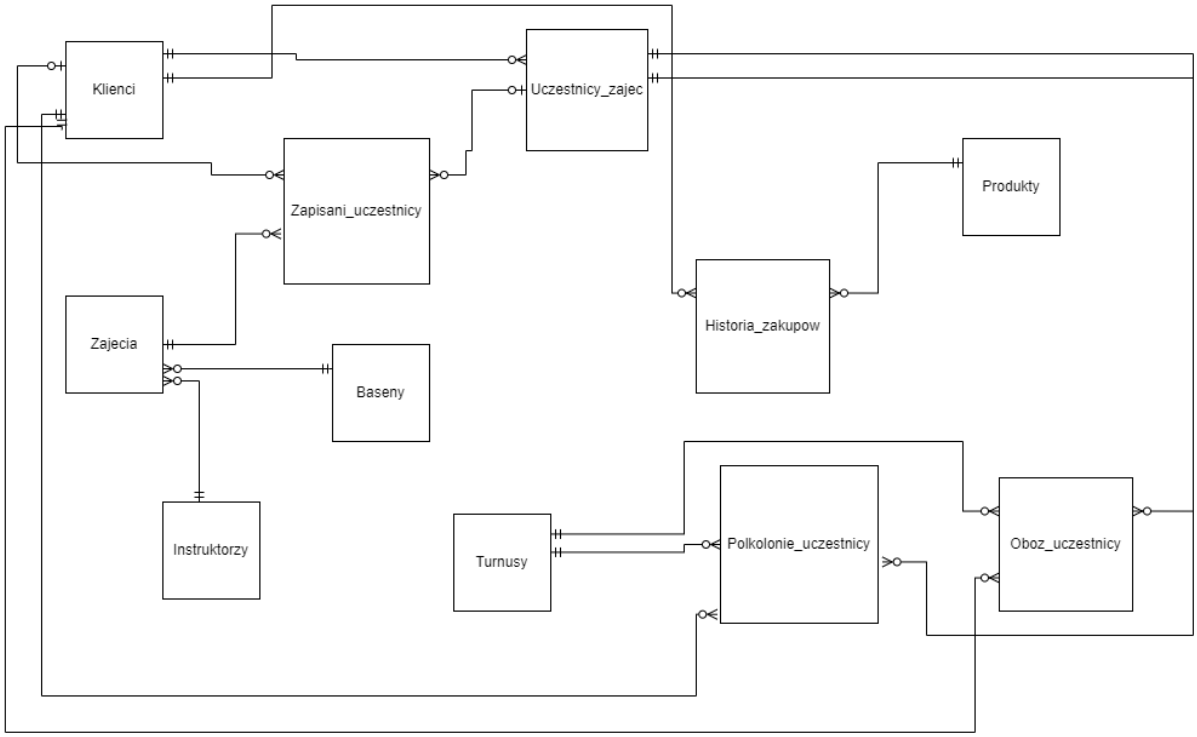
Produkty (id_przedmiotu, nazwa_przedmiotu, cena_przedmiotu, ilość_przedmiotow)

Tabela przechowuje informacje na temat produktów w sklepie prowadzonym przez szkółkę pływacką. Oprócz podstawowych cech takich jak nazwa czy cena, tabela zawiera też aktualną ilość produktów w magazynie.

Historia_zakupow (id_transakcji, id_klienta, data_zakupu, suma, id_przedmiotu)

Tabela przechowuje informacje na temat transakcji prowadzonych przez sklep.

2.1.1. Diagram związków encji



Rys. 2. Diagram związków encji

Opis notacji stosowanej do narysowania diagramu znajduje się pod Rys. 4.

2.2. Analiza liczby instancji dla każdej encji

2.2.1. Encja Klient

Aby określić liczbę encji Klient należy zsumować ilość pełnoletnich uczestników zajęć oraz ilość dzieci chodzących na zajęcia(każde dziecko uczestniczące w lekcjach ma przypisanego jednego rodzica). W analizie pominięto przypadki rodzeństw, gdzie kilka dzieci może być przypisanych do tego samego klienta, gdyż liczność takich zdarzeń jest na tyle mała, że nie wpływa znacząco na projekt bazy danych.

Szkółka zakłada prowadzenie zajęć na 3 basenach. W celu zwiększenia przejrzystości dokumentu analiza zostanie podzielona na 3 części, w których zostanie policzona ilość uczestników na każdym z basenów

Basen1:

Na Basenie1 zajęcia odbywają się od poniedziałku do piątku w godzinach 14:00-20:00. Każde zajęcia trwają 1 godzinę. Jedna grupa zajęciowa liczy do 10 uczestników. Z racji na ilość torów na basenie, jednocześnie na zajęciach przebywać mogą 3 grupy. Oznacza to, że w ciągu jednego dnia ilość uczestników zajęć wynosi maksymalnie $30 \cdot 6 = 180$ osób.

Sumując 5 dni tygodnia, w czasie których odbywają się zajęcia, wychodzi 900 osób. Należy tutaj jednak nanieść poprawkę, gdyż większość uczestników bierze udział w zajęciach 2 razy w tygodniu. Ilość indywidualnych uczestników można oszacować na 500 osób.

Basen2:

Na Basenie2 zajęcia odbywają się 3 razy w tygodniu w godzinach 16:00 – 19:00. Każde zajęcia trwają godzinę. Jedna grupa zajęciowa liczy do 8 uczestników. Z racji na ilość torów na basenie, jednocześnie na zajęciach przebywać mogą 2 grupy. Oznacza to, że w ciągu jednego dnia ilość uczestników zajęć wynosi maksymalnie $16 \cdot 3 = 48$ osób.

Sumując 3 dni tygodnia, w czasie których odbywają się zajęcia, wychodzą 144 osoby. Podobnie jak na poprzednim basenie, należy nanieść poprawkę, gdyż większość uczestników bierze udział w zajęciach 2 razy w tygodniu. Ilość indywidualnych klientów można oszacować na 85 osób.

Basen3:

Na Basenie3 zajęcia odbywają się zajęcia indywidualne przez cały tydzień w godzinach 9:00 – 18:00. Zajęcia odbywają się na1 torze, oznacza to maksymalną ilość 9 uczestników dziennie. Co na przestrzeni całego tygodnia daje ilość 63 klientów. Podobnie jak w obu poprzednich przypadkach, większość uczestników uczestniczy w zajęciach 2 albo 3 razy w tygodniu. Ilość indywidualnych klientów na przestrzeni całego tygodnia można oszacować na maksymalnie 30 osób.

Adnotacja do wszystkich basenów:

We wszystkich powyższych przypadkach uwzględniane były zajęcia zarówno dla dzieci, jak i dla dorosłych.

Suma:

Po zsumowaniu wszystkich basenów liczbę instancji encji Klient można oszacować na 615 sztuk.

2.2.2. Encja Uczestnik zajęć

W celu określenie liczby instancji tej encji należy od liczby wystąpień encji Klient odjąć ilość dorosłych uczestników zajęć.

Zajęcia dla dorosłych stanowią 80% zajęć indywidualnych. Można zatem oszacować, że wśród uczestników tych zajęć, znajduje się około 25 osób.

Do tego w skład zajęć na Basenie1 wchodzą również zajęcia grupowe dla dorosłych. Odbywają się one 2 razy w tygodniu. Jak opisano wyżej grupa liczby maksymalnie 10 osób. Uczestnicy w większości uczestniczą w zajęciach 2 razy w tygodniu. Indywidualną ilość dorosłych uczestników tych zajęć można oszacować na 12 osób.

Po zsumowaniu klientów indywidualnych i grupowych wychodzi 37 dorosłych klientów. Odejmując to od sumy obliczonej w punkcie 2.2.1. można założyć, że ilość indywidualnych uczestników zajęć, którzy są dziećmi wynosi 578 osób.

2.2.3. Encja Basen

Jak opisano wyżej, zajęcia odbywają się na 3 różnych basenach. Jest to zatem ilość instancji tej encji.

2.2.4. Encja Instruktor

Na każdą grupę zajęciową przypada 1 instruktor. W celu przejrzystości obliczeń analiza została przeprowadzona osobno dla każdego basenu.

Basen1:

Na tym basenie zajęcia odbywają się przez 6 godzin, po 3 grupy na godzinę. Do obsługi zajęć potrzebnych jest zatem 3 instruktorów. Należy jednak wziąć pod uwagę fakt, że część instruktorów to studenci, którzy nie są w stanie prowadzić dziennie 6 zajęć. Ilość różnych instruktorów w ciągu dnia można oszacować na 10 osób.

Należy również wziąć pod uwagę fakt, że nie każdy instruktor prowadzi zajęcia przez 5 dni w tygodniu. Ilość indywidualnych instruktorów w ciągu tygodnia można oszacować na 12 osób.

Basen2:

Kierując się logiką z poprzedniego akapitu, do obsługi zajęć w ciągi 1 dnia wystarczy 2 instruktorów. Biorąc pod uwagę fakt, że nie każdy instruktor prowadzi wszystkie możliwe zajęcia w ciągu dnia i w ciągu tygodnia można oszacować, że ilość indywidualnych instruktorów w ciągu tygodnia wynosi na tym basenie 5 osób.

Basen3:

Za zajęcia indywidualne na tym basenie odpowiada 7 instruktorów, którzy prowadzą zajęcia przez cały tydzień.

Suma:

Po zsumowaniu instruktorów na 3 basenach ilość instancji encji Instruktor można oszacować na 24 sztuki.

2.2.5. Encja Produkt

W tym przypadku wystarczy zsumować ilość różnych produktów dostępnych w sklepie. Jest to 25 różnych produktów.

2.2.6. Encja Turnus

Aby oszacować ilość encji Turnus należy zsumować ilość turnusów wszystkich obozów letnich oraz półkolonii.

Z racji na to, iż turnus na obozie trwa 2 tygodnie, w ciągu jednego sezonu letniego można przeprowadzić 4 turnusy obozów.

Półkolonie trwają tydzień, zatem w ciągu wakacji można przeprowadzić 8 turnusów.

Suma różnych turnusów w ciągu jednego sezonu wynosi zatem 12 sztuk.

2.2.7. Encja Uczestnik półkolonii

Maksymalna ilość uczestników jednego turnusu półkolonii wynosi 30 osób. W ciągu 8 turnusów maksymalna ilość uczestników wynosi zatem 240 osób. Należy jednak wziąć pod uwagę fakt, że uczestnicy chętnie biorą udział w półkoloniach więcej niż raz, zatem ilość indywidualnych uczestników w ciągu sezonu można oszacować na 200 osób.

2.2.8. Encja Uczestnik Obozu

Maksymalna ilość uczestników jednego turnusu wynosi 50 osób. W ciągu 4 turnusów maksymalna ilość uczestników wynosi zatem 200 osób. W tym przypadku znacznie rzadziej występuje zjawisko uczestnictwa klienta w więcej niż 1 turnusie, choć zdarzają się takie przypadki. Ilość indywidualnych uczestników w ciągu sezonu można oszacować na 190 osób.

2.2.9. Encja Historia zakupów

Po każdym zakupie, dowód w postaci encji w bazie danych przechowywany jest przez rok od daty transakcji. W ciągu tygodnia średnio przeprowadza się 15 transakcji. Można zatem oszacować, że ilość instancji tej encji będzie wynosiła maksymalnie 180 sztuk.

2.2.10. Encja Kupione produkty

Ilość wystąpień tej encji jest bezpośrednio powiązana z ilością wystąpień encji Historia zakupów, gdyż tabela Kupione_produkty jest tabelą pośrednią dla tabel Produkty i Historia_zakupow. Ilość będzie więc równa ilości oszacowanej dla encji Historia zakupów, czyli maksymalnie 180 sztuk.

2.3. Analiza użycia identyfikująca podstawowe rodzaje transakcji

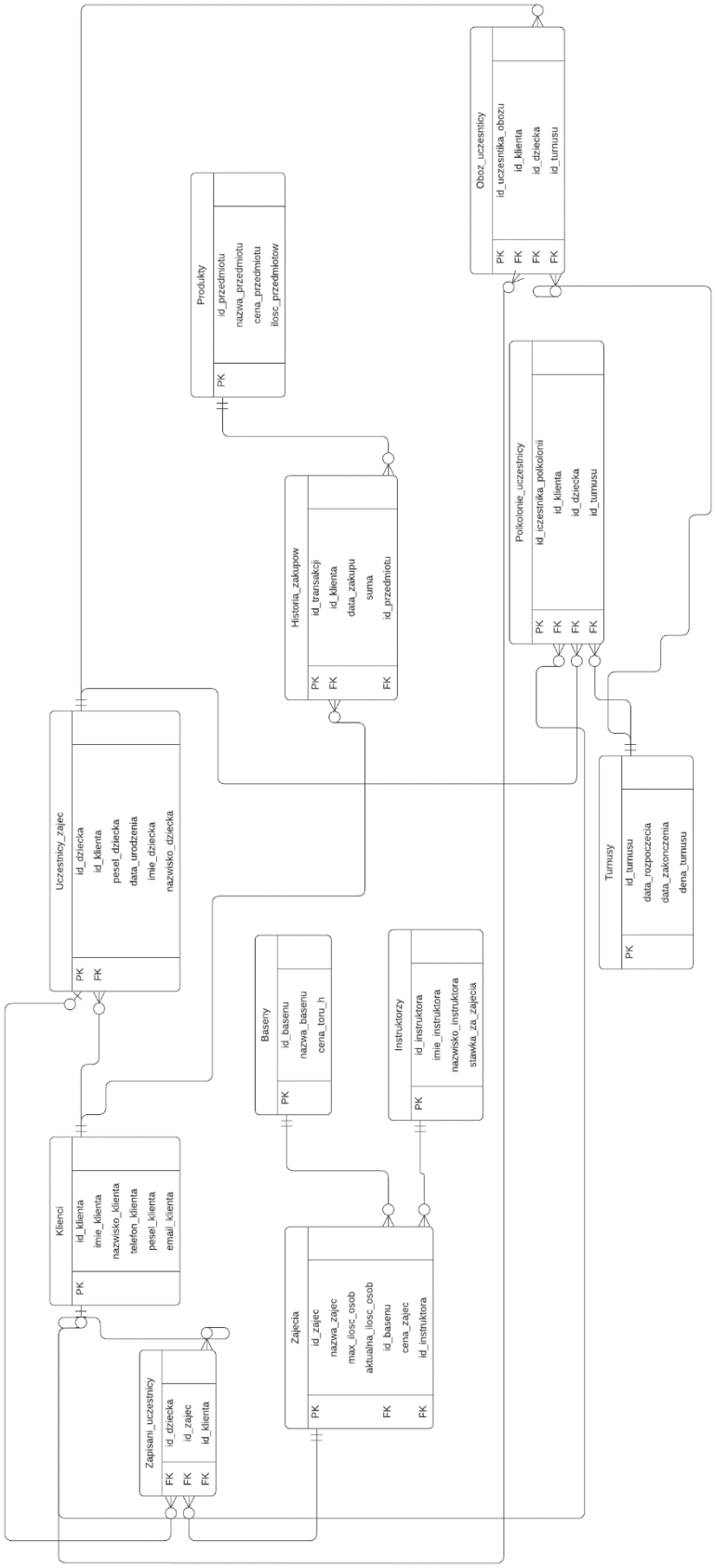
Tabela	Rodzaj transakcji
Klienci	dodawanie, usuwanie, modyfikacja, **wyszukiwanie**
Uczestnicy_zajec_dzieci	dodawanie, usuwanie, modyfikacja, **wyszukiwanie**
Zajecia	**dodawanie**, usuwanie, modyfikacja, **wyszukiwanie**
Baseny	dodawanie, usuwanie, modyfikacja, **wyszukiwanie**
Instruktorzy	dodawanie, usuwanie, modyfikacja, **wyszukiwanie**
Turnusy	dodawanie, usuwanie, modyfikacja, **wyszukiwanie**
Polkolonie_uczesticy	**dodawanie**, usuwanie, modyfikacja, **wyszukiwanie**
Oboz_uczestnicy	**dodawanie**, usuwanie, modyfikacja, **wyszukiwanie**
Produkty	dodawanie, usuwanie, modyfikacja, **wyszukiwanie**
Historia_zakupow	**dodawanie**, wyszukiwanie
Kupione_produkty	**dodawanie**, wyszukiwanie

Tabela 1. Analiza użycia identyfikująca podstawowe rodzaje transakcji

2.4. Model logiczny

Model logiczny bazy danych mówi jak należy dany system zaimplementować w bazie danych.

Z racji na duży rozmiar modelu, znajduje się on na następnej stronie.

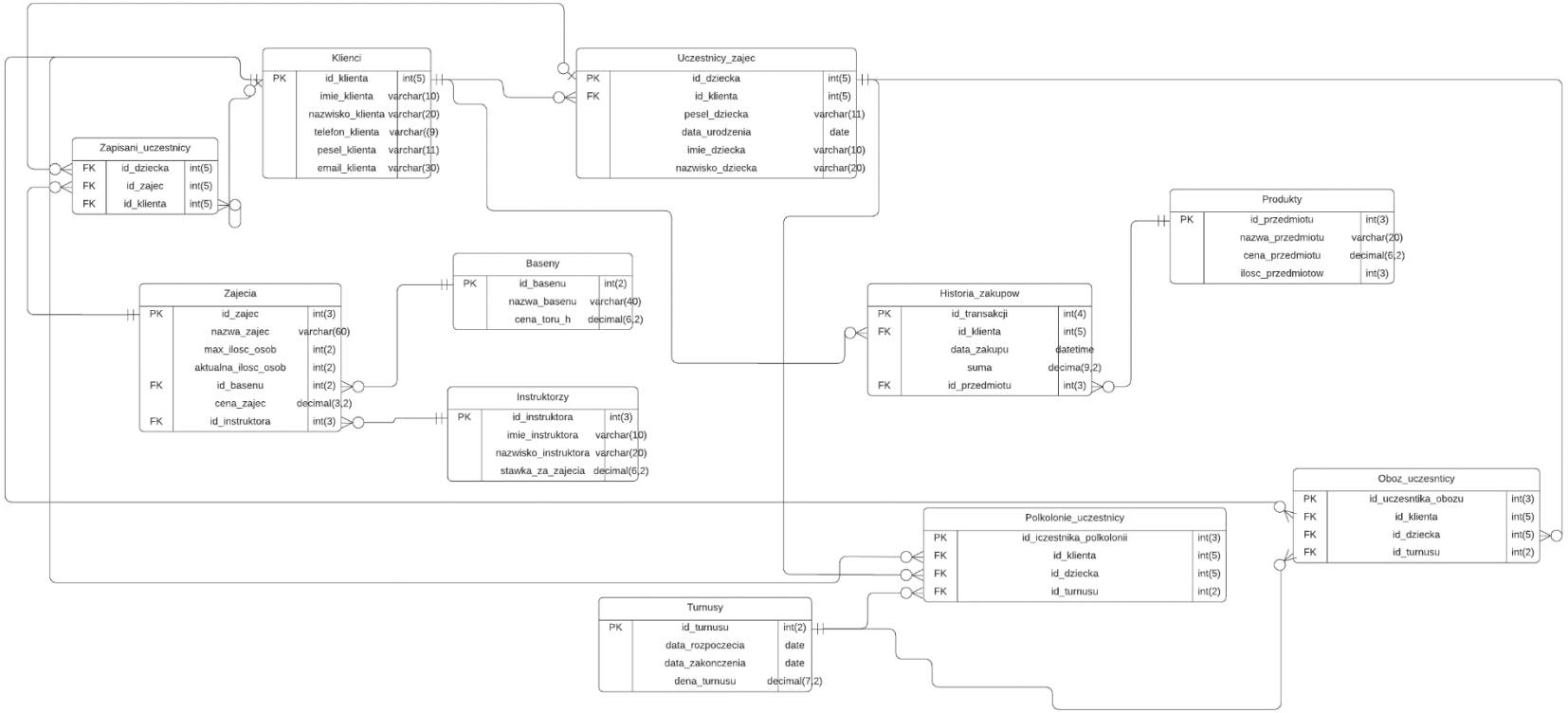


Rys. 3. Model logiczny bazy danych

2.5. Model fizyczny

Model fizyczny bazy danych mówi jak dany system powinien zostać zaimplementowany w konkretnej bazie danych.

Z racji na duży rozmiar modelu, znajduje się on na następnej stronie



Rys. 4. Model fizyczny bazy danych

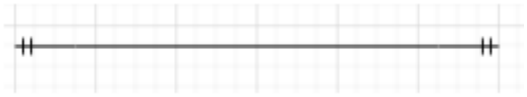
Objaśnienie znaczników pól:

PK – klucz podstawowy

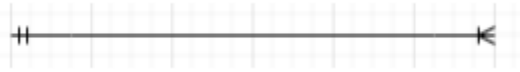
FK – klucz obcy

Objaśnienie strzałek przedstawiających związki:

Cechy związków encji opisane zostały przy użyciu notacji Martina



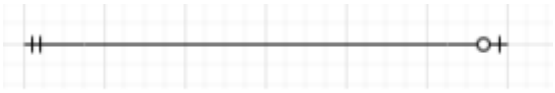
Relacja typu jeden-do-jednego. Każde wystąpienie encji po lewej stronie jest powiązane z jednym wystąpieniem encji po prawej stronie.



Relacja typu jeden-do-wielu. Każde wystąpienie encji po lewej stronie powiązane jest z wieloma wystąpieniami encji po prawej stronie.

Relacja typu wiele-do-wielu. Każde wystąpienie encji po lewej stronie powiązane jest z wieloma wystąpieniami encji po prawej stronie. I odwrotnie.

Na strzałkach przedstawiających relacje, zaznaczamy też to, czy dana strona jest wymagana czy dobrowolna.



Związek powyżej oznacza, że encja zachodząca od lewej strony nie musi być powiązana z encją występującą po prawej stronie. Natomiast encja występująca po prawej stronie musi mieć powiązanie z encją po lewej stronie.

3. Implementacja bazy danych

W celu implementacji bazy danych wybrany został serwer systemu zarządzania bazą danych MySQL Server oraz narzędzie MySQL Workbench w celu wspomagania administracją bazy danych.

3.1. Tworzenie bazy danych i tabel

Z racji na to, iż wszystkie tabele zostały zaimplementowane w analogiczny sposób, wklejony został kod 2 przykładowych tabel.

```
CREATE TABLE `Uczestnicy_zajec` (
  `id_dziecka` integer NOT NULL UNIQUE auto_increment,
  `id_klienta` integer NOT NULL,
  `pesel_dziecka` varchar(11) UNIQUE NOT NULL,
  `data_urodzenia` date NOT NULL,
  `imie_dziecka` varchar(10) NOT NULL,
  `nazwisko_dziecka` varchar(20) NOT NULL,
  PRIMARY KEY (`id_dziecka`),
  FOREIGN KEY (`id_klienta`) REFERENCES `Klienci`(`id_klienta`)
);
```

```
CREATE TABLE `Zajecia` (
  `id_zajec` integer NOT NULL UNIQUE auto_increment,
  `nazwa_zajec` varchar(60) NOT NULL,
  `max_ilosc_osob` integer NOT NULL,
  `aktualna_ilosc_osob` integer NOT NULL,
  `id_basenu` integer NOT NULL,
  `cena_zajec` decimal(5,2) NOT NULL,
  `id_instruktora` integer NOT NULL,
  PRIMARY KEY (`id_zajec`),
  FOREIGN KEY (`id_basenu`) REFERENCES `Baseny`(`id_basenu`),
  FOREIGN KEY (`id_instruktora`) REFERENCES
`Instruktorzy`(`id_instruktora`)
);
```

3.2. Wypełnienie bazy danych przykładowymi danymi

Podobnie jak w poprzednim podpunkcie, załączone zostały jedynie przykładowe 2 skrypty.

```
INSERT INTO instruktorzy
(imie_instruktora,nazwisko_instruktora,stawka_za_zajecia)
VALUES ( "Anna", "Kraul", 40.00);

INSERT INTO instruktorzy
(imie_instruktora,nazwisko_instruktora,stawka_za_zajecia)
VALUES ( "Ilona", "Żabka", 45.00);

INSERT INTO instruktorzy
(imie_instruktora,nazwisko_instruktora,stawka_za_zajecia)
VALUES ( "Marcin", "Motylek", 35.00);
```

```
INSERT INTO instruktorzy
(imie_instruktora,nazwisko_instruktora,stawka_za_zajecia)
VALUES ( "Mikołaj", "Deska", 40.00);

INSERT INTO instruktorzy
(imie_instruktora,nazwisko_instruktora,stawka_za_zajecia)
VALUES ( "Tadeusz", "Okoń", 40.00);
```

```
INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Nauka pływania dla dzieci PON 13:00", 10, 2,1, 40.00, 1);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Nauka pływania dla dzieci WT 14:00", 10, 3,1, 40.00, 2);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Doskonalenie pływania dla dzieci PON 15:00", 10, 3,1, 50.00, 1);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Nauka pływania dla dzieci PON 16:00", 10, 1,2, 50.00, 2);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Skoki do wody dzieci+dorośli ND 9:00", 10, 0,2, 60.00, 4);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Nauka pływania dla dorosłych PON 14:00", 10, 2,1, 50.00, 1);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Nauka pływania dla dorosłych WT 14:00", 10, 2,1, 50.00, 3);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Doskonalenie pływania dla dorosłych SR 18:00", 10, 3,1, 70.00, 3);

INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("Doskonalenie pływania dla dorosłych CZW 17:00", 10, 3,2, 70.00,
3);
```

```
INSERT INTO zajecia
(nazwa_zajec,max_ilosc_osob,aktualna_ilosc_osob,id_basenu,cena_zajec,id_ins
truktora)
VALUES ("BABY SWIM PT 12:00", 10, 0,3, 30.00, 1);
```

3.3. Procedury

W celu obsługi bazy danych zostały stworzone procedury umożliwiające wykonanie akcji zgodnych z diagramem przypadków użycia.

Spis zaimplementowanych procedur:

-dochod_ze_sklepu – wypisuje na ekran dochód ze sklepu prowadzonego przez szkółkę, argumentami procedury są dwie daty, które wyznaczają okres, z którego dochód chcemy obliczyć

-dodanie_basenu – procedura pozwalająca dodać nowy basen do tabeli baseny, jej argumentami są nazwa basenu, oraz cena wynajęcia toru na godzinę. Procedura nie pozwala dodać basenu o nazwie takiej samej jak nazwa basenu w innym rekordzie tabeli.

-dodanie_dziecka – procedura dodaje dziecko do tabeli oraz przypisuje mu rodzica(klienta). Niemożliwe jest dodanie dziecka do tabeli bez uprzedniego dodania rekordu rodzica(klienta) w tabeli klienci. Jako argumenty procedura przyjmuje dane osobowe dziecka oraz pesel rodzica.

-dodanie_instruktora – procedura tworzy nowy rekord instruktora w tabeli instruktorzy. Argumentami procedury są dane osobowe instruktora oraz jego stawka za zajęcia.

-dodanie_klienta – procedura dodaje klienta do tabeli klienci. Argumentami procedury są dane osobowe i kontaktowe klienta.

-dodawanie_zajec - procedura pozwala na dodanie nowych zajęć do tabeli Zajecia. Argumentami procedury są informacje dotyczące nazwy oraz terminu zajęć, koszt zajęć, maksymalna ilość osób, które mogą zapisać się na zajęcia oraz nazwisko instruktora, który zostanie przypisany do zajęć.

-ilosc_zapisanych_na_oboz – procedura oblicza ilość osób zapisanych na obóz, o numerze turnusu przekazanym jako parametr procedury.

-ilosc_zapisanych_na_polkolonie – procedura działa tak samo jak ta opisania powyżej, tylko dla półkolonii

-kupno_produktu – procedura pozwala na kupno produktu przez klienta. Argumentami procedury jest pesel klienta, id_przedmiotu oraz ilość sztuk, które chce zakupić klient. Procedura nie wykona się jeśli w sklepie będzie niewystarczająca ilość produktów. Po pomyślnym dokonaniu zakupu ilość przedmiotów w sklepie ulegnie zmianie zgodnie z ilością zakupionych przez klienta przedmiotów.

-pensje_instruktorow – procedura zwraca sumę pensji instruktorów z okresu czasowego podanego jako argument procedury.

-pokaz_historie_zakupow – procedura zwraca całą historię zakupów klienta, który jest identyfikowany po numerze pesel podanym jako argument procedury.

-przychod_z_obozow_i_polkolonii – procedura zwraca przychód z obozów i półkolonii z jednego sezonu wakacyjnego

-przychod_z_wszystkich_zajec – procedura zwraca sumę przychodów ze wszystkich zajęć prowadzonych przez szkółkę. Argumentem procedury jest okres czasu, z którego obliczany będzie przychód.

-uczestnicy_zajec – procedura zwraca imiona i nazwiska uczestników zajęć o numerze id podanym jako argument procedury.

-usuwanie_basenu – procedura usuwa basen z tabeli baseny. Basen nie może być usunięty jeśli jest przypisany do jakichś zajęć.

-usuwanie_dziecka – procedura usuwa dziecko z tabeli Uczestnicy_zajec. Dziecko zostanie automatycznie wypisane z zajęć, jeśli jest do jakichś przypisane.

-usuwanie_instruktora – procedura usuwa instruktora z tabeli instruktorzy, identyfikowanego po numerze id podanym jako argument procedury. Instruktor nie może zostać usunięty jeżeli jest przypisany do jakichś zajęć.

-usuwanie_klienta – procedura usuwa klienta z tabeli klienci. Usuwanie nie powiedzie się jeżeli klient ma przypisane jakieś dziecko. Klient przy usuwaniu zostanie automatycznie wypisany z grupy zajęciowej jeżeli jest do takowej przypisany.

-usuwanie_zajec – procedura usuwa zajęcia z tabeli Zajecia. Usuwanie nie powiedzie się jeżeli do zajęć przypisani są jacyś klienci/dzieci.

-wypisanie_dziecka_z_obozu – procedura wypisuje dziecko z obozu, jeżeli jest ono na takowy zapisane. Argumentami procedury są pesel oraz numer turnusu.

-wypisanie_dziecka_z_polkolonii –procedura działa analogicznie do tej opisanej wyżej, tylko działa dla półkolonii

-wypisanie_dziecka_z_zajec – procedura wypisuje dziecko z zajęć, jeżeli jest ono na takowe zapisane. Argumentami procedury są pesel dziecka oraz numer zajęć.

-wypisanie_klienta_z_zajec – procedura wypisuje klienta z zajęć, jeżeli jest on na takowe zapisany. Argumentami procedury są pesel klienta oraz numer zajęć.

-wyswietl_oferte_obozowa – procedura wyświetla informacje o obozach organizowanych przez szkółkę.

-wyswietl_oferte_polkolonii – procedura wyświetla informacje o półkoloniach organizowanych przez szkółkę.

-wyswietl_oferte_zajeciowa – procedura wyświetla informacje o zajęciach organizowanych przez szkółkę.

-zapis_dziecka_na_oboz – procedura dokonuje zapisu dziecka na obóz. Dziecko nie może być zapisane więcej niż 1 raz na ten sam turnus. Argumentami procedury jest pesel dziecka oraz numer turnusu

-zapis_dziecka_na_polkolonie – procedura działa analogicznie jak ta opisana powyżej ale dla półkolonii

-zapis_dziecka_na_zajecia – procedura dokonuje zapisu dziecka na wybrane zajęcia. Dziecko nie może być zapisane więcej niż 1 raz na te same zajęcia. Argumentami procedury jest pesel dziecka oraz numer zajęć.

-zapis_klienta_na_zajecia – procedura działa analogicznie jak procedura zapisu dziecka na zajęcia

-zapisani_na_oboz – procedura zwraca dane dzieci zapisanych na obóz identyfikowany przez numer turnusu, który jest podawany jako argument procedury.

-zapisani_na_polkolonie – procedura działa analogicznie jak ta opisana powyżej, tylko dla półkolonii.

-zamiana_danych_kontaktowych_klienta – procedura dokonuje zmiany danych kontaktowych i osobowych podanych przez klienta. Zmianom podlega imię, nazwisko, numer telefonu oraz email

-zmiana_instruktora_zajec – procedura zmiany instruktora przypisanego do konkretnych zajęć. Argumentami procedury są numer zajęć oraz id instruktora

Przykładowe skrypty wybranych procedur:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `pensje_instruktorow`(IN
ilosc_tygodni integer)
    DETERMINISTIC
BEGIN
        DECLARE i integer;
    DECLARE j integer;
    DECLARE nazwisko varchar(20);
    DECLARE suma decimal (9,2);
    SET suma = 0;
    SET i = (SELECT min(id_instruktora) FROM instruktorzy);
    SET j = (SELECT max(id_instruktora) FROM instruktorzy);

    sloop: LOOP
    SET nazwisko = (SELECT nazwisko_instruktora FROM instruktorzy WHERE
id_instruktora = i);
                IF((SELECT COUNT(*) FROM zajecia WHERE id_instruktora = i)
!= 0) THEN
```

```
                        SET suma = suma + (SELECT
pensja_instruktora(ilosc_tygodni, nazwisko) FROM instruktorzy WHERE
id_instruktora = i);
        END IF;

        IF( i != j) THEN
                SET i = i+1;
        ITERATE sloop;
        ELSE
                LEAVE sloop;
        END IF;

    END LOOP;

    SELECT suma;

END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zapis_dziecka_na_zajecia`(IN
pesel varchar(11) , numer_zajec integer)
BEGIN
        if((SELECT max_ilosc_osob FROM zajecia WHERE id_zajec =
numer_zajec) > (SELECT aktualna_ilosc_osob FROM zajecia WHERE id_zajec =
numer_zajec )
    AND numer_zajec NOT IN (SELECT id_zajec FROM zapisani_uczestnicy WHERE
id_zajec = numer_zajec AND id_dziecka = (SELECT id_dziecka FROM
uczestnicy_zajec WHERE pesel_dziecka = pesel)))  THEN

   INSERT INTO zapisani_uczestnicy (id_dziecka, id_zajec, id_klienta)
   VALUES ((SELECT id_dziecka FROM uczestnicy_zajec WHERE pesel_dziecka =
pesel), numer_zajec, NULL);

        UPDATE zajecia
    SET aktualna_ilosc_osob = aktualna_ilosc_osob +1
    WHERE id_zajec = numer_zajec;

        end if;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `kupno_produktu`(IN
idprzedmiotu integer, pesel varchar(11), ilosc integer)
BEGIN
        IF(((SELECT ilosc_przedmiotow FROM produkty WHERE id_przedmiotu =
idprzedmiotu) - ilosc) > 0 AND length(pesel) = 11 ) THEN
                INSERT INTO historia_zakupow(id_klienta,suma,
id_przedmiotu,data_zakupu)
        VALUES((SELECT id_klienta FROM klienci WHERE pesel_klienta =
pesel), ilosc * (SELECT cena_przedmiotu from produkty WHERE id_przedmiotu =
idprzedmiotu), idprzedmiotu, '0000-00-00');

        UPDATE produkty
        SET ilosc_przedmiotow = ilosc_przedmiotow - ilosc
        WHERE id_przedmiotu = idprzedmiotu;
```

```
    END IF;
END
```

3.4. Funkcje

Głównym zadaniem funkcji było skrócenie skryptów procedur poprzez wywoływanie ich wewnątrz procedur. Funkcje w odróżnieniu od procedur zawsze zwracały jakąś wartość. Z tego powodu były one wykorzystywane głównie do generowania danych dotyczących przychodów i kosztów.

Zaimplementowane funkcje:

-dochod_ze_sklepu – funkcja zwracała dochód uzyskany ze sprzedaży produktów w sklepie prowadzonym przez szkółkę. Argumentami funkcji są daty definiujące okres, z którego przychód chcemy obliczyć

-ilosc_zapisanych_na_turnus_oboz – funkcja zwracała ilość osób zapisanych na poszczególny turnus obozu.

-ilosc_zapisanych_na_turnus_polkolonie – funkcja działa analogicznie do tej opisanej powyżej ale dla półkolonii.

-koszty_wynajecia_torow – funkcja zwraca koszty wynajmowania torów przez szkółkę w okresie zdefiniowanym w argumencie funkcji.

-pensja_instruktora – funkcja zwraca pensję instruktora identyfikowanego po nazwisku. Drugim argumentem funkcji jest okres czasu, na podstawie którego pensja ma zostać obliczona

-przychod_z_konkretnych_zajec – funkcja zwraca przychód z konkretnych zajęć identyfikowanych po numerze zajęć. Drugim argumentem funkcji jest okres czasu, na podstawie którego funkcja ma wygenerować przychód.

-przychod_z_obozow – funkcja generuje przychód z wszystkich obozów odbywających się w sezonie

-przychod_z_polkolonii – funkcja generuje przychód z wszystkich półkolonii odbywających się w sezonie.

Kody wybranych funkcji:

```
CREATE DEFINER=`root`@`localhost` FUNCTION
`dochod_ze_sklepu`(data_poczatkowa datetime, data_koncowa datetime) RETURNS
decimal(9,2)
    DETERMINISTIC
BEGIN
        DECLARE sumakoncowa DECIMAL (9,2);
    DECLARE i integer;
    DECLARE j integer;
```

```
    SET i = (SELECT min(id_transakcji) FROM historia_zakupow WHERE
data_zakupu BETWEEN data_poczatkowa AND data_koncowa);
        SET j = (SELECT max(id_transakcji) FROM historia_zakupow WHERE
data_zakupu BETWEEN data_poczatkowa AND data_koncowa);
    SET sumakoncowa = 0;
    sloop: LOOP

        IF((SELECT data_zakupu FROM historia_zakupow WHERE id_transakcji =
i) BETWEEN data_poczatkowa AND data_koncowa)  THEN
        SET sumakoncowa = sumakoncowa + (SELECT suma from historia_zakupow
WHERE id_transakcji = i);
        END IF;

    IF( i != j) THEN
                SET i = i+1;
        ITERATE sloop;
        ELSE
                LEAVE sloop;
        END IF;

    END LOOP;

RETURN sumakoncowa;
END
```

```
CREATE DEFINER=`root`@`localhost` FUNCTION
`koszty_wynajecia_torow`(ilosc_tygodni integer) RETURNS decimal(9,2)
    DETERMINISTIC
BEGIN
        DECLARE suma decimal(9,2);
    DECLARE i integer;
    DECLARE j integer;
    SET i = (SELECT min(id_zajec) FROM zajecia);
    SET j = (SELECT max(id_zajec) FROM zajecia);
    SET suma = 0;
    sloop: LOOP
    IF((SELECT COUNT(*) FROM zajecia WHERE id_zajec = i) != 0) THEN
    SET suma = suma + (SELECT COUNT(*) FROM zajecia a INNER JOIN baseny b
ON a.id_basenu = b.id_basenu WHERE a.id_zajec = i) * ( SELECT cena_toru_h
FROM zajecia a INNER JOIN baseny b on a.id_basenu = b.id_basenu WHERE
a.id_zajec = i);
    END IF;

 IF( i != j) THEN
                SET i = i+1;
        ITERATE sloop;
        ELSE
                LEAVE sloop;
        END IF;

    END LOOP;
SET suma = suma* ilosc_tygodni;
RETURN suma;
END
```

3.5. Widoki

Widoki zostały wykorzystane głównie w celu przedstawiania użytkownikom jedynie niezbędnych danych w konkretnej tabeli oraz w celu uproszczenia skryptów w procedurach.

Zaimplementowane widoki:

-grupy_zajeciowe_view – widok pokazuje wszystkie aktualne grupy zajęciowe wraz z niezbędnymi danymi dotyczącymi zajęć: nazwa, maksymalna ilość osób, aktualna ilość osób, cena zajęć oraz nazwa basenu, na którym odbywają się zajęcia.

-obozy_view – widok przedstawia ofertę obozową szkółki

-polkolonie_view – widok przedstawia ofertę półkolonii organizowanych przez szkółkę.

-sklep_view – widok przedstawia wszystkie dostępne w sklepie produkty, czyli te, których ilość nie jest równa 0.

Przykładowy widok:

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `grupy_zajeciowe` AS
    SELECT
        `a`.`id_zajec` AS `numer_zajec`,
        `a`.`nazwa_zajec` AS `nazwa_zajec`,
        `a`.`max_ilosc_osob` AS `max_ilosc_osob`,
        `a`.`aktualna_ilosc_osob` AS `aktualna_ilosc_osob`,
        `a`.`cena_zajec` AS `cena_zajec`,
        `b`.`nazwa_basenu` AS `nazwa_basenu`
    FROM
        (`zajecia` `a`
        JOIN `baseny` `b` ON ((`a`.`id_basenu` = `b`.`id_basenu`)))
```

3.6. Triggery

W bazie danych zaimplementowany został jeden trigger. Jego zadaniem jest ustawienie dokładnej daty zakupu produktu w tabeli historia_zakupow po każdym zakupie dokonanym przez klienta.

Kod Triggera:

```
CREATE TRIGGER po_kupnie_produktu
AFTER UPDATE ON produkty
FOR EACH ROW
UPDATE historia_zakupow
SET data_zakupu = NOW()
WHERE data_zakupu = '0000-00-00';
```

4. Przypadki użycia

W niniejszej sekcji pokazane zostaną przykładowe przypadki użycia.

4.1. Zapis dziecka na zajęcia.

Aby zapisać dziecko na zajęcia należy mieć utworzony w tabeli Uczestnicy_zajec rekord reprezentujący dziecko oraz w tabeli Klienci rekord reprezentująca klienta, któremu przypisane będzie dane dziecko. Przypadek dodania klienta i dziecka nie będzie omawiany ze względu na niski poziom zaawansowania wymaganych do tego procedur.

Przy wywoływaniu procedury zapisu dziecka na zajęcia należy jako parametry przekazać pesel zapisywanego dziecka oraz ID zajęć, na które uczestnik ma zostać zapisany.

```
call zapis_dziecka_na_zajecia("12332112332", 2);
```

Powyższy skrypt zapisuje dziecko o peselu „12332112332” na zajęcia, których ID równa się 1. Procedura działa w następujący sposób:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `zapis_dziecka_na_zajecia`(IN pesel varchar(11) , numer_zajec integer)
BEGIN
    if((SELECT max_ilosc_osob FROM zajecia WHERE id_zajec = numer_zajec) >
            (SELECT aktualna_ilosc_osob FROM zajecia WHERE id_zajec = numer_zajec )
    AND numer_zajec NOT IN
    (SELECT id_zajec FROM zapisani_uczestnicy WHERE id_zajec = numer_zajec
        AND id_dziecka = (SELECT id_dziecka FROM uczestnicy_zajec WHERE pesel_dziecka = pesel)))  THEN

   INSERT INTO zapisani_uczestnicy (id_dziecka, id_zajec, id_klienta)
   VALUES ((SELECT id_dziecka FROM uczestnicy_zajec WHERE pesel_dziecka = pesel), numer_zajec, NULL);

    UPDATE zajecia
    SET aktualna_ilosc_osob = aktualna_ilosc_osob +1
    WHERE id_zajec = numer_zajec;

    end if;
END
```

Procedura sprawdza czy w wybranej grupie są jeszcze wolne miejsca, następnie dodaje rekord odpowiedzialny za przypisanie dziecka do zajęć do tabeli zapisani_uczestnicy, która jest tabelą łączącą tabelę Uczestnicy_zajec oraz tabelę Zajecia. W kolejnym kroku aktualna ilość zapisanych uczestników na dane zajęcia zostaje zinkrementowana. Procedura ma zabezpieczenie, które nie pozwala na ponowny zapis tego samego dziecka na te same zajęcia.

Tabele Zapisani_uczesnticy oraz Zajecia przed dokonaniem zapisu:

id_zajec	nazwa_zajec	max_ilosc_osob	aktualna_ilosc_osob	id_basenu	cena_zajec	id_instruktora
1	Nauka pływania dla dzieci PON 13:00	10	2	1	80.00	1
2	Nauka pływania dla dzieci WT 14:00	10	3	1	40.00	2
3	Doskonalenie pływania dla dzieci PON 15:00	10	3	1	50.00	1
4	Nauka pływania dla dzieci PON 16:00	10	1	2	50.00	2
5	Skoki do wody dzieci+dorośli ND 9:00	10	0	2	60.00	4
6	Nauka pływania dla dorosłych PON 14:00	10	2	1	50.00	1
7	Nauka pływania dla dorosłych WT 14:00	10	2	1	50.00	3
8	Doskonalenie pływania dla dorosłych SR 18:00	10	3	1	70.00	3
9	Doskonalenie pływania dla dorosłych CZW 17:00	10	3	2	70.00	3
10	BABY SWIM PT 12:00	10	0	3	30.00	1

Tabela 1. Tabela Zajecia

id_dziecka	id_zajec	id_klienta
3	1	NULL
1	1	NULL
3	2	NULL
4	2	NULL
5	2	NULL
1	3	NULL
2	3	NULL
6	3	NULL
7	4	NULL
NULL	6	1
NULL	6	2
NULL	7	3
NULL	7	2
NULL	8	4
NULL	8	5
NULL	8	6
NULL	9	7
NULL	9	8
NULL	9	11

Tabela 2. Tabela Zapisani_uczestnicy

Tabele Zapisani_uczestnicy oraz Zajecia po dokonaniu zapisu:

id_zajec	nazwa_zajec	max_ilosc_osob	aktualna_ilosc_osob	id_basenu	cena_zajec	id_instruktora
1	Nauka pływania dla dzieci PON 13:00	10	2	1	80.00	1
2	Nauka pływania dla dzieci WT 14:00	10	4	1	40.00	2
3	Doskonalenie pływania dla dzieci PON 15:00	10	3	1	50.00	1
4	Nauka pływania dla dzieci PON 16:00	10	1	2	50.00	2
5	Skoki do wody dzieci+dorośli ND 9:00	10	0	2	60.00	4
6	Nauka pływania dla dorosłych PON 14:00	10	2	1	50.00	1
7	Nauka pływania dla dorosłych WT 14:00	10	2	1	50.00	3
8	Doskonalenie pływania dla dorosłych SR 18:00	10	3	1	70.00	3
9	Doskonalenie pływania dla dorosłych CZW 17:00	10	3	2	70.00	3
10	BABY SWIM PT 12:00	10	0	3	30.00	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela 3. Zajecia po zapisie uczestnika na zajecia 2.

NULL	8	5
NULL	8	6
NULL	9	7
NULL	9	8
NULL	9	11
21	2	NULL

Tabela 4. Dół tabeli zapisani_uczestnicy po zapisie dziecka

Jak widać na powyższych obrazkach, pole aktualna_ilosc_osob dla zajec o numerze ID równym 2 w tabeli Zajecia zostało inkrementowane, natomiast w tabeli zapisani_uczestnicy pojawił się nowy rekord reprezentujący zapis dziecka na zajęcia.

4.2. Modyfikacja grup zajęciowych

W celu modyfikacji/dodania grupy zajęciowej należy użyć jednej z procedur: dodawanie_zajec, zmiana_ceny_zajec. W niniejszym podpunkcie opiszemy procedurę dodawanie_zajec, gdyż jednym z jej elementów jest również ustawienie ceny zajęć. Wywołanie tej procedury wygląda następująco:

```
call dodawanie_zajec("Nauka pływania PT 13:00", 10, "Zatoka Sportu", 40.00, "Pływak");
```

Parametrami przekazywanymi do procedury są: nazwa zajęć, maksymalna ilość osób, nazwa basenu, cena zajęć oraz nazwisko instruktora przypisanego do danych zajęć.

Ciało procedury prezentuje się następująco:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `dodawanie_zajec`
(IN nazwa varchar(60), maxiloscosob integer, nazwabasenu varchar(40), cenazajec decimal(5,2), nazwiskoinstruktora varchar(20))
BEGIN
    INSERT INTO zajecia(nazwa_zajec, max_ilosc_osob,aktualna_ilosc_osob, id_basenu, cena_zajec, id_instruktora)
    VALUES(nazwa, maxiloscosob, 0,
    (SELECT id_basenu FROM baseny WHERE nazwa_basenu = nazwabasenu), cenazajec,
    (SELECT id_instruktora FROM instruktorzy WHERE nazwisko_instruktora = nazwiskoinstruktora));

END
```

Procedura dodaje przekazane parametry do nowego rekordu w tabeli Zajecia. Warto tutaj zaznaczyć, iż procedura znajduje ID instruktora oraz ID basenu na podstawie podanej nazwy basenu oraz nazwiska instruktora. Dzięki zastosowaniu takiego mechanizmu niemożliwe jest dodanie zajęć w przypadku podania nazwy basenu, który nie występuje w tabeli Baseny oraz analogicznie w przypadku podania nazwiska instruktora, którego nie ma w tabeli instruktorzy.

Tabela Zajecia przed wywołaniem procedury:

1	Nauka pływania dla dzieci PON 13:00	10	2	1	80.00	1
2	Nauka pływania dla dzieci WT 14:00	10	6	1	40.00	2
3	Doskonalenie pływania dla dzieci PON 15:00	10	3	1	50.00	1
4	Nauka pływania dla dzieci PON 16:00	10	1	2	50.00	2
5	Skoki do wody dzieci+dorośli ND 9:00	10	0	2	60.00	4
6	Nauka pływania dla dorosłych PON 14:00	10	2	1	50.00	1
7	Nauka pływania dla dorosłych WT 14:00	10	2	1	50.00	3
8	Doskonalenie pływania dla dorosłych SR 18:00	10	3	1	70.00	3
9	Doskonalenie pływania dla dorosłych CZW 17:00	10	3	2	70.00	3
10	BABY SWIM PT 12:00	10	0	3	30.00	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela 5. Tabela zajecia przed dodaniem nowych zajec

Tabela Zajecia po dodaniu nowych zajec:

id_zajec	nazwa_zajec	max_ilosc_osob	aktualna_ilosc_osob	id_basenu	cena_zajec	id_instruktora
1	Nauka pływania dla dzieci PON 13:00	10	2	1	80.00	1
2	Nauka pływania dla dzieci WT 14:00	10	6	1	40.00	2
3	Doskonalenie pływania dla dzieci PON 15:00	10	3	1	50.00	1
4	Nauka pływania dla dzieci PON 16:00	10	1	2	50.00	2
5	Skoki do wody dzieci+dorośli ND 9:00	10	0	2	60.00	4
6	Nauka pływania dla dorosłych PON 14:00	10	2	1	50.00	1
7	Nauka pływania dla dorosłych WT 14:00	10	2	1	50.00	3
8	Doskonalenie pływania dla dorosłych SR 18:00	10	3	1	70.00	3
9	Doskonalenie pływania dla dorosłych CZW 17:00	10	3	2	70.00	3
10	BABY SWIM PT 12:00	10	0	3	30.00	1
14	Nauka pływania PT 13:00	10	0	2	40.00	7
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tabela 6. Tabela Zajecia po dodaniu nowych zajec

Jak widać na powyższych obrazkach, do tabeli został dodany rekord o ID równym 14.

4.3. Przeglądanie grup, do których przypisany jest instruktor.

Jedną z funkcjonalności Instruktora jest możliwość przeglądania danych dzieci, które są zapisane do jego zajęć. Aby to zrobić należy wywołać procedurę uczestnicy_zajec i jako jej parametr podać ID zajęć, których uczestników chcemy wyświetlić.

```
call uczestnicy_zajec(3);
```

Ciało procedury prezentuje się następująco:

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `uczestnicy_zajec`(IN idzajec integer)
BEGIN

    IF((SELECT COUNT(*) FROM zapisani_uczestnicy WHERE id_zajec = idzajec AND id_klienta IS NULL) != 0) THEN

        SELECT imie_dziecka, nazwisko_dziecka FROM uczestnicy_zajec
        WHERE id_dziecka IN (SELECT id_dziecka FROM zapisani_uczestnicy WHERE id_zajec = idzajec);

    ELSE IF((SELECT COUNT(*) FROM zapisani_uczestnicy WHERE id_zajec = idzajec AND id_dziecka IS NULL) != 0) THEN
        SELECT imie_klienta, nazwisko_klienta FROM klienci
        WHERE id_klienta IN (SELECT id_klienta FROM zapisani_uczestnicy WHERE id_zajec = idzajec);

    END IF;
    END IF;
END
```

Procedura najpierw sprawdza czy zapisanymi uczestnikami są klienci czy dzieci, a następnie przechodzi do wykonania odpowiedniego fragmentu kodu. Procedura wypisuje imiona i nazwiska uczestników na podstawie ich ID podanych w tabeli zapisani_uczestnicy.

Wynik wywołania procedury dla zajęć o numerze ID równym 3:

imie_dziecka	nazwisko_dziecka
Anna	Saleta
Maria	Okrąg
Adam	Adamski

Tabela 7. Wynik wywołania procedury uczestnicy_zajec

5. Wnioski

Początkowo baza danych nie zakładała posiadania tak dużej ilości tabel oraz relacji wiele-do-wielu, natomiast z racji na spóźnienie z oddaniem pierwszego etapu, co skutkowało obniżeniem oceny końcowej o 0,5, postanowiliśmy rozbudować naszą bazę tak by spełniania wymagania na ocenę 5,0. Do tego w miarę nauki języka SQL na laboratorium oraz wykładzie, nasze umiejętności się zwiększyły, co pozwoliło nam na implementację bardziej zaawansowanej bazy, niż początkowo zakładaliśmy. Uważamy, iż nasz wkład w ten projekt pozwolił na stworzenie bazy danych, która posiada różnorodne funkcjonalności i zastosowania, takie jak przechowywanie danych klientów i ich dzieci, monitorowanie zajęć oraz ich uczestników, organizację obozów oraz półkolonii, prowadzenie sklepu, a także liczenie zysków firmy.