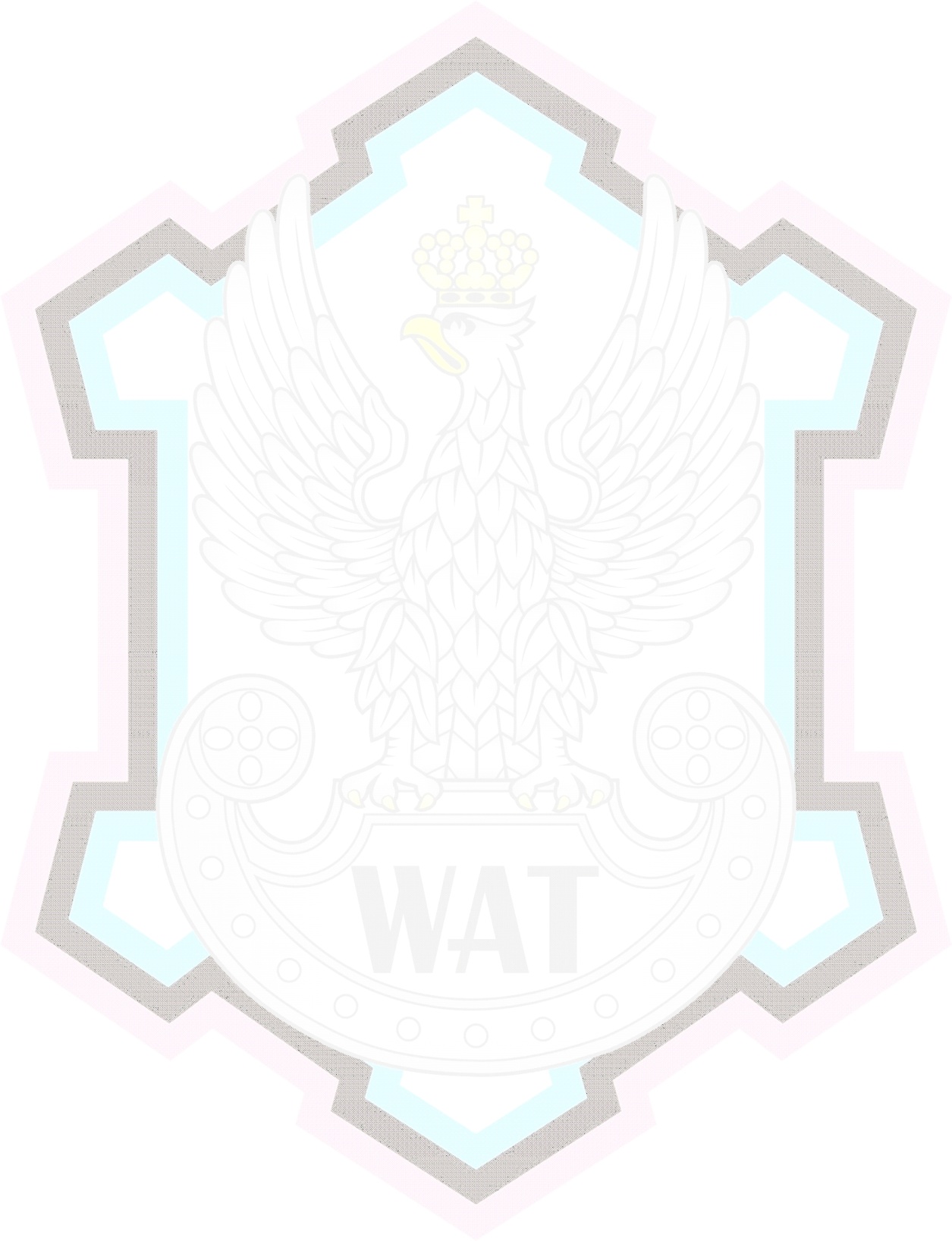
**Wojskowa Akademia Techniczna im. Jarosława Dąbrowskiego**

**Bazy danych**

Prowadzący przedmiot: dr inż. Robert Waszkowski

**Sprawozdanie z projektu przeprowadzonego w ramach ćwiczeń laboratoryjnych**

*"Przychodnia lekarska“*

**Michał Cywiński**

grupa studencka I2X4S1

Wydział Cybernetyki, Wojskowa Akademia Techniczna ul. Kaliskiego 2, 00-908 Warszawa

*Warszawa 2014*

STRESZCZENIE: W sprawozdaniu umieszczono zadanie laboratoryjne wraz z jego wykonaniem oraz treścią. Dołączono dodatkowe komentarze i zastrzeżenia dotyczące założeń projektu.

SŁOWA KLUCZOWE: SQL, baza, danych, tabele, procedura składowana, widok, widok zmaterializowany, trigger, check constraint, indeks

Spis treści

[1. Treść zadania 3](#_Toc378171827)

[2. Opis tematu 4](#_Toc378171828)

[3. Implementacja 5](#_Toc378171829)

[3.1. Skrypt główny 6](#_Toc378171830)

[3.2. Tworzenie tabel 7](#_Toc378171831)

[3.3. Tworzenie indeksów 8](#_Toc378171832)

[3.4. Tworzenie relacji 9](#_Toc378171833)

[3.5. Tworzenie procedur składowanych 10](#_Toc378171834)

[3.6. Tworzenie widoków 15](#_Toc378171835)

[3.7. Tworzenie widoku zmaterializowanego 16](#_Toc378171836)

[3.8. Tworzenie triggerów 17](#_Toc378171837)

[3.9. Tworzenie funkcji 17](#_Toc378171838)

[4. Podsumowanie i wnioski 19](#_Toc378171839)

1. Treść zadania

Zadaniem laboratoryjnym było opracowanie koncepcji bazy danych na ćwiczeniach na wybrany przez siebie temat, a następnie zaimplementowanie tej bazy danych (generacja tabel oraz zależności między nimi) w wybranym przez siebie DBMS na podstawie wiedzy i umiejętności wyniesionych z zajęć laboratoryjnych z założeniem wykorzystania następujących elementów:

* 5-7 tabel
* Klucze głowne, kluccze obce
* Widoki
* Widok zmaterializowany
* Użycie procedur składowanych
* Użycie funkcji
* Użycie triggerów
* Użycie indeksów
* Użycie Check Constraints

Jako temat wybrałem opis przychodni lekarskiej z naciskiem na różnego rodzaju akcje dotyczące obsługi pacjenta.

1. Opis tematu

Na poniższym rysunku zawarłem schemat fizycznego modelu bazy danych opartej o wybrany wcześniej temat:



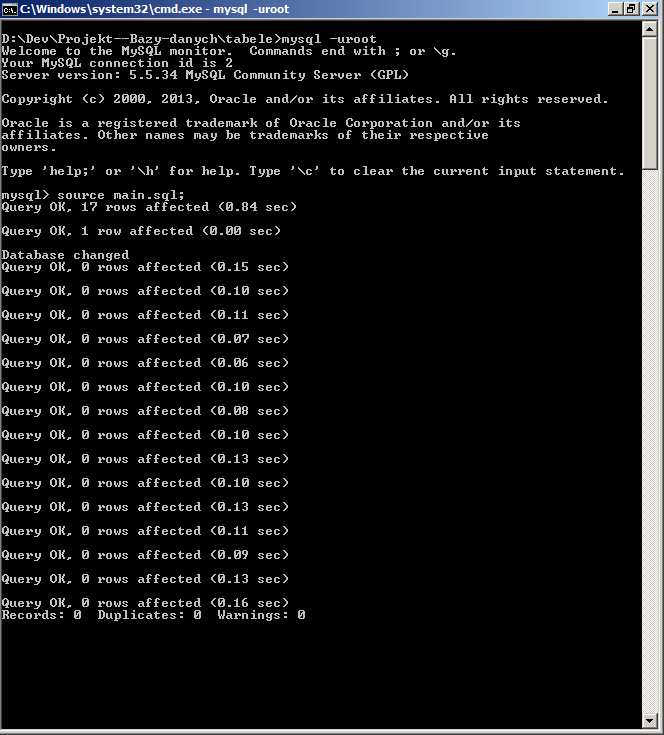
Rysunek : Schemat bazy danych

Baza danych składa się ze spisu lekarzy, pacjentów, gabinetów, leków, chorób, przeprowadzonych badań, recept oraz specjalizacji lekarzy. Tabele z końcówką \_relation służą do obsługi relacji "wiele do wielu" - na przykład obsługi pozycji w recepcie lub przypisania wielu specjalizacji lekarzom.

1. Implementacja

Jako DBMS wybrałem MySQL ze względu na dostępność, szybkość instalacji, łatwość użycia i rozbudowaną społeczność oraz kompatybilność z obecnie zdobywającą coraz większą popularność bazą MariaDB.

Poszczególne elementy bazy danych generuję z pojedynczych plików \*.sql, które są uprzednio wywoływane z pliku main.sql za pomocą komendy MySQL "*source*". Plik main.sql wywołuję jako użytkownik root z poziomu CLI MySQL.



Rysunek : Działanie skryptu main.sql

W pierwszej kolejności skrypt tworzy wszystkie tabele i odpowiednie dla nich klucze główne oraz indeksy i relacje.

Aby ułatwić przyszłe pisanie aplikacji korzystających z bazy danych stworzyłem zestaw procedur składowanych, które wykonuję operację INSERT do poszczególnych tabeli bazy danych.

Następnie skrypt główny wywołuje tworzenie trzech widoków, w tym widoku zmaterializowanego. Dwa pierwsze widoki pokazują lekarzy i ich listę kwalifikacji oraz historię wizyt pacjentów. Trzeci widok pokazuje szczegóły dotyczące danej recepty (należy podać odpowiednie ID recepty).

Przedostatnim z tworzonych elementów są triggery. Inkrementują one odpowiednio liczbę odbytych badań, przebytych chorób i zliczają wizyty w przychodni lekarskiej - wszystkie informacje są aktualizowane w odpowiednich kolumnach tabeli pacjentów.

Ostatnim elementem jest stworzenie funkcji zliczającej średnią chorób przebytych przez wszystkich pacjentów.

* 1. **Skrypt główny**

DROP DATABASE IF EXISTS przychodnia\_lekarska;

CREATE DATABASE przychodnia\_lekarska;

use przychodnia\_lekarska;

source badanie.sql;

source chor\_pac\_relation.sql;

source chor\_wiz\_relation.sql;

source choroba.sql;

source gabinet.sql;

source lek.sql;

source lek\_pac\_relation.sql;

source lek\_spec\_relation.sql;

source lekarz.sql;

source pacjent.sql;

source recepta.sql;

source recepta\_lek\_relation.sql;

source specjalizacja.sql;

source wizyta.sql;

source create\_indexes.sql;

source klucze\_obce.sql

source insert\_choroba.sql;

source insert\_pacjent.sql;

source insert\_lekarz.sql;

source insert\_gabinet.sql;

source insert\_specjalizacja.sql;

source insert\_lek.sql;

source insert\_badanie.sql;

source insert\_wizyta.sql;

source insert\_recepta.sql;

source insert\_pozycja\_recepty.sql;

source insert\_lekarz\_pacjent\_relation.sql;

source insert\_lekarz\_specjalizacja\_relation.sql;

source insert\_zdiagnozowana\_choroba.sql;

source create\_views.sql;

source create\_triggers.sql;

source func\_srednia\_chorob.sql;

* 1. **Tworzenie tabel**

CREATE TABLE lek\_spec\_relation(

id\_lek\_spec\_relation int primary key not null auto\_increment,

id\_lekarz int not null,

id\_specjalizacja int not null

);

CREATE TABLE lekarz(

id\_lekarz int primary key not null auto\_increment,

imie varchar(255),

nazwisko varchar(255),

adres\_zamieszkania text,

data\_urodzenia datetime,

CONSTRAINT chc\_birth\_date CHECK (data\_urodzenia < NOW())

);

CREATE TABLE lek(

id\_lek int primary key not null auto\_increment,

nazwa varchar(255) not null,

opis text,

informacje\_dodatkowe text

);

CREATE TABLE recepta\_lek\_relation(

id\_recepta\_lek\_relation int primary key not null auto\_increment,

id\_recepta int not null,

id\_lek int not null

);

CREATE TABLE specjalizacja(

id\_specjalizacja int primary key not null auto\_increment,

nazwa varchar(255),

id\_gabinet int

);

CREATE TABLE lek\_pac\_relation(

id\_lek\_pac\_relation int primary key not null auto\_increment,

id\_pacjent int not null,

id\_lekarz int not null

);

CREATE TABLE chor\_wiz\_relation(

id\_chor\_wiz\_relation int primary key not null auto\_increment,

id\_choroba int not null,

id\_wizyta int not null

);

CREATE TABLE chor\_pac\_relation(

id\_chor\_pac\_relation int primary key not null auto\_increment,

id\_choroba int not null,

id\_pacjent int not null

);

CREATE TABLE pacjent(

id\_pacjent int not null auto\_increment primary key,

imie varchar(255) not null,

nazwisko varchar(255) not null,

data\_urodzenia datetime,

adres\_zamieszkania text,

liczba\_wizyt int default 0,

liczba\_badan int default 0,

liczba\_chorob int default 0,

CONSTRAINT chk\_counters CHECK (liczba\_wizyt > -1 AND liczba\_badan > -1 AND liczba\_chorob > -1),

CONSTRAINT chc\_birth\_date CHECK (data\_urodzenia < NOW())

);

CREATE TABLE gabinet(

id\_gabinet int primary key not null auto\_increment,

nazwa varchar(255)

);

CREATE TABLE recepta(

id\_recepta int primary key not null auto\_increment,

data\_wystawienia datetime,

id\_pacjent int not null,

id\_lekarz int not null,

id\_wizyta int

);

CREATE TABLE badanie(

id\_badanie int primary key not null auto\_increment,

nazwa\_badania varchar(255),

szczegoly text,

id\_pacjent int,

id\_wizyta int

);

CREATE TABLE choroba(

id\_choroba int primary key not null auto\_increment,

nazwa varchar(255),

opis text

);

CREATE TABLE wizyta(

id\_wizyta int primary key not null auto\_increment,

id\_gabinet int not null,

id\_pacjent int not null,

id\_lekarz int not null,

data\_wizyty datetime,

szczegoly text

);

* 1. **Tworzenie indeksów**

CREATE INDEX pacjent\_imienazwisko

ON pacjent (imie, nazwisko);

CREATE INDEX lekarz\_imienazwisko

ON lekarz (imie, nazwisko);

CREATE INDEX specjalizacja\_nazwa

ON specjalizacja (nazwa);

CREATE INDEX choroba\_nazwa

ON choroba (nazwa);

CREATE INDEX lek\_nazwa

ON lek(nazwa);

* 1. **Tworzenie relacji**

ALTER TABLE recepta\_lek\_relation

add foreign key (id\_recepta) references recepta(id\_recepta) on delete cascade;

ALTER TABLE recepta\_lek\_relation

add foreign key (id\_lek) references lek(id\_lek) on delete cascade;

ALTER TABLE chor\_wiz\_relation

add foreign key (id\_choroba) references choroba(id\_choroba) on delete cascade;

ALTER TABLE chor\_wiz\_relation

add foreign key (id\_wizyta) references wizyta(id\_wizyta) on delete cascade;

ALTER TABLE chor\_pac\_relation

add foreign key (id\_choroba) references choroba(id\_choroba) on delete cascade;

ALTER TABLE chor\_pac\_relation

add foreign key (id\_pacjent) references pacjent(id\_pacjent) on delete cascade;

ALTER TABLE lek\_pac\_relation

add foreign key (id\_pacjent) references pacjent(id\_pacjent) on delete cascade;

ALTER TABLE lek\_pac\_relation

add foreign key (id\_lekarz) references lekarz(id\_lekarz) on delete cascade;

ALTER TABLE lek\_spec\_relation

add foreign key (id\_specjalizacja) references specjalizacja(id\_specjalizacja) on delete cascade;

ALTER TABLE lek\_spec\_relation

add foreign key (id\_lekarz) references lekarz(id\_lekarz) on delete cascade;

ALTER TABLE recepta

add foreign key (id\_pacjent) references pacjent(id\_pacjent) on delete cascade;

ALTER TABLE recepta

add foreign key (id\_lekarz) references lekarz(id\_lekarz) on delete cascade;

ALTER TABLE recepta

add foreign key (id\_wizyta) references wizyta(id\_wizyta) on delete cascade;

ALTER TABLE wizyta

add foreign key (id\_gabinet) references gabinet(id\_gabinet) on delete cascade;

ALTER TABLE wizyta

add foreign key (id\_pacjent) references pacjent(id\_pacjent) on delete cascade;

ALTER TABLE wizyta

add foreign key (id\_lekarz) references lekarz(id\_lekarz) on delete cascade;

ALTER TABLE badanie

add foreign key (id\_pacjent) references pacjent(id\_pacjent) on delete cascade;

ALTER TABLE badanie

add foreign key (id\_wizyta) references wizyta(id\_wizyta) on delete cascade;

ALTER TABLE specjalizacja

add foreign key (id\_gabinet) references gabinet(id\_gabinet) on delete cascade;

* 1. **Tworzenie procedur składowanych**

delimiter //

CREATE PROCEDURE insert\_specjalizacja\_lekarza(

IN in\_id\_lekarz int,

IN in\_id\_specjalizacja int

)

BEGIN

INSERT INTO lek\_spec\_relation(

id\_lekarz,

id\_specjalizacja

)

VALUES(

in\_id\_lekarz,

in\_id\_specjalizacja

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_gabinet(

IN in\_numer\_pokoju varchar(255)

)

BEGIN

INSERT INTO gabinet(

nazwa

)

VALUES(

in\_numer\_pokoju

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_recepta(

IN in\_data\_wystawienia datetime,

IN in\_id\_pacjent int,

IN in\_id\_lekarz int,

IN in\_id\_wizyta int

)

BEGIN

INSERT INTO recepta(

data\_wystawienia,

id\_pacjent,

id\_lekarz,

id\_wizyta

)

VALUES(

in\_data\_wystawienia,

in\_id\_pacjent,

in\_id\_lekarz,

in\_id\_wizyta

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_specjalizacja(

IN in\_nazwa varchar(255),

IN id\_gabinet int

)

BEGIN

INSERT INTO specjalizacja(

nazwa,

id\_gabinet)

VALUES(

in\_nazwa,

in\_gabinet

);

END

//

delimiter ; delimiter //

CREATE PROCEDURE insert\_pozycja\_recepty(

IN in\_id\_recepta int,

IN in\_id\_lek int

)

BEGIN

INSERT INTO recepta\_lek\_relation(

id\_recepta,

id\_lek

)

VALUES(

in\_id\_recepta,

in\_id\_lek

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_choroba(

IN in\_nazwa varchar(255),

IN in\_opis text

)

BEGIN

INSERT INTO choroba(

nazwa,

opis)

VALUES(

in\_nazwa,

in\_opis

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_pacjent(

IN in\_imie varchar(255),

IN in\_nazwisko varchar(255),

IN in\_data\_urodzenia datetime,

IN in\_adres\_zamieszkania text,

IN in\_liczba\_wizyt int,

IN in\_liczba\_badan int,

IN in\_liczba\_chorob int

)

BEGIN

INSERT INTO pacjent(

imie,

nazwisko,

data\_urodzenia,

adres\_zamieszkania,

liczba\_wizyt,

liczba\_badan,

liczba\_chorob)

VALUES(

in\_imie,

in\_nazwisko,

in\_data\_urodzenia,

in\_adres\_zamieszkania,

in\_liczba\_wizyt,

in\_liczba\_badan,

in\_liczba\_chorob

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_badanie(

IN in\_nazwa\_badania varchar(255),

IN in\_szczegoly text,

IN in\_id\_pacjent int,

IN in\_id\_wizyta int

)

BEGIN

INSERT INTO badanie(

nazwa\_badania,

szczegoly,

id\_pacjent,

id\_wizyta

)

VALUES(

in\_nazwa\_badania,

in\_szczegoly,

in\_id\_pacjent,

in\_id\_wizyta

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_zdiagnozowana\_choroba(

IN in\_id\_choroba int,

IN in\_id\_wizyta int

)

BEGIN

INSERT INTO chor\_wiz\_relation(

id\_choroba,

id\_wizyta

)

VALUES(

in\_id\_choroba,

in\_id\_wizyta

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_pacjent\_lekarza(

IN in\_id\_pacjent int,

IN in\_id\_lekarz int

)

BEGIN

INSERT INTO lek\_pac\_relation(

id\_pacjent,

id\_lekarz

)

VALUES(

in\_id\_pacjent,

in\_id\_lekarz

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_wizyta(

IN in\_id\_gabinet int,

IN in\_id\_pacjent int,

IN in\_id\_lekarz int,

IN in\_data\_wizyty datetime,

IN in\_szczegoly text

)

BEGIN

INSERT INTO wizyta(

id\_gabinet,

id\_pacjent,

id\_lekarz,

data\_wizyty,

szczegoly

)

VALUES (

in\_id\_gabinet,

in\_id\_pacjent,

in\_id\_lekarz,

in\_data\_wizyty,

in\_szczegoly

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_lekarz(

IN in\_imie varchar(255),

IN in\_nazwisko varchar(255),

IN in\_data\_urodzenia datetime,

IN in\_adres\_zamieszkania text

)

BEGIN

INSERT INTO lekarz(

imie,

nazwisko,

data\_urodzenia,

adres\_zamieszkania)

VALUES(

in\_imie,

in\_nazwisko,

in\_data\_urodzenia,

in\_adres\_zamieszkania

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_lek(

IN in\_nazwa varchar(255),

IN in\_opis text,

IN in\_informacje\_dodatkowe text

)

BEGIN

INSERT INTO lek(

nazwa,

opis,

informacje\_dodatkowe

)

VALUES(

in\_nazwa,

in\_opis,

in\_informacje\_dodatkowe

);

END

//

delimiter ;

delimiter //

CREATE PROCEDURE insert\_gabinet(

IN in\_numer\_pokoju varchar(255)

)

BEGIN

INSERT INTO gabinet(

nazwa

)

VALUES(

in\_numer\_pokoju

);

END

//

delimiter ;

* 1. **Tworzenie widoków**

CREATE VIEW view\_lista\_kwalifikacji AS

SELECT CONCAT(l.imie, " ", l.nazwisko) "Lekarz", s.nazwa "Specjalność", g.nazwa "Pokój"

FROM specjalizacja s, gabinet g, lekarz l, lek\_spec\_relation rel

WHERE (rel.id\_lekarz = l.id\_lekarz) AND (rel.id\_specjalizacja = s.id\_specjalizacja) AND (s.id\_gabinet = g.id\_gabinet)

ORDER BY l.id\_lekarz ASC;

CREATE VIEW view\_historia\_wizyt AS

SELECT CONCAT(l.imie, " ", l.nazwisko) "Lekarz", w.data\_wizyty "Data wizyty", CONCAT(p.imie, " ", p.nazwisko) "Pacjent"

FROM lekarz l, wizyta w, pacjent p

WHERE (w.id\_lekarz = l.id\_lekarz) AND (w.id\_pacjent = p.id\_pacjent)

ORDER BY p.id\_pacjent, w.data\_wizyty;

* 1. **Tworzenie widoku zmaterializowanego**

delimiter //

CREATE PROCEDURE informacje\_o\_recepcie(

IN in\_id\_recepta int

)

BEGIN

DROP TABLE IF EXISTS widok\_w\_recepcie;

CREATE TABLE widok\_w\_recepcie(

wystawiajacy varchar(255),

pacjent varchar(255),

data\_wystawienia datetime,

liczba\_lekow int

);

INSERT INTO widok\_w\_recepcie(wystawiajacy, pacjent, data\_wystawienia, liczba\_lekow)

VALUES (

(SELECT CONCAT(l.imie, ' ', l.nazwisko)

FROM lekarz l

WHERE id\_lekarz = (SELECT id\_lekarz

FROM recepta

WHERE id\_recepta = in\_id\_recepta)),

(SELECT CONCAT(p.imie, ' ', p.nazwisko)

FROM pacjent p

WHERE id\_pacjent = (SELECT id\_pacjent

FROM recepta

WHERE id\_recepta = in\_id\_recepta)),

(SELECT data\_wystawienia

FROM recepta

WHERE id\_recepta = in\_id\_recepta),

(SELECT COUNT(\*)

FROM recepta\_lek\_relation

WHERE id\_recepta = in\_id\_recepta)

);

END

//

delimiter ;

* 1. **Tworzenie triggerów**

delimiter //

CREATE TRIGGER trigger\_aktualizuj\_badanie

AFTER INSERT ON badanie

FOR EACH ROW

BEGIN

UPDATE pacjent p SET p.liczba\_badan = p.liczba\_badan + 1 WHERE p.id\_pacjent = new.id\_pacjent;

END

//

delimiter ;

delimiter //

CREATE TRIGGER trigger\_aktualizuj\_choroby

AFTER INSERT ON chor\_pac\_relation

FOR EACH ROW

BEGIN

UPDATE pacjent p SET p.liczba\_chorob = p.liczba\_chorob + 1 WHERE p.id\_pacjent = new.id\_pacjent;

END

//

delimiter ;

delimiter //

CREATE TRIGGER trigger\_aktualizuj\_wizyty

AFTER INSERT ON wizyta

FOR EACH ROW

BEGIN

UPDATE pacjent p SET p.liczba\_wizyt = p.liczba\_wizyt + 1 WHERE p.id\_pacjent = new.id\_pacjent;

END

//

delimiter ;

* 1. **Tworzenie funkcji**

delimiter //

CREATE FUNCTION f\_srednia\_zachorowalnosc ()

RETURNS INT

BEGIN

DECLARE avarage INT;

SET avarage = (SELECT AVG(liczba\_chorob) FROM pacjenci);

RETURN avarage;

END;

//

delimiter ;

1. Podsumowanie i wnioski

Zadane elementy bazy danych udało się poprawnie zaimplementować. Jedynym problematycznym elementem okazały się CHECK CONSTRAINTS. MySQL pomyślnie parsuje polecenia zawierające te elementy, natomiast ignoruje ich obecność.

Należy także zaznaczyć, że Packages są elementem typowym dla baz danych Oracle, więc nie było możliwe użyć tej funkcjonalności w MySQL.

Aby polecenia procedur składowanych zostały poprawnie zinterpretowane na etapie ich dodawania, użyłem polecenia "*delimiter*" które definiuje znak kończący polecenie - a więc podmienia średnik odpowiednią kombinacją znaków - w moim wypadku była to kombinacja //.

Budowa bazy danych odbyła się na podstawie wcześniej stworzonego schematu w programie Microsoft Visio. Stworzenie schematu wymaga odpowiedniego przemyślenia logiki związanej z reprezentacją danych oraz ewentualnym podłączeniem aplikacji pod bazę danych. Pewne koncepcje mogą również ulec zmianie podczas pisania kodu języka SQL i wiąże się to z powrotem do Visio oraz odpowiednią modyfikacją.

Proces konwersji schematu na rzeczywistą bazę danych za pomocą SQL miałem okazję dokładnie poznać interfejs CLI MySQL oraz narzędzia phpMyAdmin i MySQL Workbench. Była to okazja do wprawienia się w szybkim rozwiązywaniu problemów na podstawie komunikatów błędów oraz świetna praktyka w poszukiwaniu rozwiązań problemów związanych z samą logiką własnych skryptów.