МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

***КАФЕДРА CИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ***



**Розрахункова робота**

з дисципліни«Організація баз даних та знань»

на тему:“PetApp”

Виконав студент групи КН-209: Мисак Максим

|  |  |
| --- | --- |
| Балів | Дата |
|  |  |

Викладач:  Мельникова Н.І.

Львів-2020

ЗМІСТ

1. Вступ

2. Логічна схема бази даних

3. Опис структури бази даних

4. Фізична модель

5. Ділова модель

6. Запити до бази даних

7. Висновок

8. Список використаних літературних джерел

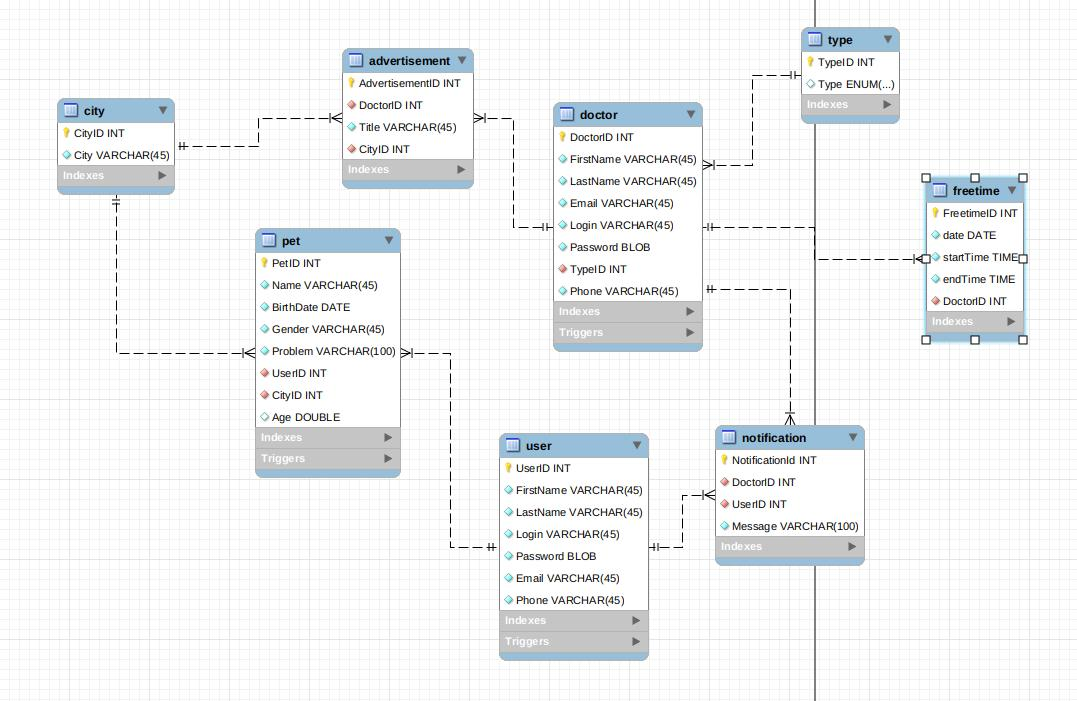
ВСТУП

Сервіс “PetApp” був створений для власників домашніх улюбленців, яким дійсно потрібно швидко та зручно знайти хорошого та кваліфікованого спеціаліста для свого улюбленця, і для ветеринарів та інших спеціалістів, котрі можуть швидко та зручно виставити свої послуги та розпочати власну кар’єру.

Наші переваги:

* Зекономлений час. Вам більше не потрібно обдзвонювати усі клініки міста та домовлятись за часом, адже у нас є зручні фільтрації пошуку за часом.
* Ви можете легко комунікувати з ветеринаром без зайвих телефонних дзвінків або ж навпаки можете скористатись контактними даними ветеринара та поспілкуватись з ним особисто.
* Простота використання. Дизайн нашого продукту є максимально простим і не потрібно більше розбиратись годинами в пошуках тої чи іншої послуги.
* Сервіс “PetApp” є дуже затребуваний як і серед ветеринарів, так і серед власників домашніх улюбленців. Отож він закриває одразу дві області.

ЛОГІЧНА СХЕМА БД



ОПИС СТРУКТУРИ БД

База даних “PetApp” містить 8 таблиць. Всі вони зведені до 3-ої нормальної форми.

1. **Таблиця “doctor”**

Дана таблиця створена для того, щоб зберігати інформацію про ветеринара. Вона містить такі поля: DoctorID (PRIMARY KEY типу int), FirstName (iм’я типу varchar), LastName (прізвище типу varchar), Email (електронна пошта типу varchar), Login (нікнейм на сайті типу varchar), Password (типу varchar), TypeID (FOREIGN KEY посилання на таблицю type, в якій зберігаються дані про тип лікаря (groomer or vet)).

2. **Таблиця “user”**

Дана таблиця була створена для того, щоб зберігати інформацію про користувача. Вона мітсить такі поля: UserID (PRIMARY KEY типу int), FirstName (iм’я типу varchar), LastName (прізвище типу varchar),Password (пароль типу varchar), Email (електронна пошта типу varchar), Phone (номер телефону типу varchar).

3. **Таблиця “pet”**

Дана таблиця була створена для того, щоб зберігати інформацію про оголошення про домашніх улюбленців. Вона містить такі поля:PetID(PRIMARY KEY типу int), Name (кличка типу varchar), BirthDate(дата народження типу date), Gender(стать типу varchar), Problem (опис проблеми, яка турбує домашнього улюбленця), UserID(FOREIGN KEY посилання на таблицю user), CityID(FOREIGN KEY посилання на таблицю city).

4. **Таблиця “freetime”**

Дана таблиця була створена для збереження інформації про розклад ветеринара. Вона містить такі поля:FreetimeID(PRIMARY KEY типу int), date (дата, в який саме день ветеринар має такий вільний час типу date), startTime( початкова година вільного часу типу time), endTime(кінцева година вільного часу типу time), DoctorID(FOREIGN KEY посилання на таблицю ветеринара).

5. **Таблиця “advertisement”**

Дана таблиця була створена для збереження інформації про оголошення, яке виставляє ветеринар.Вона містить такі поля: AdvertisementID(PRIMARY KEY типу int), DoctorID (FOREIGN KEY посилання на таблицю doctor), Title(заголовок оголошення, в якому ветеринар вказує свої послуги типу varchar), CityID(FOREIGN KEY посилання на таблицю city, тобто в даному полі вказується місто роботи ветеринара).

6. **Таблиця “notification”**

Дана таблиця містить інформацію про сповіщеня. Вона містить такі поля: NotificationID(PRIMARY KEY типу int), DoctorID(FOREIGN KEY посилання на таблицю doctor, в якому буде вказуватись ветеринар який відсилає або якому приходить сповіщення), UserID( FOREIGN KEY посилання на таблицю user, в якому буде вказуватись користувач який відсилає або якому приходить сповіщення), Message(текст повідослення типу varchar), Status(enum зі значеннями “Read”-прочитано і “Unread”-непрочитано).

7. **Таблиця “city”**

Дана таблиця була створена для збереження міст. Вона містить такі поля: CityID( PRIMARY KEY типу int), City(назва міста типу varchar). Призначена для зберігання міст.

8. **Таблиця “type”**

Дана таблиця містить інформацію про тип ветеринара. Вона містить такі поля:

TypeID( PRIMARY KEY типу int), Type ( enum зі значеннями Groomer і Vet). Призначена для зберігання типів ветеринарів.

ФІЗИЧНА МОДЕЛЬ БД

CREATE DATABASE IF NOT EXISTS cmd;

USE cmd;

CREATE TABLE IF NOT EXISTS type (

TypeID INT NOT NULL,

Type enum('Groomer','Vet'),

PRIMARY KEY (TypeID));

CREATE TABLE IF NOT EXISTS doctor (

DoctorID INT NOT NULL,

FirstName VARCHAR(45) NOT NULL,

LastName VARCHAR(45) NOT NULL,

Email VARCHAR(45) NOT NULL,

Login VARCHAR(45) NOT NULL,

Password VARCHAR(45) NOT NULL,

TypeID INT NOT NULL,

Phone VARCHAR(45) NOT NULL,

PRIMARY KEY (DoctorID),

CONSTRAINT fk\_doctor\_type1

FOREIGN KEY (TypeID)

REFERENCES type (TypeID)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

CREATE TABLE IF NOT E

XISTS city (

CityID INT NOT NULL,

City VARCHAR(45) NOT NULL,

PRIMARY KEY (CityID));

CREATE TABLE IF NOT EXISTS advertisement (

AdvertisementID INT NOT NULL,

DoctorID INT NOT NULL,

Title VARCHAR(45) NOT NULL,

CityID INT NOT NULL,

PRIMARY KEY (AdvertisementID),

CONSTRAINT appointment\_ibfk\_2

FOREIGN KEY (DoctorID)

REFERENCES doctor (DoctorID),

CONSTRAINT fk\_advertisement\_city1

FOREIGN KEY (CityID)

REFERENCES city (CityID)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

CREATE TABLE IF NOT EXISTS user (

UserID INT NOT NULL,

FirstName VARCHAR(45) NOT NULL,

LastName VARCHAR(45) NOT NULL,

Login VARCHAR(45) NOT NULL,

Password VARCHAR(45) NOT NULL,

Email VARCHAR(45) NOT NULL,

Phone VARCHAR(45) NOT NULL,

PRIMARY KEY (UserID));

CREATE TABLE IF NOT EXISTS freetime (

FreetimeID INT NOT NULL,

date DATE NOT NULL,

startTime TIME NOT NULL,

endTime TIME NOT NULL,

DoctorID INT NOT NULL,

PRIMARY KEY (FreetimeID),

CONSTRAINT fk\_freetime\_doctor1

FOREIGN KEY (DoctorID)

REFERENCES doctor (DoctorID)

ON DELETE CASCADE

ON UPDATE NO ACTION);

CREATE TABLE IF NOT EXISTS notification (

NotificationId INT NOT NULL,

DoctorID INT NOT NULL,

UserID INT NOT NULL,

Message VARCHAR(100) NOT NULL,

Status enum('Read','Unread'),

PRIMARY KEY (NotificationId),

CONSTRAINT notification\_ibfk\_1

FOREIGN KEY (DoctorID)

REFERENCES doctor (DoctorID),

CONSTRAINT notification\_ibfk\_2

FOREIGN KEY (UserID)

REFERENCES user (UserID));

CREATE TABLE IF NOT EXISTS pet (

PetID INT NOT NULL,

Name VARCHAR(45) NOT NULL,

BirthDate DATE NOT NULL,

Gender VARCHAR(45) NOT NULL,

Problem VARCHAR(100) NOT NULL,

UserID INT NOT NULL,

CityID INT NOT NULL,

PRIMARY KEY (PetID),

CONSTRAINT fk\_pet\_user1

FOREIGN KEY (UserID)

REFERENCES user (UserID)

ON DELETE CASCADE

ON UPDATE NO ACTION,

CONSTRAINT fk\_pet\_city1

FOREIGN KEY (CityID)

REFERENCES city (CityID)

ON DELETE NO ACTION

ON UPDATE NO ACTION);

ДІЛОВА МОДЕЛЬ

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблиці  Функції | user | doctor | pet | advertisement | freetime | city | type | notification |
| Cтворення оголошення про домашнього улюбленця | \* |  | \* |  |  | \* |  |  |
| Перевірка на правильність введення емейлу | \* | \* |  |  |  |  |  |  |
| Перевірка на правильність введення паролю | \* | \* |  |  |  |  |  |  |
| Фільтрування оголошень за містом |  | \* |  | \* | \* | \* | \* |  |
| Фільтрування оголошень за часом |  | \* |  | \* | \* | \* | \* |  |
| Валідація на правильність введення часу під час створення “freetime” |  | \* |  |  | \* |  |  |  |
| Автоматичний  статус “unread” під час створення notification | \* | \* |  |  |  |  |  | \* |

– Створення нового оголошення про домашнього улюбленця відбувається за допомогою INSERT INTO в таблицю pet.

– Фільтрування оголошень за містом відбувається за допомогою SELECT

в який входять поля: advertisement.title а також об’єднано з (INNER JOIN) doctor.FirstName, doctor.LastName, doctor.Phone, doctor.Email, об’єднано з freetime.startTime, freetime.endTime, об’єднано з city.City, об”єднано з type.Type. Перевірка вказаного користувачем міста відбувається за допомогою WHERE.

– Фільтрування оголошень за містом відбувається за допомогою SELECT в який входять поля: advertisement.title а також об”єднано з (INNER JOIN) doctor.FirstName, doctor.LastName, doctor.Phone, doctor.Email, об’єднано з (INNER JOIN) freetime.startTime, freetime.endTime, об’єднано з city.City, об”єднано з INNER JOIN type.Type. Перевірка вказаного користувачем міста відбувається за допомогою WHERE.

-Перевірка на правильність введення емейлу це тригер, який відбувається перед внесенням записів в таблицю user за допомогою регулярного виразу, який приймає емейли формату “[example@gmail.com](mailto:example@gmail.com)”.

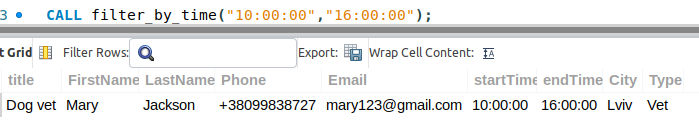
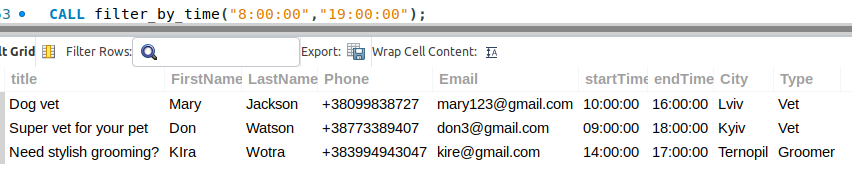
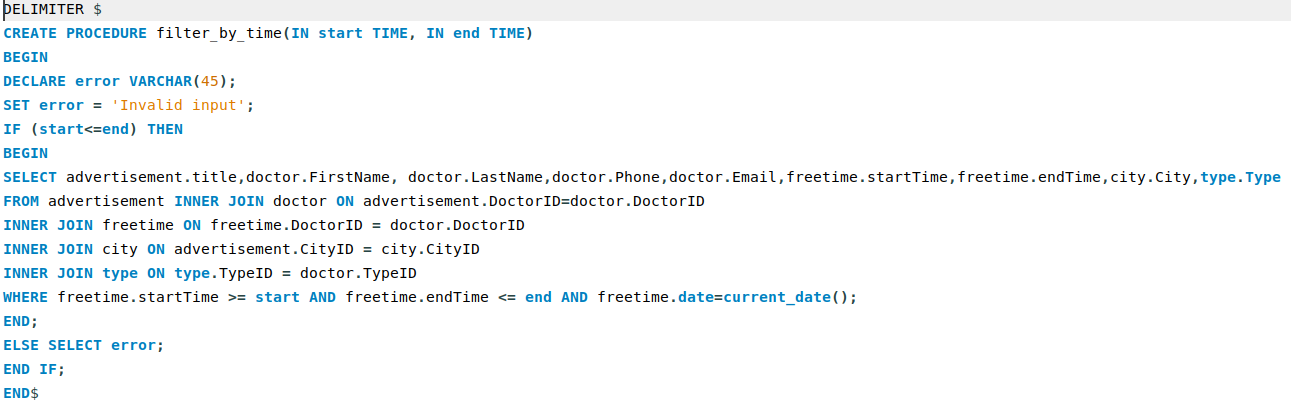
-Перевірка на правильність введення паролю відбувається перед внесенням записів в таблицю user відбувається за допомогою регулярного виразу і функції length(). Щоб пароль прийняло, він має бути велечиною більше 8 символів і містити хоча б одну цифру.

-Валідація на правильність введення часу під час створення “freetime” відбувається за допомогою порівняння полів “startTime” та “endTime”, якщо “startTime” > “endTime”, то виведеться помилка, оскільки таке неможливо.

-Автоматичний статус “unread” під час створення notification – це тригер, який перед внесенням запису в таблицю notification, вказує його статус “Unread”- непрочитано.

ЗАПИТИ ДО БД

* Фільтрування за вільним часом:



* Перевірка формату email адреси:

use cmd;

DELIMITER $

CREATE trigger emailCheck before insert on user

for each row

begin

if not new.Email regexp '^[A-Z0-9.\_%-]+@[A-Z0-9.-]+\.[A-Z]{2,63}$'

then

signal sqlstate '45000'

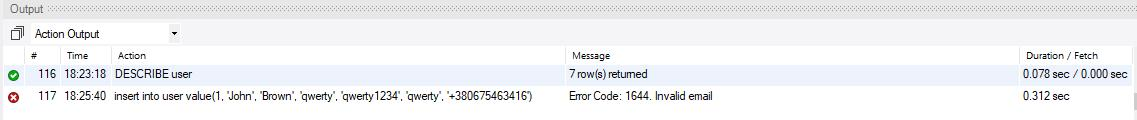
set message\_text = 'Invalid email';

end if;

end $

DELIMITER ;

insert into user value(1, 'John', 'Brown', 'qwerty', 'qwerty1234', 'qwerty', '+380675463416');



* Перевірка пароля на довжину та наявність цифр:

DELIMITER $

CREATE trigger passwordCheck before insert on user

for each row

begin

if not length(new.Password) < 8 or not (new.Password regexp '[0-9]')

then

signal sqlstate '45000'

set message\_text = 'Too short password or not containing numbers!';

end if;

end $

insert into user value(1, 'John', 'Brown', 'qwerty', 'qwerty', 'qwerty@gmail.com', '+380675463416');



* Фільтрування за містом:

DELIMITER $

CREATE PROCEDURE filter\_by\_city(IN city varchar(40))

BEGIN

SELECT advertisement.title,doctor.FirstName, doctor.LastName,doctor.Phone,doctor.Email,freetime.startTime,freetime.endTime,city.City,type.Type

FROM advertisement INNER JOIN doctor ON advertisement.DoctorID=doctor.DoctorID

INNER JOIN freetime ON freetime.DoctorID = doctor.DoctorID

INNER JOIN city ON advertisement.CityID = city.CityID

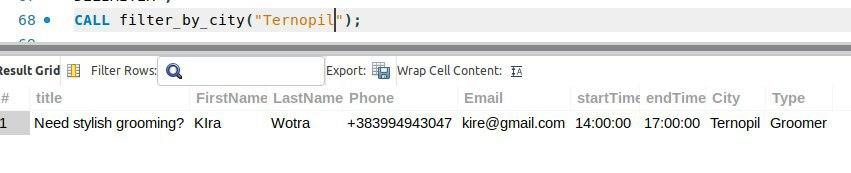
INNER JOIN type ON type.TypeID = doctor.TypeID

WHERE city.City = city;

END$

delimiter ;

CALL filter\_by\_city("10:00:00","16:00:00");



* Перевірка коректного запису вільного часу:

DELIMITER $

CREATE trigger freetimeCheck before insert on freetime

for each row

begin

if not new.startTime < new.endTime

then

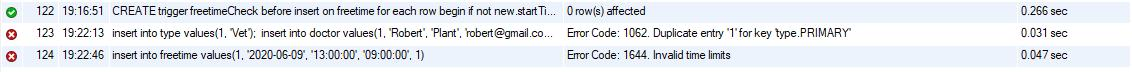
signal sqlstate '45000'

set message\_text = 'Invalid time limits';

end if;

end $

insert into freetime values(1, '2020-06-09', '13:00:00', '09:00:00', 1);



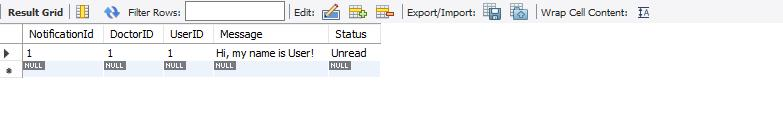
* Автоматичний статус непрочитаного повідомлення

create trigger unreadNotif before insert on notification

for each row

set new.status = 'Unread';

insert into notification value(1, 1, 1, 'Hi, my name is User!', null);



ВИСНОВОК

Виконуючи дану розрахункову роботу, я закріпив навички, набуті на даному курсі. Ознайомився зі створенням ділової моделі бази даних. Для запитів створював тригери та процедури. В запитах використовував регулярні вирази, щоб перевіряти внесені дані.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. <https://dev.mysql.com/doc/workbench/en/>
2. <https://dev.mysql.com/doc/connector-python/en/>
3. <https://dev.mysql.com/doc/dev/mysql-server/latest/PAGE_MYSQL_TEST_RUN.html>
4. <http://bourabai.kz/dbt/MySQL11.htm>
5. <https://metanit.com/sql/mysql/2.5.php>
6. <https://oracle-patches.com/db/3211-проектирование-баз-данных>
7. <https://www.w3resource.com/mysql/encryption-and-compression-functions/md5().php>
8. <https://en.wikipedia.org/wiki/Database_normalization>
9. <http://www.mysql.ru/docs/man/JOIN.html>
10. <https://dev.mysql.com/doc/mysql-em-plugin/en/>
11. <https://dev.mysql.com/doc/refman/8.0/en/mysql-indexes.html>