МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Звіт до лабораторної роботи №2

з дисципліни
“Організація Баз Даних та знань”

Виконала:
ст. гр. КН-210
Заремба Вікторія

Викладач:
Мельникова Н.І.

Львів – 2019

Лабораторна робота №2
з курсу “ОБДЗ”
на тему:
“Створення таблиць бази даних засобами SQL”

Мета роботи: Побудувати даталогічну модель бази даних; визначити типи, розмірності та обмеження полів; визначити обмеження таблиць; розробити SQL запити для створення спроектованих таблиць.

Короткі теоретичні відомості.

Щоб створити нову базу даних у командному рядку клієнта MySQL (mysql.exe) слід виконати команду `CREATE DATABASE`, опис якої подано нижче. Тут і надалі, квадратні дужки позначають необов’язковий аргумент команди, символ "|" позначає вибір між аргументами.

`CREATE {DATABASE | SCHEMA}` `[IF NOT EXISTS]` *`ім’я_бази`*
`[[DEFAULT] CHARACTER SET` *`кодування`*`]`
`[[DEFAULT] COLLATE` *`набір_правил`*`]`

`ім’я_бази` – назва бази даних (латинські літери і цифри без пропусків);
`кодування` – набір символів і кодів (`koi8u`, `latin1`, `utf8`, `cp1250` тощо);
`набір_правил` – правила порівняння рядків символів (див. результат команди `show collation`).

Нижче наведені деякі допоміжні команди для роботи в СУБД MySQL. Кожна команда і кожен запит в командному рядку повинні завершуватись розділяючим символом ";".

1. Перегляд існуючих баз даних:
`SHOW DATABASES`
2. Вибір бази даних для подальшої роботи:
`USE DATABASE` *`ім’я_бази`*
3. Перегляд таблиць в базі даних:
`SHOW TABLES [FOR` *`ім’я_бази`*`]`
4. Перегляд опису таблиці в базі:
`DESCRIBE` *`ім’я_таблиці`*
5. Виконати набір команд з зовнішнього файлу:
`SOURCE` *`назва_файлу`*
6. Вивести результати виконання подальших команд у зовнішній файл:
`\T` *`назва_файлу`*

Для роботи зі схемою бази даних існують такі основні команди:
`ALTER DATABASE` – зміна опису бази даних;
`CREATE TABLE` – створення нової таблиці;
`ALTER TABLE` – зміна структури таблиці;
`DELETE TABLE`– видалення таблиці з бази даних;
`CREATE INDEX` – створення нового індексу (для швидкого пошуку даних);
`DROP INDEX` – видалення індексу;
`DROP DATABASE` – видалення бази даних.

Розглянемо команду створення таблиці в MySQL та її основні аргументи.

`CREATE` `[TEMPORARY]` **`TABLE`** `[IF NOT EXISTS] ім’я_таблиці`
`[(опис_таблиці,...)]`
`[додаткові_параметри] ...`
`[вибірка_даних]`

`опис_таблиці:`
`назва_поля опис_поля`
`| [CONSTRAINT [ім’я_обмеження]] PRIMARY KEY (назва_поля,...)`

```
[тип_обмеження]
| {INDEX|KEY} [ім'я_обмеження] (назва_поля,...)[ тип_обмеження]
| [CONSTRAINT [ім'я_обмеження]] UNIQUE [INDEX|KEY]
[ім'я_обмеження](назва_поля,...) [тип_обмеження]
| {FULLTEXT|SPATIAL} [INDEX|KEY] [ім'я_обмеження]
(назва_поля,...)
[тип_обмеження]
| [CONSTRAINT [ім'я_обмеження]] FOREIGN KEY
[ім'я_обмеження] (назва_поля,...) опис_зв'язку
| CHECK (вираз)
```

опис_поля:

тип_даних [NOT NULL | NULL] [DEFAULT *значення_за_замовчуванням*]
[AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]

опис_зв'язку:

REFERENCES *ім'я_таблиці* (назва_поля, ...)
[ON DELETE дія]
[ON UPDATE дія]

дія:

CASCADE

Одночасне видалення, або оновлення відповідного значення у зовнішній таблиці.

RESTRICT

Аналог NO ACTION. Дія над значенням поля ігнорується, якщо існує відповідне йому значення у зовнішній таблиці. Опція задана за замовчуванням.

SET NULL

При дії над значенням у первинній таблиці, відповідне значення у зовнішній таблиці замінюється на NULL.

додаткові_параметри:

```
{ENGINE|TYPE} [=] тип_таблиці
| AUTO_INCREMENT [=] значення_приросту_лічильника
| AVG_ROW_LENGTH [=] значення
| [DEFAULT] CHARACTER SET [=] кодування
| CHECKSUM [=] {0 | 1}
| [DEFAULT] COLLATE [=] набір_правил
| COMMENT [=] 'коментар до таблиці'
| DATA DIRECTORY [=] 'абсолютний шлях'
| DELAY_KEY_WRITE [=] {0 | 1}
| INDEX DIRECTORY [=] 'абсолютний шлях'
| MAX_ROWS [=] значення
| MIN_ROWS [=] значення
| ROW_FORMAT {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}
```

вибірка_даних:

[IGNORE | REPLACE] [AS] SELECT ... (вибір даних з інших таблиць)

вираз:

Логічний вираз, що повертає TRUE або FALSE.

Опис аргументів:

ім'я_таблиці

Назва таблиці. Або назва_бази.назва_таблиці.

тип_таблиці

В MySQL крім типів таблиць MyISAM та InnoDB існують типи MEMORY, BDB, ARCHIVE тощо.

тип_обмеження

Задає тип індексу для ключового поля: USING {BTREE | HASH | RTREE}.

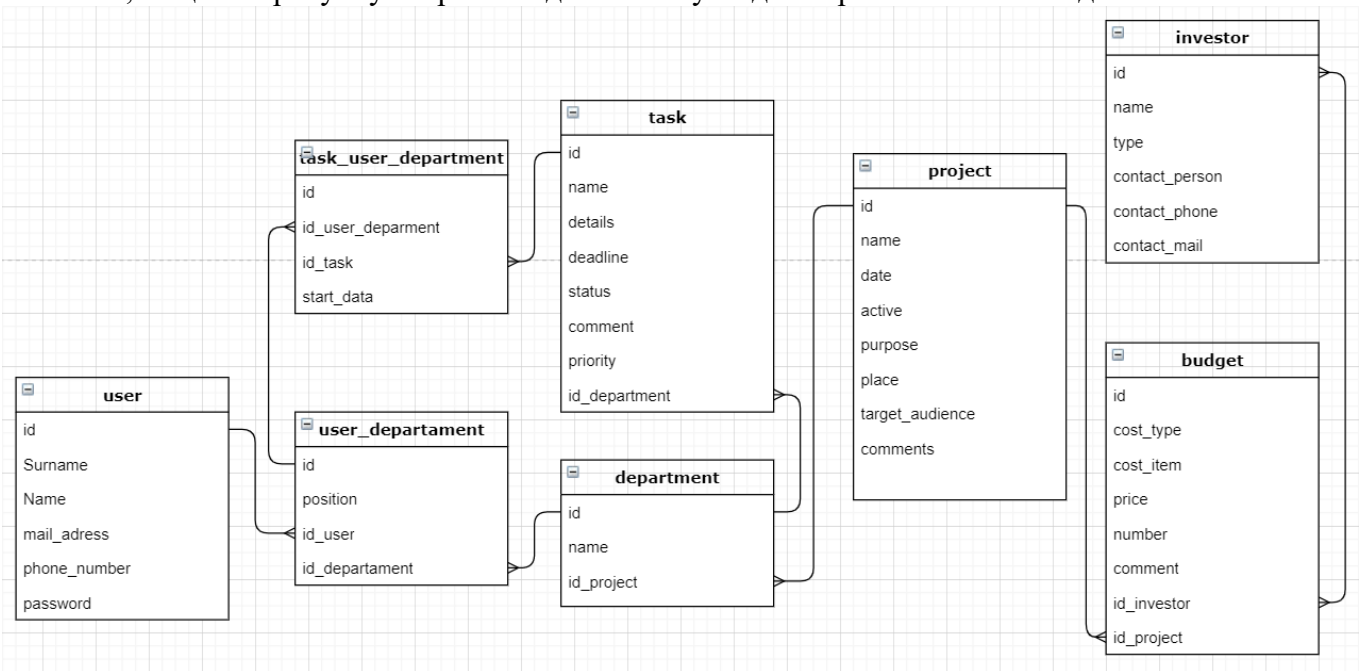
TEMPORARY
Створення тимчасової таблиці, яка буде знищена після завершення зв'язку з сервером.
CONSTRAINT
Вказує на початок оголошення PRIMARY KEY, UNIQUE, або FOREIGN KEY обмеження.
NULL | NOT NULL
Директива, що дозволяє/забороняє null-значення для даного поля.
PRIMARY KEY
Вказує, що дане поле буде первинним ключем в таблиці.
UNIQUE
Вказує на те, що в даному полі будуть зберігатися унікальні значення.
FOREIGN KEY ... REFERENCES
Створює зовнішній ключ, зв'язаний із вказаним полем (полями).
AVG_ROW_LENGTH
Приблизне значення середньої довжини рядків зі змінною довжиною.
DATA DIRECTORY
Вказує шлях, за яким таблиця має зберігатись у файловій системі.
CHECKSUM
Якщо параметр = 1, то для рядків таблиці буде рахуватись контрольна сума. Це сповільнює оновлення таблиці, але робить легшим пошук пошкоджених таблиць.
ROW_FORMAT
Вказує на спосіб зберігання рядків таблиці (залежно від типу таблиці).
FULLTEXT|SPATIAL
Тип індексу (повнотекстовий/просторовий; тільки для таблиць типу MyISAM).
Основні типи даних у СУБД MySQL:

Тип даних	Опис
TINYINT[(*k*)] [UNSIGNED]	Ціле число з *k*-біт: -127 .. 128. UNSIGNED: 0 .. 255.
BOOL	Логічний тип (1-бітне число). Число 0 – фальш, відмінне від нуля – істина.
SMALLINT[(*k*)] [UNSIGNED]	Ціле число з *k*-біт: -32768 .. 32767. UNSIGNED: 0 .. 65535.
MEDIUMINT[(*k*)] [UNSIGNED]	Ціле число з *k*-біт: -8388608 .. 8388607. UNSIGNED: 0 .. 16777215.
INT[(*k*)] [UNSIGNED]	Ціле число з *k*-біт: -2147483648 .. 2147483647. UNSIGNED: 0 .. 4294967295.
BIGINT[(*k*)] [UNSIGNED]	-9223372036854775808 .. 9223372036854775807. UNSIGNED: 0 .. 18446744073709551615.
SERIAL	Синонім для типу BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
FLOAT[(*n*,*m*)] [UNSIGNED]	Число з плаваючою крапкою, де *n* – кількість всіх цифр, *m* – кількість цифр після крапки. Від -3.402823466E+38 до -1.175494351E-38 UNSIGNED: 1.175494351E-38 .. 3.402823466E+38
DOUBLE[(*n*,*m*)] [UNSIGNED]	Від -1.7976931348623157E+308 до -2.2250738585072014E-308 UNSIGNED: від 2.2250738585072014E-308 до 1.7976931348623157E+308.
DECIMAL[(*n*[,*m*])] [UNSIGNED]	Число з фіксованою крапкою. *n* – кількість цифр (максимально – 65), *m* – кількість цифр після крапки

	(максимально – 30, за замовчуванням – 0). `UNSIGNED`: від'ємні значення заборонені.
`DATE`	Дата. Від "1000-01-01" до "9999-12-31".
`DATETIME`	Дата і час. Від "1000-01-01 00:00:00" до "9999-12-31 23:59:59".
`TIMESTAMP`	Часова мітка. Може присвоюватись автоматично. Від "1970-01-01 00:00:01" до "2038-01-09 03:14:07"
`TIME`	Час у форматі "HH:MM:SS" (рядок або число).
`CHAR[(n)]`	Рядок з *n*-символів (макс. – 255, за замовчуванням – 1).
`VARCHAR(n)`	Рядок змінної довжини. Для кодування *utf8* максимальна довжина складає 21844 символи.
`TEXT(n)`	Рядок змінної довжини. Максимальна кількість однобайтових символів – 65535.
`MEDIUMTEXT`	16777215 однобайтових символів (16 Мб тексту).
`BLOB`	Бінарні дані (65535 байт).
`MEDIUMBLOB`	Бінарні дані (16 Мб)
`LONGBLOB`	Бінарні дані (4 Гб, залежно від налаштувань системи)
`ENUM('знач1','знач2',...)`	Перелік значень. Зберігається лише одне.
`SET('знач1','знач2',...)`	Множина значень. Зберігається одне, або більше (максимально – 64).

Хід Роботи

Даталогічна модель вимагає визначення конкретних полів бази даних, їхніх типів, обмежень на значення, тощо. На рисунку зображено даталогічну модель проектованої бази даних



Створимо нову базу даних, виконавши такі команди

```
CREATE DATABASE PM_System CHARACTER SET utf8;

CREATE TABLE PM_System.user(
    id_user INT UNSIGNED NOT NULL AUTO_INCREMENT,
    surname varchar(30) NOT NULL,
    name varchar(40) NOT NULL,
    mail_adress varchar(50) NOT NULL,
    phone_number char(12) NOT NULL,
    password varchar(99) NOT NULL,
    PRIMARY KEY (id_user)
);

CREATE TABLE PM_System.project(
    id_project INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name varchar(99) NOT NULL,
    date DATE NOT NULL,
    active BOOL NOT NULL,
    purpose TEXT,
    place TEXT,
    target_audience VARCHAR(100),
    comments TEXT,
    PRIMARY KEY (id_project)
);

CREATE TABLE PM_System.department(
    id_department INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    id_project INT UNSIGNED NOT NULL,
    PRIMARY KEY (id_department),
    CONSTRAINT fk_department_project FOREIGN KEY (id_project)
        REFERENCES PM_System.project(id_project) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE PM_System.task(
    id_task INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name TEXT NOT NULL,
    details TEXT,
    deadline DATE,
    status BOOL,
    comment TEXT,
    priority TINYINT,
    id_department INT UNSIGNED NOT NULL,
    PRIMARY KEY (id_task),
    CONSTRAINT fk_task_department FOREIGN KEY (id_department)
        REFERENCES PM_System.department(id_department) ON DELETE NO ACTION ON UPDATE NO
ACTION
);

CREATE TABLE PM_System.user_department(
    id_user_department INT UNSIGNED NOT NULL AUTO_INCREMENT,
    position VARCHAR(99) NOT NULL,
    id_user INT UNSIGNED NOT NULL,
    id_department INT UNSIGNED NOT NULL,
    PRIMARY KEY (id_user_department),
    CONSTRAINT fk_user_user_department FOREIGN KEY(id_user)
        REFERENCES PM_System.user(id_user) ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_department_user_department FOREIGN KEY(id_department)
        REFERENCES  PM_System.department(id_department) ON DELETE NO ACTION ON UPDATE NO
ACTION
```

```
);

CREATE TABLE PM_System.task_user_department(
    id_task_user_department INT UNSIGNED NOT NULL AUTO_INCREMENT,
    id_user_department INT UNSIGNED NOT NULL,
    id_task INT UNSIGNED NOT NULL,
    start_data DATE NOT NULL,
    PRIMARY KEY (id_task_user_department),
    CONSTRAINT fk_tud_ud FOREIGN KEY(id_user_department)
        REFERENCES PM_System.user_department(id_user_department) ON UPDATE NO ACTION ON
DELETE NO ACTION,
    CONSTRAINT fk_tud_task FOREIGN KEY(id_task)
        REFERENCES  PM_System.task(id_task) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE PM_System.budget(
    id_budget INT UNSIGNED NOT NULL AUTO_INCREMENT,
    cost_type VARCHAR(250) NOT NULL,
    cost_item VARCHAR(250) NOT NULL,
    price INT UNSIGNED NOT NULL,
    number INT UNSIGNED NOT NULL,
    comment TEXT,
    id_project INT UNSIGNED NOT NULL,
    PRIMARY KEY (id_budget),
    CONSTRAINT fk_budget_project FOREIGN KEY (id_project)
        REFERENCES PM_System.project(id_project) ON DELETE NO ACTION ON UPDATE NO ACTION
);

CREATE TABLE PM_System.investor(
    id_investor INT UNSIGNED NOT NULL AUTO_INCREMENT,
    name VARCHAR(250) NOT NULL,
    type VARCHAR(250),
    contact_person VARCHAR(250) NOT NULL,
    contact_phone VARCHAR(12),
    contact_mail VARCHAR(12),
    PRIMARY KEY (id_investor)
);

CREATE TABLE PM_System.budget_investor(
    id_budget INT UNSIGNED NOT NULL,
    id_investor INT UNSIGNED NOT NULL,
    sum FLOAT NOT NULL,
    PRIMARY KEY (id_investor, id_budget),
    CONSTRAINT fk_bud_inv FOREIGN KEY (id_budget)
        REFERENCES PM_System.budget(id_budget) ON DELETE NO ACTION ON UPDATE NO ACTION,
    CONSTRAINT fk_inv_bud FOREIGN KEY (id_investor)
        REFERENCES PM_System.investor(id_investor) ON DELETE NO ACTION ON UPDATE NO ACTION
);
```

Висновок: *на цій лабораторній роботі було завершено моделювання і засобами SQL створено базу даних, що складається з восьми таблиць.*