МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»
Кафедра систем штучного інтелекту


# Лабораторна робота №7
З курсу “Організація баз даних та знань”

Виконав:
студент групи КН-210
Марій Павло

Викладач:
Мельникова Наталя Іванівна

Львів – 2020

**Тема:** Запити на вибір даних з таблиць бази даних.

**Мета:** Розробити SQL запити відбору даних з одиничних та з'єднаних таблиць, в тому числі з використанням підзапитів, натурального, умовного та лівого з'єднання, із застосуванням у критеріях вибірки функцій та операторів, в т. ч. LIKE, BETWEEN, IS NULL, IS NOT NULL, IN (…), NOT IN (…), ALL, SOME, ANY, EXISTS.

## Теоретичні відомості

Для вибирання даних з таблиць використовується директива **SELECT**, яка може містити інші директиви **SELECT** (підзапити, або вкладені запити) та директиви з'єднання таблиць.

**SELECT**
[ALL | DISTINCT | DISTINCTROW ]
[STRAIGHT_JOIN]
[SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
елемент_вибірки [, елемент_вибірки ...]
[**FROM** перелік_таблиць]
[**WHERE** умова_відбору]
[**GROUP BY** {ім'я_поля | синонім | позиція_поля}
[ASC | DESC], ...]
[**HAVING** умова_відбору]
[**ORDER BY** {ім'я_поля | синонім | позиція_поля}
[ASC | DESC], ...]
[**LIMIT** {к-сть_рядків [OFFSET зміщення]}
[**PROCEDURE** ім'я_процедури(аргументи)]
[**INTO** OUTFILE 'ім'я_файлу' опції_експорту
| INTO DUMPFILE 'ім'я_файлу'
| INTO змінна [, змінна]]

**Параметри**:

SELECT

Вказує поля, константи та вирази, що будуть відображатися у результатах запиту. Директива вимагає чіткого дотримання порядку ключових слів FROM, WHERE і т.д.

елемент_вибірки

Вказує елемент, який буде включатися в результати відбору. Такими елементами можуть бути: ім’я поля, константа або вираз. Кожному елементу можна присвоїти ім’я-псевдонім, яке буде відображатись у результатах запиту. Для цього після назви елемента слід дописати AS псевдонім.

перелік_таблиць

Назви таблиць, з яких здійснюється вибір значень. Тут можна задавати синоніми назвам таблиць (ім’я_таблиці AS синонім), використовувати підзапити SELECT для формування таблиці з вказаним синонімом, з’єднувати декілька таблиць.

WHERE

Вказує критерії порівняння (або підзапити) для відбору рядків.

GROUP BY

Групує (і одночасно сортує) рядки за вказаними полями. Поля можна вказувати за іменами, синонімами або порядковими номерами в таблиці.

ORDER BY

Сортує рядки за вказаними полями. За замовчуванням – за зростанням значень (ASC).

HAVING

Дає можливість застосування до значень полів агрегатних функцій (COUNT, AVG, MIN, MAX тощо) при відборі чи групуванні рядків. Після слова WHERE ці функції не працюють, однак у всіх інших випадках слід використовувати саме WHERE.

LIMIT

Обмежує кількість рядків, повернутих в результаті запиту.

OFFSET

Вказує зміщення для LIMIT – з якого рядка в результатах запиту почати відбирати потрібну кількість рядків.

PROCEDURE

Задає назву збереженої процедури, яка повинна обробляти результат запиту.

INTO

Вказує місце, куди будуть збережені результати запиту. Це може бути як зовнішній файл, так і параметри чи змінні, визначені користувачем. Кількість змінних має бути рівна кількості полів у результаті.

DISTINCT | DISTINCTROW

Видалення з результату рядків-дублікатів. За замовчуванням вибираються всі рядки.

STRAIGHT_JOIN

Опція, яка строго задає порядок вибирання кортежів зі з'єднуваних таблиць в порядку переліку таблиць. (Оптимізатор запитів MySQL іноді змінює цей порядок.)

SQL_CACHE | SQL_NO_CACHE

Явним чином вмикає/вимикає зберігання результатів запиту у кеші запитів MySQL. За замовчуванням, кешування запитів залежить від системної змінної query_cache_type.

SQL_CALC_FOUND_ROWS

Вказує, що при виконанні запиту слід обчислити загальну кількість рядків в результаті, ігноруючи опцію обмеження LIMIT. Цю кількість рядків потім можа отримати командою SELECT FOUND_ROWS().

Для вибору записів зі з'єднаних таблиць використовується директива SELECT разом із директивами JOIN у переліку таблиць. Наприклад:

```
SELECT * FROM author INNER JOIN comment
ON author.authorID = comment.authorID;
```

## Хід роботи

Для вивчення роботи директив вибору даних з таблиць розробимо та виконаємо такі запити над таблицями user, Role, comment.

1. Показати email заданого користувача.
2. Показати користувачів і їхні коментарі (ліве з'єднання таблиць).
3. Показати перелік користувачів у групі Guests (натуральне з'єднання).
4. Показати всі коментарі користувачів з груп Guests та Group2 (умовне з'єднання).
5. Показати останні 3 оголошення користувачів з груп Group1 та Group2 (підзапит).
6. Визначити користувачів, які не написали жодного повідомлення.
7. Визначити користувачів, паролі яких не відповідають вимогам безпеки (менші за 8 символів або не містять цифр).

Вміст таблиць.

user:

| id | name | surname | telephone_number | email |
|---|---|---|---|---|
| 1 | name1 | surname1 | 0987654321 | email1@gmail.com |
| 2 | name2 | surname2 | 0987654322 | email2@gmail.com |
| 3 | name3 | surname3 | 0987654323 | email3@gmail.com |
| 4 | name4 | surname4 | 0987654324 | email4@gmail.com |
| 5 | name5 | surname5 | 0987654325 | email5@gmail.com |
| 6 | name6 | surname6 | 0987654326 | email6@gmail.com |
| 7 | name7 | surname7 | 0987654327 | email7@gmail.com |
| 8 | name8 | surname8 | 0987654328 | email8@gmail.com |
| 9 | name9 | surname9 | 0987654329 | email9@gmail.com |
| 10 | name10 | surname10 | 0987654330 | email10@gmail.com |
| NULL | NULL | NULL | NULL | NULL |

comment:

| id | text | advertisement_id | user_id |
|---|---|---|---|
| 1 | "comment1" | 2 | 4 |
| 2 | "comment2" | 2 | 4 |
| 3 | "comment3" | 4 | 3 |
| 4 | "comment4" | 4 | 3 |
| 5 | "comment5" | 5 | 1 |
| NULL | NULL | NULL | NULL |

1. Знайдемо email користувача з номером 9.

Код скрипту:

```
SELECT email FROM mydb.user WHERE id=9;
```

Результат:

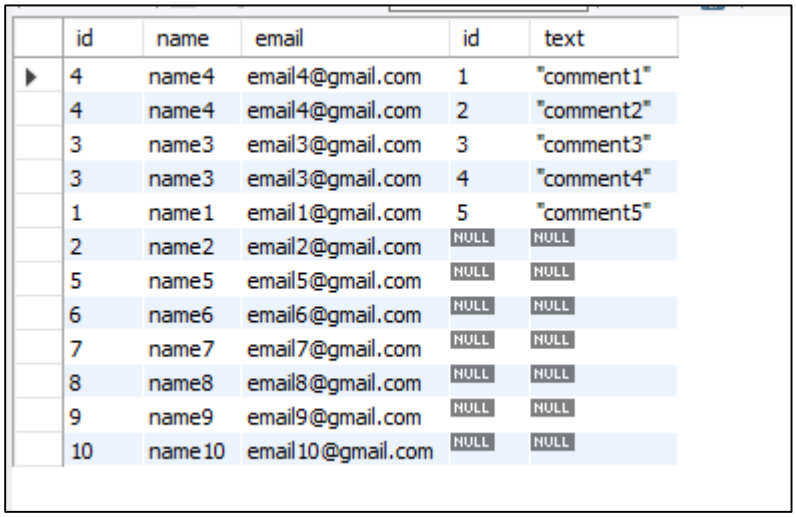
| email |
|---|
| email9@gmail.com |

2. Виберемо всіх користувачів з їхніми коментарями. Для цього потрібно виконати ліве з’єднання. Для користувачів, які не написали жодного коментаря в результатах буде відображено порожні значення.

Код скрипту:

```
SELECT user.id, user.name, user.email,
comment.id, comment.text
FROM mydb.user LEFT JOIN mydb.comment ON
user.id = comment.user_id;
```

Результат:



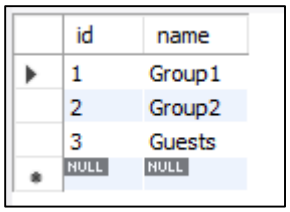
| id | name | email | id | text |
|---|---|---|---|---|
| 4 | name4 | email4@gmail.com | 1 | "comment1" |
| 4 | name4 | email4@gmail.com | 2 | "comment2" |
| 3 | name3 | email3@gmail.com | 3 | "comment3" |
| 3 | name3 | email3@gmail.com | 4 | "comment4" |
| 1 | name1 | email1@gmail.com | 5 | "comment5" |
| 2 | name2 | email2@gmail.com | NULL | NULL |
| 5 | name5 | email5@gmail.com | NULL | NULL |
| 6 | name6 | email6@gmail.com | NULL | NULL |
| 7 | name7 | email7@gmail.com | NULL | NULL |
| 8 | name8 | email8@gmail.com | NULL | NULL |
| 9 | name9 | email9@gmail.com | NULL | NULL |
| 10 | name10 | email10@gmail.com | NULL | NULL |

3. Створимо нову таблицю Group, яка буде групувати користувачів у певні групи, і заповнимо її, створивши 3 групи.

Код скрипту:

```
CREATE TABLE IF NOT EXISTS `mydb`.`group` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE INDEX `id_UNIQUE`  (`id` ASC) )
ENGINE = InnoDB;
```

Результат:



| id | name |
|---|---|
| 1 | Group1 |
| 2 | Group2 |
| 3 | Guests |
| NULL | NULL |

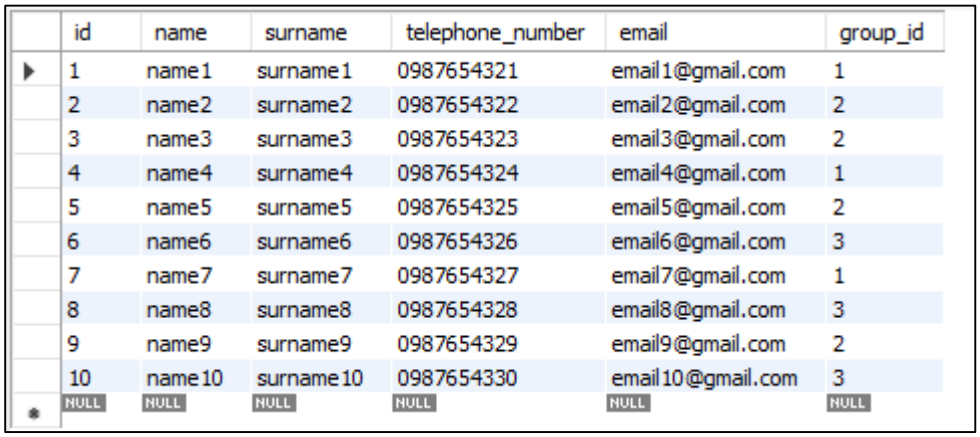
Далі додамо користувачів в певні групи. Для цього додамо таблиці user поле group_id.

Код скрипту:

```
ALTER TABLE mydb.user
        ADD COLUMN group_id INT NOT NULL,
        ADD CONSTRAINT user_group FOREIGN KEY (group_id)
```

REFERENCES mydb.group (id) ON DELETE NO ACTION;

Я додав поле group_id і створив зовнішній ключ. Внесу кілька значень:



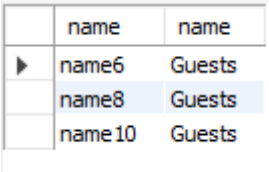
| id | name | surname | telephone_number | email | group_id |
|---|---|---|---|---|---|
| 1 | name1 | surname1 | 0987654321 | email1@gmail.com | 1 |
| 2 | name2 | surname2 | 0987654322 | email2@gmail.com | 2 |
| 3 | name3 | surname3 | 0987654323 | email3@gmail.com | 2 |
| 4 | name4 | surname4 | 0987654324 | email4@gmail.com | 1 |
| 5 | name5 | surname5 | 0987654325 | email5@gmail.com | 2 |
| 6 | name6 | surname6 | 0987654326 | email6@gmail.com | 3 |
| 7 | name7 | surname7 | 0987654327 | email7@gmail.com | 1 |
| 8 | name8 | surname8 | 0987654328 | email8@gmail.com | 3 |
| 9 | name9 | surname9 | 0987654329 | email9@gmail.com | 2 |
| 10 | name10 | surname10 | 0987654330 | email10@gmail.com | 3 |
| NULL | NULL | NULL | NULL | NULL | NULL |

Виберемо користувачів з групи Guests. Для цього виконаємо умовне з’єднання таблиць user і group за атрибутом group_id, використовуючи директиву INNER JOIN.

Код скрипту:

```
SELECT user.name, group.name
FROM mydb.user INNER JOIN mydb.group ON group.id = user.group_id
WHERE group.name = 'Guests';
```

Результат:



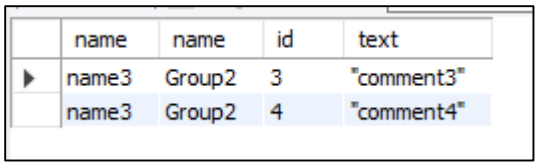
| name | name |
|---|---|
| name6 | Guests |
| name8 | Guests |
| name10 | Guests |

4. Виберемо всі коментарі користувачів з групи Guests та Group2. Для цього виконаємо умовне з’єднання таблиць user і group за атрибутом group_id, та таблиці comment використовуючи директиву INNER JOIN.

Код скрипту:

```
SELECT user.name, group.name, comment.id, comment.text
FROM (mydb.user INNER JOIN mydb.group) INNER JOIN mydb.comment
ON group.id = user.group_id
AND comment.user_id = user.id
WHERE group.name IN ('Guests', 'Group2');
```

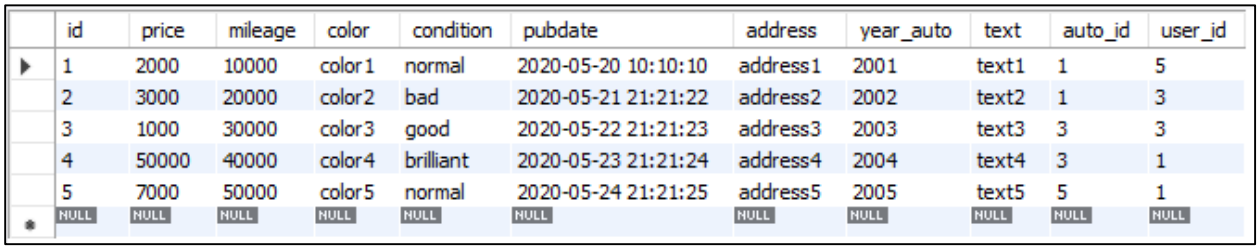
Результат:



| name | name | id | text |
|---|---|---|---|
| name3 | Group2 | 3 | "comment3" |
| name3 | Group2 | 4 | "comment4" |

Тут показані коментарі користувачів, які належать до груп Guests та Group2. Користувачі, які не залишали коментарів, тут не показуються.

5. Виберемо останні 3 оголошення, які розміщували користувачі, які належать до Group1 та Group2. Для цього замість директиви JOIN використаємо підзапит в умові відбору, який буде повертати номери потрібних груп. Дані таблиці advertisement:

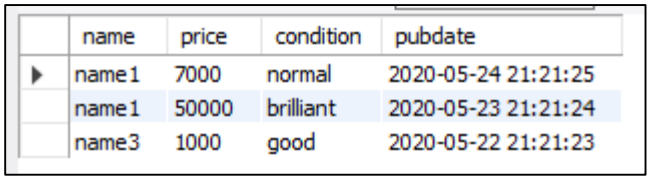


| id | price | mileage | color | condition | pubdate | address | year_auto | text | auto_id | user_id |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2000 | 10000 | color1 | normal | 2020-05-20 10:10:10 | address1 | 2001 | text1 | 1 | 5 |
| 2 | 3000 | 20000 | color2 | bad | 2020-05-21 21:21:22 | address2 | 2002 | text2 | 1 | 3 |
| 3 | 1000 | 30000 | color3 | good | 2020-05-22 21:21:23 | address3 | 2003 | text3 | 3 | 3 |
| 4 | 50000 | 40000 | color4 | brilliant | 2020-05-23 21:21:24 | address4 | 2004 | text4 | 3 | 1 |
| 5 | 7000 | 50000 | color5 | normal | 2020-05-24 21:21:25 | address5 | 2005 | text5 | 5 | 1 |
| NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL | NULL |

Код скрипту:

```
SELECT user.name, advertisement.price, advertisement.condition,
advertisement.pubdate
FROM mydb.user INNER JOIN mydb.advertisement
ON user.id = advertisement.user_id
WHERE user.group_id IN (SELECT group.id FROM mydb.group
WHERE group.name IN ('Group1', 'Group2'))
ORDER BY advertisement.pubdate DESC LIMIT 3;
```

Результат:



| name | price | condition | pubdate |
|---|---|---|---|
| name1 | 7000 | normal | 2020-05-24 21:21:25 |
| name1 | 50000 | brilliant | 2020-05-23 21:21:24 |
| name3 | 1000 | good | 2020-05-22 21:21:23 |

Тут показані останні опубліковані 3 оголошення, які належать користувачам, які належать до Group1 або Group2.

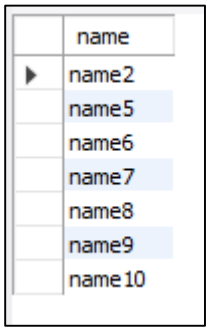
6. Визначимо користувачів, які не написали жодного коментаря.

Код скрипту:

```
SELECT user.name FROM mydb.user
WHERE NOT EXISTS
(SELECT * FROM mydb.comment WHERE
```

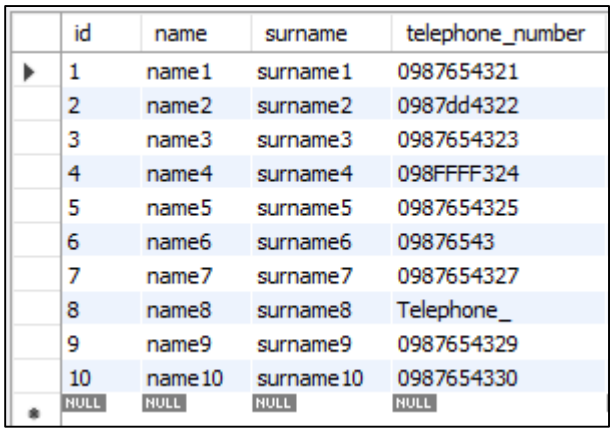
comment.user_id = user.id);

Результат:



| name |
|---|
| name2 |
| name5 |
| name6 |
| name7 |
| name8 |
| name9 |
| name10 |

7. Знайдемо всіх користувачів, які ввели невірний номер телефону (Вірний номер – 10 цифр, без букв).
Для цього спершу модифікую дані, і введу кілька невірних значень:



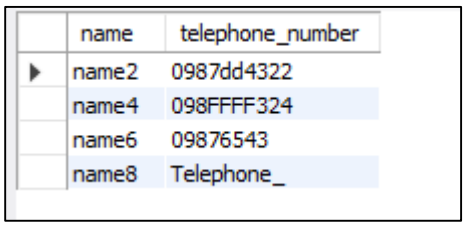
| id | name | surname | telephone_number |
|---|---|---|---|
| 1 | name1 | surname1 | 0987654321 |
| 2 | name2 | surname2 | 0987dd4322 |
| 3 | name3 | surname3 | 0987654323 |
| 4 | name4 | surname4 | 098FFFF324 |
| 5 | name5 | surname5 | 0987654325 |
| 6 | name6 | surname6 | 09876543 |
| 7 | name7 | surname7 | 0987654327 |
| 8 | name8 | surname8 | Telephone_ |
| 9 | name9 | surname9 | 0987654329 |
| 10 | name10 | surname10 | 0987654330 |
| NULL | NULL | NULL | NULL |

Код скрипту:

```
SELECT user.name, user.telephone_number
FROM mydb.user
WHERE user.telephone_number NOT REGEXP '[0-9]{10}';
```

Тут регулярний вираз означає, що telephone_number повинен складатися з цифр ( [0-9] ), і його довжина повинна бути 10 символів ( {10} ).

Результат:



| name | telephone_number |
|---|---|
| name2 | 0987dd4322 |
| name4 | 098FFFF324 |
| name6 | 09876543 |
| name8 | Telephone_ |

**Висновок**: на цій лабораторній роботі було вивчено методи вибору даних зі з’єднаних таблиць БД засобами SQL та виконано запити до бази даних з використанням директив SELECT та JOIN, а також складних критеріїв в умові вибірки.