**МІНІСТЕРСТВО ОСВІТИ І НАУКИ**
**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**
**Інститут комп'ютерних наук та інформаційних технологій**
**Кафедра систем штучного інтелекту**


# Звіт до практичної роботи №7

**з дисципліни “ ОБДЗ ”**

**На тему: «Запити на вибір даних з таблиць бази даних»**

**Виконав**:
ст. гр. КН-211
Качмарик Віктор
**Викладач:**
Якимишин Х. М.

Львів – 2020

**Мета роботи:** 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.

Короткі 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;
```

Параметри дерективи:

INNER JOIN

Внутрішнє з'єднання. Результати вибору будуть містити тільки ті рядки, для яких існують один або більше відповідних рядків з іншої таблиці. В MySQL – є синонімом директиви CROSS JOIN. Слід зауважити, що вибір рядків директивою SELECT з кількох таблиць, вказаних через кому, є аналогічним до явного використання директиви INNER JOIN. В обох випадках MySQL формує декартовий добуток усіх кортежів, і з результату вибирає лише ті, для яких виконується умова відбору (порівняння) ON.

LEFT JOIN

Вказує на те, що результати вибору будуть містити всі рядки з таблиці, яка стоїть зліва від слова JOIN і тільки відповідні їм рядки з таблиці справа (ті, для яких виконується вказана умова). Якщо відповідний рядок відсутній, виводяться значення NULL.

RIGHT JOIN

Вказує на те, що результати вибору будуть містити всі рядки з таблиці, яка вказана справа від JOIN і тільки відповідні їм рядки з таблиці зліва. Для сумісності на практиці використовують в основному LEFT JOIN.

ON

умова Вказує поля, за якими слід з'єднувати таблиці. Замість ON можна також використовувати USING перелік_спільних_полів. В цьому випадку спільне поле буде відображене в результатах запиту лише один раз.

NATURAL JOIN

Еквівалент внутрішньому з'єднанню за всіма однаковими полями (з опцією USING *).

У таблиці нижче описано основні функції порівняння, які можна використовувати при формуванні складних критеріїв вибору.

| Функція | Опис |
|---|---|
| STRCMP(*рядок1*, *рядок2*) | Порівнює два рядки. Повертає значення 0 (False) якщо рядки однакові, -1 якщо перший рядок менший за другий, і 1 (True) в усіх інших випадках. |
| LIKE *рядок* | Порівняння з рядком-шаблоном. В шаблоні можна використовувати знаки % (довільні символи) і _ (довільний символ). |
| REGEXP *рядок* | Порівняння з рядком з використанням регулярних виразів. Функція-синонім – RLIKE. |
| MATCH (*поля*) AGAINST (*рядок*) | Здійснює пошук рядка у вказаних текстових полях таблиці. (Тільки для MyISAM-таблиць.) |
| BETWEEN ... AND ... | Повертає 1, якщо значення належить даному діапазону. |
| NOT BETWEEN ... AND ... | Повертає 1, якщо значення не належить діапазону. |
| IN(*арг1*, *арг2*, ...) | Перевірка належності множині. Повертає 1, якщо значення співпадає хоча б із одним аргументом, і 0 – у протилежному випадку. Повертає NULL, якщо значення є NULL, або якщо співпадіння не знайдено, а один із аргументів є NULL. |

| | |
|---|---|
| NOT IN(*арг1*, *арг2*, ...) | Повертає 1, якщо значення не міститься у множині аргументів, і 0 – у протилежному випадку. Повертає NULL аналогічно до функції IN(). |
| IS NULL, IS NOT NULL | Перевірка визначеності значення. |
| LEAST(*арг1*, *арг2*, ...) | Повертає мінімальне значення серед аргументів. Повертає NULL, якщо хоча б один із аргументів є NULL. |
| GREATEST(*арг1*, *арг2*, ...) | Повертає максимальне значення серед аргументів. Повертає NULL, якщо хоча б один із аргументів є NULL. |

Для формування критеріїв вибору та підзапитів також використовують наступні оператори порівняння:

=

Оператор перевірки рівності двох виразів. Якщо відбувається порівняння двох не NULL значень, то повертає значення 1 (True) коли обидва вирази рівні, інакше результатом є значення 0 (False). Якщо хоча б один з виразів приймає значення NULL, то результатом є значення NULL.

<=>

Перевірка рівності виразів, яке враховує NULL значення. Повертає 1, якщо обидва вирази приймають значення NULL, або рівні значення. Повертає 0, якщо один із виразів приймає значення NULL, або значення виразів не рівні.

>, >=

Порівняння двох виразів. Результатом є 1, якщо ліве значення більше (більше рівне) ніж праве, інакше результатом є 0. Якщо хоча б один з виразів приймає значення NULL, то результатом теж стає NULL.

<, <=

Порівняння двох виразів. Результатом є 1, якщо ліве значення менше (менше рівне) ніж праве, інакше результатом є 0. Якщо хоча б один з виразів приймає значення NULL, то результатом теж є NULL.

!=, <>

Перевірка на не рівність. Результат набуває значення 1, якщо ліве значення менше або більше ніж праве, інакше результатом є 0. Якщо хоча б один з виразів приймає значення NULL, то результатом теж є NULL.

ALL, SOME, ANY

Оператори, які можна використовувати після операторів порівняння. Задають необхідність виконання оператора хоча б для одного (SOME, ANY) чи всіх (ALL) елементів, отриманих в результаті підзапиту. На відміну від функцій IN(), NOT IN() оператори не працюють зі списками значень.

[NOT] EXISTS

Оператор, який використовують після ключового слова WHERE. Повертає 1, якщо підзапит повертає хоча б одне визначене значення, і 0 – у протилежному випадку.

## Хід роботи

1. Знайдемо пароль користувача з номером 7.

```
SELECT password
FROM customer WHERE customer_id = 7;
```

Результат запиту:

```
+----------+
| password |
+----------+
| roman    |
+----------+
```

2. Виберемо усі аптеки з їх ліками. Для цього потрібно виконати ліве з’єднання. Для аптек, в яких немає ліків у результатах буде відображено порожні значення.

```
SELECT pharmacy.pharmacy_id, pharmacy.pharmacy_name, pharmacy.street,
medicine.medicine_name, medicine.quantity
FROM pharmacy LEFT JOIN medicine
ON pharmacy.pharmacy_id = medicine.pharmacy_id;
```

Результат виконання:

| pharmacy_id | pharmacy_name | street | medicine_name | quantity |
|---|---|---|---|---|
| 1 | Phar1 | Zelena | Pentalgin | 10 |
| 1 | Phar1 | Zelena | Mezym | 7 |
| 2 | Phar2 | Zelena | Pentalgin | 5 |
| 2 | Phar2 | Zelena | Sorbex | 6 |
| 3 | Phar3 | Zelena | NULL | NULL |
| 4 | Phar4 | Lypneva | Mezym | 8 |
| 5 | Phar5 | Lyubinska | Sorbex | 10 |
| 6 | Phar6 | Chornovola | Card | 9 |
| 7 | Phar7 | Lypneva | Ramtex | 7 |
| 8 | Phar8 | Lypneva | NULL | NULL |
| 9 | Phar9 | Lyubinska | NULL | NULL |
| 10 | Phar10 | Zelena | NULL | NULL |

3. Виберемо ліки виду “Drops”. Для цього виконаємо умовне з’єднання таблиць medicine і all_medicine за атрибутом medicine_name, використовуючи директиву INNER JOIN.

```
SELECT medicine.medicine_name, all_medicine.medicine_description
FROM medicine INNER JOIN all_medicine
ON medicine.medicine_name = all_medicine.medicine_name
WHERE all_medicine.medicine_description = "Drops";
```

Результат виконання:

| medicine_name | medicine_description |
|---|---|
| Card | Drops |
| Sorbex | Drops |
| Sorbex | Drops |

4. Виберемо всі замовлення які відбулися в усіх аптеках на вулиці зеленій. Для цього виконаємо умовне з’єднання таблиць order і customer за атрибутом customer_id, та таблиці pharmacy використовуючи директиву INNER JOIN.

```
SELECT  `order`.order_id, customer.first_name, pharmacy.pharmacy_name, `order`.order_date
FROM (`order` INNER JOIN customer) INNER JOIN pharmacy
ON customer.customer_id = `order`.customer_id
AND pharmacy.pharmacy_id = `order`.pharmacy_id
WHERE pharmacy.street = "Zelena";
```

Результат виконання:

| order_id | first_name | pharmacy_name | street | order_date |
|---|---|---|---|---|
| 1 | Viktor | Phar1 | Zelena | NULL |
| 3 | Vlad | Phar2 | Zelena | NULL |
| 2 | Andrew | Phar3 | Zelena | NULL |

5. Виберемо 2 останні ліки які є таблетками. Для цього замість дирактиви JOIN використаємо підзапит в умові відбору, який буде повертати номери потрібних груп.

```
SELECT pharmacy.pharmacy_name, medicine.medicine_name, medicine.quantity
FROM medicine INNER JOIN pharmacy
ON pharmacy.pharmacy_id = medicine.pharmacy_id
WHERE medicine.medicine_name IN (SELECT all_medicine.medicine_name FROM all_medicine
WHERE all_medicine.medicine_description = "Tablets")
ORDER BY medicine.medicine_id DESC LIMIT 2;
```

Результат виконання:

| pharmacy_name | medicine_name | quantity |
|---|---|---|
| Phar7 | Ramtex | 7 |
| Phar4 | Mezym | 8 |

6. Визначимо покупців, які не зробили жодного замовлення.

```
SELECT customer.first_name FROM customer
WHERE NOT EXISTS (SELECT * FROM `order`
WHERE `order`.customer_id = customer.customer_id);
```

Результат виконання:

| first_name |
|---|
| Max |
| Olesya |
| Yuliana |
| Roman |
| Petro |
| Sasha |
| Yuriy |

7. Визначимо покупців, паролі яких не відповідають вимогам безпеки (менші за 5 символів).

```
SELECT login, password
FROM customer
WHERE CHAR_LENGTH(password)<5;
```

Результат виконання:

```
+-----------------------------+----------+
| login                       | password |
+-----------------------------+----------+
| vlad.kondratskiyv@gmail.com | vlad     |
| max.kachmaryk@gmail.com     | max      |
+-----------------------------+----------+
```

**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.