МІНІСТЕРСТВО ОСВІТИ І НАУКИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра систем штучного інтелекту

Звіт до лабораторної роботи №7

з дисципліни
“Організація Баз Даних та знань”

Виконала:
ст. гр. КН-210
Заремба Вікторія

Викладач:
Мельникова Н.І.

Львів – 2020

Лабораторна робота №7

з курсу “ОБДЗ”

на тему:

“Запити на вибір даних з таблиць бази даних”

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 директиви:

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

Для вивчення роботи директив вибору даних з таблиць розробимо та виконаємо такі запити над таблицями User.

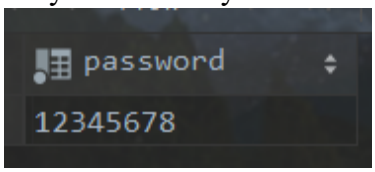
1. Показати пароль заданого користувача.
2. Показати користувачів і їхні завдання (ліве з'єднання таблиць).
3. Показати перелік користувачів у групі Guests (натуральне з'єднання).
4. Показати всі коментарі користувачів з груп Guests та SUgroup2 (умовне з'єднання).
5. Показати останні 3 коментарі користувачів з груп Guests та SUgroup2 (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).

1. **Показати пароль заданого користувача.**

Знайдемо пароль конкретного користувача. Для цього в умові відбору вкажемо номер потрібного користувача:

```
SELECT password
    FROM pm_system.user WHERE id_user = 1;
```

Результат запиту:



2. **Показати користувачів і їхні коментарі (ліве з'єднання таблиць).**
Виберемо всіх користувачів і їхні завдання. Таблиця user має зв'язок з user_department, user_department з таблицею department, а department з таблицею task. Виводити будемо лише значення з таблиць user і task, але з'єднувати усі 4. Тому запит виглядатиме так:

```
SELECT pm_system.user.id_user, pm_system.user.name, pm_system.user.surname,
pm_system.task.name, pm_system.task.deadline
    FROM pm_system.user LEFT JOIN pm_system.user_department ON
    user.id_user = pm_system.user_department.id_user
    LEFT JOIN pm_system.department ON
    pm_system.user_department.id_department = pm_system.department.id_department
    LEFT JOIN pm_system.task ON
    pm_system.department.id_department = pm_system.task.id_department;
```

Результат запиту:

	id_user	user.name	surname	task.name	deadline
1	1	Viktoria	Zaremba	Сконтактувати з шклами	<null>
2	7	Sophia	Osovska	Написати статтю	<null>
3	7	Sophia	Osovska	Сконтактувати з ЗМІ	<null>
4	7	Sophia	Osovska	Домовитись за рекламу	<null>
5	5	Bogdana	Kondratyk	Скласти програму	2018-12-05
6	3	Denys	Boyechko	Написати лист спонсорам	2018-12-02
7	6	Roman	Rogynski	Написати лист спонсорам	2018-12-02
8	8	Настя	Мацків	<null>	<null>
9	9	Оля	Корінь	Скласти програму табору	2020-04-15
10	9	Оля	Корінь	написати легенду	2020-03-25
11	9	Оля	Корінь	підготувати документи	2020-05-20
12	12	Віка	Зубик	Скласти програму табору	2020-04-15
13	12	Віка	Зубик	написати легенду	2020-03-25
14	12	Віка	Зубик	підготувати документи	2020-05-20
15	1	Viktoria	Zaremba	Закупити реманент	2020-06-25
16	10	Надійка	Осташевська	Написати меню	2020-05-11
17	11	Ярина	Драбик	Написати меню	2020-05-11
18	2	Anastasia	Vyshnevska	<null>	<null>

3. **Показати перелік департаментів кожного проекту (натуральне з'єднання).**
Виведемо список департаментів для кожного проекту. Для цього використаємо натуральне з'єднання. Відповідно, якщо проект немає департаментів (проект з ID = 3) його в таблиці результатів не буде.

```
SELECT pm_system.project.id_project, pm_system.project.name,
pm_system.department.name
    FROM pm_system.project INNER JOIN pm_system.department ON
    pm_system.project.id_project = pm_system.department.id_project;
```

Результат запиту:

	id_project	project.name	department.name
1	1	By St. Nicolas steps	Комунікації
2	1	By St. Nicolas steps	Програма
3	1	By St. Nicolas steps	Зв'язки
4	1	By St. Nicolas steps	Фандрейзинг
5	2	Курінний табір	Бунчужна
6	2	Курінний табір	Писарство
7	2	Курінний табір	Інтендантка
8	2	Курінний табір	Реманент

4. **Показати всі закріплені до департаментів 'Писарство', 'Інтендантка', 'Реманент' (умовне з'єднання).**

```
SELECT pm_system.department.name, pm_system.task.name
    FROM pm_system.department INNER JOIN pm_system.task
    ON department.id_department = task.id_department
    WHERE pm_system.department.name IN ('Писарство', 'Інтендантка', 'Реманент');
```

Результат запиту:

	department.name	task.name
1	Писарство	Скласти програму табору
2	Писарство	написати легенду
3	Писарство	підготувати документи
4	Інтендантка	Написати меню
5	Реманент	Закупити реманент

5. **Визначити користувачів, які не закріплені за жодним департаментом.**

Оскільки між користувачами і департаментами зв'язок багато-до-багатьох, в нас є проміжна таблиця user_department. Нею і скористуємось для визначення користувачів, які не закріплені за жодним департаментом.

```
SELECT pm_system.user.surname, pm_system.user.name FROM user
    WHERE NOT EXISTS
        (SELECT * FROM user_department WHERE user.id_user =
user_department.id_user);
```

Результат запиту:

	surname	name
1	Vyshnevska	Anastasia

6. **Визначити користувачів, паролі яких не відповідають вимогам безпеки (менші за 8 символів або не містять цифр).**

```
SELECT surname, name, password
    FROM user
    WHERE CHAR_LENGTH(password)<8 OR (password NOT REGEXP '[0-9]');
```

Результат запиту:

	surname	name	password
1	Rogynski	Roman	jocs,lm
2	Osovska	Sophia	uhdosj.k
3	Мацків	Настя	ujdscdosj.k
4	Корінь	Оля	udjsksj.k
5	Осташевська	Надійка	uhvdjkj.k
6	Драбик	Ярина	uhdosj.kjdj
7	Зубик	Віка	uhdokdjv.k

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.