

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"
ІНСТИТУТ ПІСЛЯДИПЛОМНОЇ ОСВІТИ
КАФЕДРА СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ



ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №7

Підготувала:

Студентка групи КН-209

Кульчицька Олена

Викладач:

Мельникова Н.І.

Лабораторна робота №7

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'єднаних 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 можна 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(<i>рядок1</i> , <i>рядок2</i>)	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 <i>рядок</i>	Порівняння з рядком-шаблоном. В шаблоні можна використовувати знаки % (довільні символи) і _ (довільний символ).
REGEXP <i>рядок</i>	Порівняння з рядком з використанням регулярних виразів. Функція-синонім – RLIKE.
MATCH (<i>поля</i>) AGAINST (<i>рядок</i>)	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(<i>arg1</i> , <i>arg2</i> , ...)	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 співпадиння не знайдено, а один із аргументів є NULL.
NOT IN(<i>arg1</i> , <i>arg2</i> , ...)	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(<i>arg1</i> , <i>arg2</i> , ...)	Повертає мінімальне значення серед аргументів. Повертає NULL, якщо хоча б один із аргументів є NULL.
GREATEST(<i>arg1</i> , <i>arg2</i> , ...)	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 Customer, Order, Staff:

- Показати ім'я та email заданого користувача.
 - Показати імена відвідувачів в порядку спадання.
 - Показати відвідувачів, номери яких починаються з цифри 8.
 - Показати відвідувачів та дати, час їхніх замовлень(ліве з'єднання).
 - Показати відвідувачів , які зробили замовлення о 14:15:00(натуральне з'єднання).
 - Показати відвідувачів та персонал,які обслуговували замовлення о 14:15:00 або 14:00:20(умовне з'єднання).
 - Показати 4 замовлення відвідувачів, обслужених персоналом під індексом 2 або 3(підзапит).
 - Визначимо відвідувачів, які не зробили жодного замовлення.
 - Визначимо відвідувачів, номери яких не відповідають вимогам.
1. Визначимо ім'я та email відвідувача, айді якого = 8. Для цього виконуємо селекцію.

```
SELECT id,first_name,email  
FROM confectionary.customer WHERE id = 8;
```

Результат запиту:

id	first_name	email
8	Jan	robrun@qmail.com
NULL	NULL	NULL

2. Для того,щоб вивести імені відвідувачів в порядку спадання, використовуємо команду ORDER BY.

```
SELECT id,first_name  
FROM confectionary.customer  
ORDER BY first_name DESC
```

Результат запиту:

	id	first_name
	5	Rob
	4	Olivia
	1	Olena
	7	Mark
	2	Lina
	3	Lina
	6	Lara
	8	Jan
	NULL	NULL

3. Для того, щоб вивести відвідувачів, номери телефонів яких починаються з цифри 8 використовуємо LIKE.

```
SELECT id,first_name,telephone
FROM confectionary.customer
WHERE telephone LIKE '8%'
```

Результат запиту:

	id	first_name	telephone
	1	Olena	8564735
	8	Jan	8887894
	NULL	NULL	NULL

4. Виберемо всіх користувачів з датою та часом їхніх замовлень. Для цього потрібно виконати ліве з'єднання. Для відвідувачів, які не зробили жодного замовлення в результатах буде відображено порожні значення.

```
SELECT customer.id, customer.first_name, order.date, order.time
FROM confectionary.customer LEFT JOIN confectionary.order ON
customer.id = order.customer_id;
```

Результат запиту:

	id	first_name	date	time
	1	Olena	2020-03-20	14:15:00
	1	Olena	2020-03-23	14:15:00
	2	Lina	2020-03-21	14:15:00
	3	Lina	2020-03-14	18:30:00
	4	Olivia	NULL	NULL
	5	Rob	2020-03-07	10:45:20
	6	Lara	NULL	NULL
	7	Mark	NULL	NULL
	8	Jan	2020-02-25	14:00:20

5. Виберемо відвідувачів, які зробили замовлення о 14:15:00. Для цього виконаємо умовне з'єднання таблиць Customer і Order за атрибутом *customer_id*, використовуючи директиву INNER JOIN.

```
SELECT customer.first_name, order.time
FROM confectionary.customer INNER JOIN confectionary.order ON
customer.id = order.customer_id
WHERE order.time='14:15:00';
```

Результат запиту:

	first_name	time
	Olena	14:15:00
	Lina	14:15:00
	Olena	14:15:00

6. Виберемо відвідувачів та персонал, які обслуговували замовлення о 14:15:00 або 14:00:20. Для цього виконаємо умовне з'єднання таблиць Customer і Order за атрибутом *customer_id*, та таблиці Staff використовуючи директиву INNER JOIN.

```
SELECT customer.first_name AS cusName, order.time, order.date, staff.first_name AS staffName
FROM (confectionary.customer INNER JOIN confectionary.order) INNER JOIN confectionary.staff
ON customer.id = order.customer_id
AND staff.id = order.staff_id
WHERE order.time IN ('14:15:00', '14:00:20');
```

Результат запити:

	cusName	time	date	staffName
	Olena	14:15:00	2020-03-20	Alex
	Lina	14:15:00	2020-03-21	Alex
	Olena	14:15:00	2020-03-23	Alex
	Jan	14:00:20	2020-02-25	Alex

7. Виберемо 4 замовлення відвідувачів, обслугованих персоналом під індексом 2 або 3. Для цього замість директиви JOIN використаємо підзапит в умові відбору.

```
SELECT customer.first_name, order.time, order.date
FROM confectionary.order INNER JOIN confectionary.customer
ON order.customer_id = customer.id
WHERE order.staff_id IN (SELECT staff.id FROM confectionary.staff
WHERE staff.id IN ('2', '3'))
ORDER BY order.date DESC LIMIT 4;
```

Результат запити:

	first_name	time	date
	Olena	14:15:00	2020-03-23
	Lina	14:15:00	2020-03-21
	Olena	14:15:00	2020-03-20
	Lina	18:30:00	2020-03-14

8. Визначимо відвідувачів, які не зробили жодного замовлення.

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

Результат запити:

	first_name
	Olia
	Lara
	Mark

9. Визначимо відвідувачів, номери яких не відповідають вимогам(менші за 7 символів або містять букви).

```
INSERT INTO confectionary.customer
VALUES (9, 'Lily', 'Raf', 'abc12', 'lily@gmail.com');
SELECT id, first_name, second_name, telephone
FROM confectionary.customer
WHERE CHAR_LENGTH (telephone) < 7 OR
(telephone) REGEXP '[a-z]';
```

Результат запити:

	id	first_name	second_name	telephone
	9	Lilv	Raf	abc12
	NULL	NULL	NULL	NULL

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.