Лабораторна робота № 2

Тема: Фільтрування даних.

Мета: Здобути навички з фільтрування даних в SQL запитах.

Використання WHERE

У таблицях баз даних зазвичай міститься дуже багато інформації, і досить рідко виникає необхідність витягувати всі рядки з таблиці. Набагато частіше потрібно отримати якусь частину даних для подальшої обробки або складання звітів. У цьому випадку необхідно вказати критерій відбору.

В інструкції SELECT дані фільтруються шляхом вказівки критерію відбору є інструкція WHERE. Ця інструкція вказується відразу після назви таблиці (FROM) в такий спосіб.

Запит

```
SELECT name, price
FROM products
WHERE price = 3.49;
```

Аналіз

Наведений запит витягує два стовпці з таблиці товарів, але повертає не всі рядки, а лише ті значення в стовпці price яких дорівнює 3.49

Результат

name	price
Fish bean bag toy	3.49
Bird bean bag toy	3.49
Rabbit bean bag toy	3.49

У цьому прикладі використовується проста перевірка на рівність: спочатку перевіряється, чи міститься в стовпці вказане значення, а потім фільтруються дані відповідним чином.

Однак SQL дозволяє виконувати не лише перевірку на рівність.

Оператор	**Перевірка**
=	Рівність
<>	Нерівність
!=	Нерівність
<	Менше
<=	Менше або дорівнює
!<	Не менше
>	Більше
>=	Більше або дорівнює
!>	Не більше
BETWEEN	Входження в діапазон
IS NULL	Значення NULL

ПОПЕРЕДЖЕННЯ: **сумісність операторів**

Деякі оператори, наведені в таблиці надмірні.
Наприклад, <> - це те ж саме, що і !=. Застосування оператора !< (не менше ніж) дає той самий результат, що і >= (більше або одно).
Майте на увазі: не всі з цих операторів підтримуються всіма СУБД. Зверніться до документації СУБД, щоб дізнатись, які логічні оператори вона підтримує.

Ми вже розглянули приклад перевірки на рівність. Тепер ознайомимося з іншими операторами.

У прикладі виводяться назви товарів, вартість яких не перевищує 10 доларів.

Запит

```
SELECT name, price
FROM products
WHERE price < 10;
```

Результат

name	price
Fish bean bag toy	3.49
Bird bean bag toy	3.49
Rabbit bean bag toy	3.49
8 inch teddy bear	5.99
12 inch teddy bear	8.99

Raggedy Ann	4.99
King doll	9.49
Queen doll	9.49

Комбінування умов WHERE

Всі пропозиції WHERE, подані вище, фільтрують дані з використанням одного критерію. Для створення більш складних фільтрів можна використовувати декілька умов WHERE.

Для цього призначені оператори AND та OR.

Оператор AND

Щоб відфільтрувати дані по кількох стовпцях, необхідно скористатися оператором AND для додавання умов WHERE. Ось як це робиться.

Запит

```
SELECT id, name, price
FROM products
WHERE vendor = 'DLL01' AND price <= 4;
```

Аналіз

За допомогою цього запиту вилучаються назви та ціни всіх товарів, що пропонуються постачальником DLL01, з ціною 4 долари і менше. Пропозиція WHERE в інструкції SELECT містить дві умови, а оператор AND використовується для їхнього об'єднання. Оператор AND змушує СУБД повертати лише ті рядки, які задовольняють перераховані умови.

Якщо товар пропонується постачальником DLL01, але коштує більше 4 доларів, він не потрапить до результатів запиту. Аналогічно, товари, які коштують менше 4 доларів і пропонуються іншими постачальниками, також не будуть виведені. Результат запиту будуть таким.

Результат

id	name	products
BNBG01	Fish bean bag toy	3.49
BNBG02	Bird bean bag toy	3.49
BNBG03	Rabbit bean bag toy	3.49

AND

Ключове слово, що використовується в пропозиції WHERE для того, щоб повертались лише ті рядки, які задовольняють всім зазначеним умовам.

У цьому прикладі використовувалися дві умови фільтрації, об'єднані оператором AND. Можна ставити і більше умов. Кожне з них має відокремлюватись оператором AND.

Оператор OR

Дія оператора OR протилежна дії оператора AND. Він змушує СУБД витягувати лише ті рядки, які задовольняють хоча б одній умові. У більшості СУБД друга умова навіть не розглядається у пропозиції WHERE, якщо виконується перша умова (у такому разі рядок буде виведений незалежно від другої умови).

Запит

```
SELECT name, price
FROM products
WHERE vendor = 'DLL01' OR vendor = 'BRS01';
```

Аналіз

За допомогою цього запиту вилучаються назви та ціни всіх товарів, виготовлених одним із зазначених постачальників. Оператор OR змушує СУБД застосовувати якусь одну умову, а не обидві відразу. Якби тут використовувався оператор AND, ми не отримали жодних даних. Результат запиту будуть такими.

.

Результат

name	products
Fish bean bag toy	3.49
Bird bean bag toy	3.49
Rabbit bean bag toy	3.49
8 inch teddy bear	5.99

12 inch teddy bear	8.99
Raggedy Ann	4.99
18 inch teddy bear	11.99

OR

Ключове слово, що застосовується з WHERE для того, щоб поверталися всі рядки, що задовольняють будь-кому з зазначених умов.

Порядок обробки операторів

Пропозиції WHERE можуть містити будь-яку кількість логічних операторів AND та OR. Комбінуючи їх, можна створювати складні фільтри. Однак при комбінуванні операторів AND та OR виникає одна проблема. Розглянемо приклад. Необхідно вивести список всіх товарів, що пропонуються постачальниками DLL01 і BRS01, ціна яких - 10 доларів і вище. У наступній інструкції SELECT використовується комбінація операторів AND та OR для формування умови відбору рядків.

Запит

```
SELECT name, price
FROM products
WHERE vendor = 'DLL01' OR vendor = 'BRS01' AND price >= 10;
```

Результат

name	price
Fish bean bag toy	3.49
Bird bean bag toy	3.49
Rabbit bean bag toy	3.49
Raggedy Ann	4.99
18 inch teddy bear	11.99

Аналіз

У чотирьох рядках, що повертаються, значаться ціни нижче 10 доларів - очевидно, рядки не були відфільтровані так, як треба. Що сталося? Причина полягає у порядку обробки операторів. SQL (як і більшість інших мов) спочатку обробляє логічні оператори AND, а потім - логічні оператори OR. Коли СУБД зустрічає таку пропозицію WHERE, вона інтерпретує його так: витягти всі товари, які коштують 10 доларів і більше, а при цьому пропонуються постачальником BRS01, а також всі товари, пропоновані постачальником DLL01, незалежно від їхньої ціни.

Іншими словами, у зв'язку з тим, що пріоритет у логічного оператора AND вищий, були об'єднані не ті оператори.

Вирішення цієї проблеми полягає у використанні дужок для точного угруповання необхідних логічних операторів. Розглянемо наступну інструкцію SELECT та її результати.

Запит

```
SELECT name, price
FROM products
WHERE (vendor = 'DLL01' OR vendor = 'BRS01') AND price >= 10;
```

Результат

name	products
18 inch teddy bear	11.99

Аналіз

Єдиною відмінністю між попереднім запитом та цим є дужки, в які укладені перші дві умови пропозиції WHERE. Оскільки дужки мають ще більший пріоритет, ніж логічні оператори AND та OR, СУБД спочатку обробляє умова OR у дужках. Відповідно, запит розумітиметься так: витягти всі товари, пропоновані або постачальником DLL01, або постачальником BRS01, які коштують 10 доларів та більше. Це саме те, що потрібно

Оператор IN

Оператор IN служить для вказівки діапазону умов, будь-яка з яких може бути виконано. При цьому значення, ув'язнені у дужки, перераховуються через кому. Розглянемо наступний приклад.

Запит

```
SELECT name, price
FROM products
WHERE vendor IN ( 'DLL01', 'BRS01')
ORDER BY name;
```

Результат

name	products
12 inch teddy bear	8.99
18 inch teddy bear	11.99
8 inch teddy bear	5.99
Bird bean bag toy	3.49
Fish bean bag toy	3.49
Rabbit bean bag toy	3.49
12 inch teddy bear	8.99
Raggedy Ann	4.99

Аналіз

Інструкція SELECT витягує всі товари, пропоновані постачальниками DLL01 та BRS01. Після оператора IN вказано список значень через кому, а весь список укладено у дужки. Якщо ви думаєте, що оператор IN виконує ту ж функцію, щоі OR, то будете абсолютно праві.

Навіщо потрібен оператор IN? Його переваги такі:

- Під час роботи з довгими списками необхідних значень синтаксис логічного оператора IN набагато зрозуміліший.
- Під час використання оператора IN разом із операторами AND та OR набагато легше керувати порядком обробки.
- Оператори IN майже завжди швидше обробляються, ніж списки операторів OR (втім, це складно помітити у разі коротких списків).
- Найбільша перевага логічного оператора IN полягає в тому, що в ньому може бути ще одна інструкція SELECT, а це дозволяє створювати дуже гнучкі пропозиції WHERE. Докладніше про це буде у наступних роботах.

Оператор NOT

Логічний оператор NOT у пропозиції WHERE служить тільки однієї мети - заперечувати всі умови, що йдуть за ним. Оскільки він ніколи не використовується сам по собі (а лише разом з іншими логічними операторами), його синтаксис трохи відрізняється. Оператор NOT вставляється перед назвою стовпця, значення якого потрібно відфільтрувати, а не після.

NOT
Ключове слово, застосовуване у реченні WHERE для заперечення будь-якої умови.

У цьому прикладі демонструється застосування логічного оператора NOT. Тут витягується список товарів, що пропонуються всіма постачальниками, за винятком DLL01.

Запит

```
SELECT name
FROM products
WHERE NOT vendor = ' DLL01'
ORDER BY name;
```

Результат

name
12 inch teddy bear
18 inch teddy bear
8 inch teddy bear
King doll
Queen doll

Аналіз

Навіщо використовувати логічний оператор NOT? Звичайно, для таких простих пропозицій WHERE, які тут розглянуті, він не обов'язковий. Його переваги виявляються у більш складніших умовах. Наприклад, для знаходження всіх рядків, які не співпадають зі списком критеріїв, можна використовувати логічний оператор NOT у зв'язці з оператором IN.

Резюме

В рамках цієї лабораторної работи ви дізналися, як використовувати логічні оператори AND та OR у запитах WHERE. Було також показано, як керувати порядком обробки операторів та як застосовувати оператори IN та NOT.

ЗАВДАННЯ:

Для виконання роботи необхідно імпортувати таблицю cities.

Необхідно написати наступні запити:

1. Отримати список міст-мільйонерів України
2. Отримати список міст з регіону E та W відсортований за кількістю населення
3. Отримати список міст з населенням більше ніж 50000 з регіонів S, C та N (вікористовуючи оператор IN)
4. Отримати перші 20 міст (за назвою) у яких населення більше ніж 150000 але не більше ніж 350000 та які знаходяться в регіонах E,W,N
5. Отримати другу десятку міст за кількістю населення які не входять до регіонів E та W

Усі запити мають бути збережені у файл в форматі .sql

Таблиця cities

```
SET NAMES utf8;
SET time_zone = '+00:00';
SET foreign_key_checks = 0;
SET sql_mode = 'NO_AUTO_VALUE_ON_ZERO';

DROP TABLE IF EXISTS `cities`;
CREATE TABLE `cities` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `population` int(10) unsigned DEFAULT NULL,
  `region` varchar(5) COLLATE utf8_unicode_ci DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

INSERT INTO `cities` (`id`, `name`, `population`, `region`) VALUES
(1,     'Київ',2888470,    'N'),
(2,     'Харків',      1444540,    'E'),
(3,     'Одеса',       1010000,    'S'),
(4,     'Дніпро',      984423,     'C'),
(5,     'Донецьк',  932562,     'E'),
(6,     'Запоріжжя',       758011,     'E'),
(7,     'Львів',       728545,     'W'),
(8,     'Кривий Ріг',      646748,     'S'),
(9,     'Миколаїв',494381,     'S'),
(10,    'Маріуполь',       458533,     'S'),
(11,    'Луганськ', 417990,     'E'),
```

(12, 'Севастополь', 412630, 'S'),
(13, 'Вінниця', 372432, 'W'),
(14, 'Макіївка', 348173, 'E'),
(15, 'Сімферополь', 332608, 'S'),
(16, 'Херсон', 296161, 'S'),
(17, 'Полтава', 294695, 'E'),
(18, 'Чернігів', 294522, 'N'),
(19, 'Черкаси', 284459, 'C'),
(20, 'Суми', 268409, 'E'),
(21, 'Житомир', 268000, 'N'),
(22, 'Хмельницький', 267891, 'W'),
(23, 'Чернівці', 264427, 'W'),
(24, 'Горлівка', 250991, 'E'),
(25, 'Рівне', 249477, 'W'),
(26, 'Кам\'янське', 240477, 'C'),
(27, 'Кропивницький', 232052, 'C'),
(28, 'Івано-Франківськ', 229447, 'W'),
(29, 'Кременчук', 224997, 'C'),
(30, 'Тернопіль', 217950, 'W'),
(31, 'Луцьк', 217082, 'W'),
(32, 'Біла Церква', 211080, 'N'),
(33, 'Краматорськ', 160895, 'E'),
(34, 'Мелітополь', 156719, 'S'),
(35, 'Керч', 147668, 'S'),
(36, 'Сєвєродонецьк', 130000, 'E'),
(37, 'Хрустальний', 124000, 'E'),
(38, 'Нікополь', 119627, 'C'),
(39, 'Бердянськ', 115476, 'S'),

```
(40, 'Слов\'янськ', 115421, 'E'),
(41, 'Ужгород', 115195, 'W'),
(42, 'Алчевськ', 111360, 'E'),
(43, 'Павлоград', 110144, 'E'),
(44, 'Євпаторія', 106115, 'S'),
(45, 'Лисичанськ', 103459, 'E'),
(46, 'Кам\'янець-Подільський', 101590, 'W'),
(47, 'Бровари', 100374, 'N'),
(48, 'Дрогобич', 98015, 'W'),
(49, 'Кадіївка', 92132, 'E'),
(50, 'Конотоп', 92000, 'E');
```