Лабораторна робота № 3

Тема: Фільтрування з використанням метасимволів.

Обчислювані поля.

Мета: Здобути навички з фільтрування строкових даних в SQL запитах. Здобути навички роботи з обчислюваними полями в SQL.

Використання оператора LIKE

Усі попередні оператори, які ми розглядали, виконували фільтрацію за відомими значеннями. Вони шукали збіги за одним або декількома значеннями, здійснювали перевірку “більше-менше” або перевірку на входження в діапазон значень. При цьому скрізь знаходилось відоме значення.

Однак фільтрація даних таким способом не завжди працює. Наприклад, як би ви шукали товари, у назві яких містяться слова “bean bag”? Цього не можна зробити в простих операціях порівняння, і тут на допомогу приходить пошук з використанням метасимволів.

Завдяки метасимволам можна створювати розширені умови відбору рядків. У цьому прикладі, щоб знайти всі товари, в назві яких містяться слова bean bag, необхідно скласти шаблон пошуку, що дозволяє знайти текст bean bag в будь-якому місці назви товару.

Метасимволи

Спеціальні символи для пошуку частини значення.

Шаблон пошуку

Умова відбору рядків, що складається з тексту, метасимволів та будь-якої їх комбінації.

Метасимволи самі по собі є спеціальними знаками, які особливим чином трактуються за умови WHERE. У SQL підтримуються метасимволи кількох типів. Щоб використовувати метасимволи в умовах відбору рядків, необхідно використовувати ключове слово LIKE. Воно повідомляє СУБД у тому, що наступний шаблон пошуку необхідно аналізувати з урахуванням метасимволів, а чи не шукати точні збіги.

Пошук з використанням метасимволів може здійснюватися лише у текстових полях (рядках). Метасимволи не можна застосовувати при пошуку нетекстових полів.

Метасимвол "знак відсотка" (%)

Найчастіше використовуваний метасимвол - знак відсотка (%).
У шаблоні пошуку знак % означає знайти всі входження будь-якого символу. Наприклад, щоб знайти всі товари, назви яких починаються зі слова Fish, можна виконати наступний запит.

Запит

```
SELECT id, name
FROM products
WHERE name LIKE 'Fish%';
```

Результат

id	name
BNBG01	Fish bean bag toy

Аналіз

У цьому прикладі використовується шаблон пошуку 'Fish%'. При перевірці цієї умови повертаються всі значення, які починаються із символів Fish. Знак % змушує СУБД приймати всі символи після слова Fish незалежно від кількості.

ВАЖЛИВО: метасимволи в Microsoft Access

Під час роботи з Microsoft Access необхідно використовувати символ * замість %.

ВАЖЛИВО: чутливість до регістру символів

Залежно від СУБД та її конфігурації операції пошуку можуть бути чутливими до регістру символів. В такому випадку значення Fish bean bag toy не буде відповідати умові 'fish%';

Метасимволи можуть зустрічатися в будь-якому місці шаблону пошуку, причому у необмеженій кількості. У наступному прикладі використовуються два метасимволи, по одному на обох кінцях шаблону.

Запит

```
SELECT id, name
FROM products
WHERE name LIKE '%bean bag%';
```

Результат

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

Аналіз

Шаблон ' %bean bag% ’ означает найти все значения, содержащие текст bean bag в любом месте названия, независимо от количества символов перед указанным текстом или после него.

Метасимвол может находиться в шаблоне поиска, хотя это редко бывает полезным. В следующем примере выполняется поиск всех товаров, названия которых начинаются на F и заканчиваются на y.

Запит

```
SELECT id, name
FROM products
WHERE name LIKE 'F%y';
```

Метасимвол "знак підкреслення" (_)

Ще одним корисним метасимвол є знак підкреслення (_). Він використовується так само, як знак %, але при цьому враховується не безліч символів, а тільки один.

Запит

```
SELECT id, name
FROM products
WHERE name LIKE '__ inch teddy bear';
```

Результат

id	name
BNBG02	12 inch teddy bear
BNBG03	18 inch teddy bear

Аналіз

У шаблоні пошуку цієї пропозиції WHERE використано два метасимволу, після яких слідує текст. У результаті було відібрано лише ті рядки, які задовольняють умові: за двома символами підкреслення було знайдено число 12 у першому рядку та 18 – у другому. Товар 8 inch teddy bear не був відібраний, так як у шаблоні пошуку потрібно два збіги, а не один. Для порівняння: у наступній інструкції SELECT використовується метасимвол %, внаслідок чого витягуються три товари.

Запит

```
SELECT id, name
FROM products
WHERE name LIKE '% inch teddy bear';
```

Результат

id	name
BNBG01	8 inch teddy bear
BNBG02	12 inch teddy bear
BNBG03	18 inch teddy bear

На відміну від знака %, який має на увазі також відсутність символів, знак _ завжди означає один символ - не більше і не менше

Метасимвол "квадратні дужки" ([ ])

Метасимвол "квадратні дужки" ([ ]) служить для вказівки набору символів, кожен з яких повинен збігатися з потрібним значенням, причому в точно вказаному місці (у позиції метасимволу).

ВАЖЛИВО: набори не завжди підтримуються

На відміну від описаних раніше метасимволів, використання квадратних дужок для створення перевірочних наборів не підтримується багатьма СУБД. Набори підтримуються в Microsoft Access та Microsoft SQL Server. Зверніться до документації СУБД, щоб визначити, чи підтримуються в ній набори.

Наприклад, щоб знайти всіх клієнтів, імена яких починаються на букву J або M, необхідно виконати наступний запит.

Запит

```
SELECT contact
FROM customers
WHERE contact LIKE  '[JM]%'
ORDER BY contact;
```

Результат

contact
Jim Jones
John Smith
Michelle Green

Аналіз

Умова WHERE у цій інструкції має такий вигляд: '[JM]%'.

У цьому шаблоні пошуку використовуються два різних метасимволи. За виразом [JM] здійснюється пошук всіх клієнтів, імена яких починаються на одну із вказаних у дужках букв, але при цьому враховується лише один символ. Тому всі імена довші за один символ не будуть вилучені. Завдяки метасимволу %, наступному після [JM], виконується пошук будь-якої кількості символів після першої літери, що призводить до потрібного результату.

Поради щодо використання метасимволів

Метасимволи в SQL – дуже потужний механізм. Але за цю міць доводиться платити: на пошук із використанням метасимволів йде більше часу, ніж на будь-які інші види запитів, які розглядалися раніше. Нижче наведено декілька порад щодо використання метасимволів.

- Не зловживайте метасимволами. Якщо можна використати інший оператор пошуку, слід задіяти його.
- У разі використання метасимволів намагайтеся по можливості не вставляти їх на початок шаблону пошуку. Шаблони, починаючи що з метасимволів, обробляються найповільніше.
- Уважно стежте за позицією метасимволів. Якщо вони знаходяться не на своєму місці, буде вилучено не ті дані.

Проте слід сказати, що метасимволи дуже важливі і корисні при пошуку, ви часто будете ними користуватися.

Створення обчислювальних полів

Дані, що зберігаються в таблицях бази даних, зазвичай бувають представлені не такому вигляді, який необхідний у додатках.

Ось кілька прикладів:

- Вам необхідно відобразити поле, що містить назву компанії та її адресу, але ця інформація знаходиться в різних стовпці таблиці.
- Місто, штат та поштовий індекс зберігаються в окремих стовпцях (як і має бути), але для програми друку поштових наліпок ця інформація потрібна в одному коректно відформатованому полі.
- Дані в стовпці введені великими та малими буквами, але у звіті необхідно використовувати лише великі.
- У таблиці з елементами замовлення зберігається ціна кожного товару та його кількість, але не повна вартість (ціна одного товару, помножена на його кількість). Щоб роздрукувати інвойс, необхідно обчислити повні ціни.
- Вам потрібно знати загальну суму, середнє значення чи інші підсумкові показники, засновані на даних, що зберігаються в таблиці.

У кожному з цих прикладів дані зберігаються не в тому вигляді, якому їх необхідно надати додатку. Замість того, щоб витягувати дані, а потім переформатувати їх у клієнтському додатку або звіт, краще витягувати вже перетворені, підраховані або відформатовані дані прямо з бази даних.

Саме тут на допомогу приходять поля, що обчислюються. Взагалі-то, в таблицях бази даних ніяких обчислювальних стовпців немає. Вони створюються на льоту інструкцією SELECT.

Важливо відзначити, що тільки база даних “знає”, які стовпці в інструкції SELECT є реальними стовпцями таблиці, а які полями, що обчислюються. З погляду клієнта (наприклад, користувача програми) дані обчислюваного поля повертаються так само, як і дані з будь-якого іншого стовпця.

Конкатенація полів

Щоб продемонструвати застосування обчислюваних полів, розглянемо простий приклад: створення заголовка, що складається із двох стовпців.

У таблиці vendors зберігається назва постачальника та його країни. Припустимо, необхідно створити звіт по постачальнику і вказати його країну як частину його імені у вигляді ім (страна) .

У звіті має бути одне значення, а дані в таблиці зберігаються у двох стовпцях: name і country. Крім того, значення vend_country необхідно укласти в дужки, яких немає в таблиці бази даних. Інструкція SELECT, яка повертає імена постачальників та адреси, досить проста, але як створити комбіноване значення?

Конкатенація Комбінування значень (шляхом приєднання їх один до одного) для отримання одного "довгого" значення.

Для цього потрібно з'єднати два значення. В інструкції

SELECT можна виконати конкатенацію двох столпців за допомогою спеціального оператора. Залежно від СУБД може бути знак “плюс” (+) чи дві вертикальні рисочки (| |). У випадку MySQL та MariaDB доведеться використовувати спеціальну функцію.

ВАЖЛИВО. оператор + чи || ?

У Access та SQL Server для конкатенації використовується оператор **+**.
У DB2, Oracle, PostgreSQL, SQLite та OpenOffice Base під тримається оператор | |. За більш детальною інформацією зверніться до документації СУБД.

Приклад використання оператора + (синтаксис, прийнятий у більшості СУБД).

Запит

```
SELECT name + ' (' + country + ')'
FROM vendors
ORDER BY name;
```

Результат

Bear Emporium (USA)
Bears R Us (USA)
Doll House Inc. (USA)
Fun and Games (England)
Furball Inc. (USA)
Jouets et ours (France)

Еквівалентна інструкція для MySQL та MariaDB.

Запит

```
SELECT Concat(name, ' (', country, ')')
FROM vendors
ORDER BY name;
```

Аналіз

У попередніх інструкціях SELECT було виконано конкатенація наступних елементів:

► ім'я, що зберігається в стовпці name;

► рядок, що містить пробіл і круглу дужку, що відкриває;

► назва країни, що зберігається в стовпці country;

► рядок, що містить круглу дужку, що закриває.

Як видно з результатів запиту, інструкція SELECT повертає один стовпець (обчислюване поле), що містить всі чотири елементи як одне ціле.

Виконання математичних обчислень

Ще одним способом використання обчислювальних полів є виконання математичних операцій над отриманими даними. Розглянемо приклад. У таблиці orders зберігаються всі отримані замовлення, а таблиці order_items — списки товарів для кожного замовлення. Наступний запит отримує всі товари, що стосуються замовлення з номером 20008.

Запит

```
SELECT id, quantity, price
FROM order_items
WHERE number = 20008;
```

Результат

id	quantity	price
RGAN01	5	4.99
BR03	5	11.99
BNBG01	10	3.49
BNBG02	10	3.49
BNBG03	10	3.49

У стовпці price міститься ціна за одиницю товару, включеного на замовлення. Щоб дізнатись повну вартість (ціна за одиницю, помножена на кількість товарів у замовленні), необхідно модифікувати запит наступним чином.

Запит

```
SELECT      id, quantity, price, quantity*price AS expanded_price
FROM order_items
WHERE number = 20008;
```

Результат

id	quantity	price	expanded_price
RGAN01	5	4.99	24.95
BR03	5	11.99	59.95
BNBG01	10	3.49	34.90
BNBG02	10	3.49	34.90
BNBG03	10	3.49	34.90

Аналіз

Стовпець expanded_price в даному випадку є полем, що обчислюється. Обчислення було простим: quantity * price.

Тепер клієнтська програма може використовувати новий обчислювальний стовпець, як і будь-який інший у таблиці

У SQL підтримуються основні математичні оператори, перелічені у таблиці:

Оператор	**Перевірка**
+	Додавання
-	Віднімання
*	Множення
/	Ділення.

ЗАВДАННЯ:

Для виконання роботи необхідно імпортувати таблицю cities.

Необхідно написати наступні запити:

1. Отримати список міст України які закінчуються на ‘ськ’
2. Отримати список міст України у назві яких є ‘донец’
3. Отримати список міст з населенням більше ніж 100 000 з назвою у вигляді *НАЗВА_МІСТА (РЕГІОН)*. Результат відсортувати за алфавітом
4. Отримати перші 50 міст України (за населенням) та додати до запити обчислюване поле в якому буде вказано який процент від усього населення країни проживає у цьому місті. За загальну кількість населення України беремо число - 40 000 000.
5. Отримати список міст України у яких населення більше чи дорівнює 0.1% від загального населення України. Список має бути у форматі *НАЗВА_МІСТА - ПРОЦЕНТ_НАСЕЛЕННЯ %.* Результат відсортувати за процентом населення.

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');
```