Лабораторна робота № 4

Тема: Функцій обробки даних.

Підсумкові обчислення.

Мета: Здобути навички з функціями обробки даних та з функціями підсумкових обчислень.

Функція

Як і в більшості інших мов програмування, SQL підтримується використання функцій для роботи з даними.

Функції - це операції, які найчастіше доводиться виконувати над даними, включаючи різні перетворення та обчислення. Прикладом може бути функція RTRIM(), яку можно використовувати для видалення прогалин в кінці рядка.

Проблеми з функціями

Перш ніж переходити до прикладів, треба зауважити, що використання SQL-функцій може бути проблематичним.

На відміну від інструкцій SQL (наприклад, SELECT), які в здебільшого підтримуються всіма СУБД однаково, в різних СУБД можуть застосовуватися різні функції для тих самих цілей. Лише деякі функції у різних СУБД називаються однаково. Загальна функціональність доступна в кожній СУБД, але назви функцій та його синтаксис можуть істотно відрізнятися.

Задача	Синтаксис
Вибірка частини рядка	У Access використовується функція MID(), в DB2, Oracle, PostgreSQL і SQLite - SUBSTR(), в MariaDB, MySQL і SQL Server - SUBSTRING()
Перетворення типу даних	У Access і Oracle використовують кілька функцій, по одній на кожен тип перетворення. В DB2 и PostgreSQL використовується функція CAST(), в MariaDB, MySQL и SQL Server - CONVERT()
Отримання поточної дати	У Access використовується функція NOW(), в DB2 і PostgreSQL - CURRENT_DATE ,в MariaDB і MySQL - CURDATE(), Oracle - SYSDATE, в SQL Server - GETDATE(), в SQLite - DATE()

Застосування функцій

У більшості реалізацій SQL підтримуються такі типи функцій.

- Текстові функції. Використовуються для обробки текстових рядків (наприклад, для відсікання пробілів, або заповнення рядків пробілами, або перетворення символів у верхній або нижній регістр).
- Числові функції. Використовуються для виконання математичних операцій над числовими даними (таких, як ня в ступінь, вилучення кореня і т.п.).

- Дата та час. Використовуються для обробки значень дати і часу, а також для отримання окремих компонентів цих значень (наприклад, визначення різниці між датами та перевірки коректності дати).
- Системні функції. Повертають інформацію, специфічну для конкретної СУБД (наприклад, відомості про облікову запису користувача).

Функції можна застосовувати як у інших пропозицій інструкції SELECT (наприклад, за умови WHERE), так і в інших інструкціях SQL (про це ви дізнаєтесь нанаступних уроках).

Функції для роботи з текстом

На предыдущем уроке функция RTRIM() применялась для удаления пробелов в конце значения столбца. Ниже приведен другой пример, в котором используется функция UPPER()

Запит

```
SELECT name, UPPER(name) AS name_upcase
FROM vendors
ORDER BY name;
```

Результат

name	name_upcase
Bear Emporium	BEAR EMPORIUM
Bears R Us	BEARS R US
Doll House Inc.	DOLL HOUSE INC.
Fun and Games	FUN AND GAMES
Furball Inc.	FURBALL INC.
Jouets et ours	JOUETS ET OURS

Аналіз

Функція UPPER() перетворює символи у верхній регістр, тому в даному прикладі ім'я кожного постачальника перераховано двічі: перший раз у тому вигляді, в якому воно зберігається в таблиці vendors, а вдруге - перетвореним у верхній регістр, у вигляді стовпця name_upcase.

Найчастіше використовувані текстові функції

Функція	Опис
LEFT()	Повертає символи з лівої частини рядка
LENGTH (а также DATALENGTH() АБО LEN( ))	Повертає довжину рядка
LOWER() (LCASE () в Access)	Перетворює рядок на нижній регістр
LTRIM()	Видаляє прогалини в лівій частині рядка
RIGHT()	Повертає символи з правої частини

	рядка
SOUNDEX()	Повертає значення SOUNDEX рядка
UPPER() (UCASE() в Access)	Перетворює рядок у верхній регістр

SOUNDEX - це алгоритм, що перетворює текстовий рядок у буквено-цифровий шаблон, що описує фонетичне уявлення даного тексту. Функція SOUNDEX() бере до уваги схожі за звучанням літери та склади, дозволяючи порівнювати рядки не за тим, як вони пишуться, а за тим, як вони звучать. Незважаючи на те, що SOUNDEX() не відповідає основним концепціям SQL, більшість СУБД підтримує цю функцію.

Функції для роботи з датою та часом

Значення дати та часу зберігаються в таблицях з використанням відповідних типів даних, які у кожній СУБД свої. Завдяки наявності спеціальних форматів, ці значення можна швидко відсортувати або відфільтрувати, а крім того, вони займають досить мало місця на диску.

Формат, в якому зберігаються значення дати та часу, зазвичай

не можна використовувати в додатках, тому майже завжди доводиться застосовувати спеціальні функції для отримання цих значень та маніпулювання ними. У результаті такі функції є одними з найважливіших у SQL. На жаль, вони менш узгоджені і менш сумісні в різних реалізаціях SQL.

Щоб продемонструвати застосування цих функцій, розглянемо

Три простий приклад. У таблиці Orders всі замовлення зберігаються з датою замовлення. Щоб отримати список усіх замовлень, зроблених у 2012 році, у SQL Server необхідно виконати таку інструкцію.

Запит

```
SELECT num
FROM orders
WHERE DATEPART(yy, order_date) = 2012;
```

Результат

num
20005
20006
20007
20008
20009

Функції для роботи з числами

Числові функції призначені обробки числових даних. Вони застосовуються в основному для виконання алгебраїчних, тригонометричних та геометричних обчислень, тому потреба в них виникає не так часто, як у рядкових функціях або функціях роботи з датою та часом.

За іронією долі, у більшості СУБД саме числові функції є найбільш стандартизованими.

Функція	**Що повертається**
ABS()	Модуль числа
COS()	Косинус заданого кута
EXP()	Експонента заданого числа
PI()	Число п
SIN()	Синус заданого кута
SQRT()	Квадратний корінь заданого числа
TAN()	Тангенс заданого кута

Використання підсумкових функцій

Часто буває необхідно підбити підсумки, не відображаючи вихідні дані, і SQL для цього передбачені спеціальні функції. SQL-запити з такими функціями часто використовуються для аналізу даних та створення звітів. Прикладами таких запитів можуть бути:

- підрахунок числа рядків у таблиці (або числа рядків, які задовольняють певну умову або містять певне значення);
- визначення суми за набором рядків у таблиці;
- пошук найбільшого, найменшого та середнього значень у стовпці таблиці (по всіх чи якихось конкретних рядках).

У кожному з цих прикладів користувачеві потрібні підсумкові зведення таблиці, а не вихідні дані. Тому вилучення даних з таблиці було б марною витратою часу та ресурсів. Отже, все, що вам потрібно, – лише підсумкова інформація.

Щоб полегшити вилучення подібної інформації, в SQL передбачено набір із п'яти підсумкових функцій. Ці функції дозволяють виконувати всі варіанти запитів, які перелічені вище. На відміну від функцій обробки

даних, підсумкові функції підтримуються без особливих змін у більшості реалізацій SQL.

Функція	**Що повертається**
AVG()	Среднее значение по столбцу
COUNT()	Число строк в столбце
MAX()	Найбільше значення у стовпці
MIN()	Найменше значення у стовпці
SUM()	Сума значень стовпця

Функція COUNT()

Функція COUNT() підраховує кількість рядків. З її допомогою можна дізнатися загальну кількість рядків у таблиці або кількість рядків, які відповідають певному критерію.

Цю функцію можна використовувати двома способами:

- у вигляді виразу COUNT(*) для підрахунку числа рядків у таблиці незалежно від того, містять стовпці значення NULL чи ні;
- у вигляді виразу COUNT(стовпець) для підрахунку числа рядків, які мають значення у зазначених стовпцях, притому значення NULL ігноруються.

У першому прикладі повертається загальна кількість імен клієнтів, які у таблиці customers.

Запит

```
SELECT COUNT (*) AS num_cust
FROM customers;
```

Результат

num oust
5

Аналіз

У цьому прикладі функція COUNT(*) використовується для підрахунку всіх рядків незалежно від значень. Сума повертається у вигляді стовпця із псевдонімом num_cust.

Функція SUM()

Функція SUM() повертає суму значень у зазначеному стовпчику. Розглянемо приклад. У таблиці order_items містяться елементи замовлення, причому кожному елементу відповідає певна кількість, вказана у замовленні. Загальна кількість замовлених товарів (суму всіх значень стовпця quantity) можна визначити наступним чином.

Запит

```
SELECT SUM(quantity) AS item_ordered
FROM order_iterns
WHERE order_item = 20005;
```

Результат

item ordered
200

Комбінування підсумкових функцій

В усіх прикладах застосування підсумкових функцій, наведених досі використовувалася лише одна функція. Але насправді інструкція SELECT може містити стільки підсумкових функцій, скільки потрібно запиту. Розглянемо приклад.

Запит

```
SELECT    COUNT (*) AS num_items,
          MIN(price) AS price_min,
          MAX(price) AS price_max,
          AVG(price) AS proce_avg
FROM products;
```

Результат

num_items	price_min	price_max	proce_avg
9	3.4900	11.9900	6.823333

Аналіз

В даному випадку в одній інструкції SELECT використовуються одразу чотири підсумкові функції і повертаються чотири значення (кількість елементів у таблиці products, найвища, найнижча та середня вартість товарів).

Підсумкові функції призначені обчислення базових статистичних даних. В SQL підтримуються п'ять таких функцій, кожна з яких може використовуватися декількома способами для отримання тільки необхідних в даний момент результатів.

Ці функції досить ефективні і зазвичай повертають результат набагато швидше, ніж якби аналогічні обчислення виконувались у клієнтській програмі

ЗАВДАННЯ:

Для виконання роботи необхідно імпортувати таблицю cities.

Необхідно написати наступні запити:

1. Повернути другу п'ятірку міст України (за алфавітом) назву повернути у верхньому регістрі.
2. Отримати назву міста і в окремому стовпчику довжину назви. У результат не повинні потрапити міста з довжиною назва 8,9 та 10 символів.
3. Отримати кількість населення у регіоні C та S
4. Отримати середню кількість населення у містах з регіону W.
5. Отримати кількість міст у регіоні E.

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');
```