МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ
ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ БУДІВНИЦТВА, ТРАНСПОРТУ ТА ЕНЕРГЕТИКИ
КАФЕДРА АВТОМАТИЗАЦІЇ ВИРОБНИЧИХ ПРОЦЕСІВ

«СИСТЕМИ БАЗИ ДАНИХ І ЗНАНЬ»
Частина ІІ
(SQL)

М Е Т О Д И Ч Н І В К А З І В К И

для виконання практичних робіт студентами спеціальності
G7 «Автоматизація, комп'ютерно-інтегровані технології та робототехніка»

Затверджено на засіданні кафедри
Автоматизації виробничих процесів
Протокол №_1_ від_29.08.2025_р.

КРОПИВНИЦЬКИЙ
2025

Методичні вказівки до виконання практичних робіт з курсу “Системи бази даних і знань” (SQL) ч. II для студентів спеціальності G7 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка», 76 с.

Укладачі: М.О. Федотова, канд. техн. наук, асистент
Д.В. Трушаков, канд. техн. наук, доцент

Рецензент: О.К. Дідик, канд. техн. наук, доцент, завідувач каф АВП

ПРАКТИЧНА РОБОТА №1

Тема: MySQL – засіб створення баз даних
Мета: Отримати інформацію про переваги використання MySQL

MySQL є швидкою та стабільною системою. Це є основною причиною її популярності. В лютому 2009 року електронне видання eWeek (www.eweek.com) надрукувало результати тестування основних систем керування базами даних, включно Oracle, Microsoft SQL Server, DB2 та MySQL. Результати тестування вивели MySQL та Oracle 9 на перше місце з продуктивності. Слід відмітити, що тестування виконувалося для альфа-версії MySQL 4.0, але версія MySQL 4.1 працює ще швидше.

Система MySQL добре перевірена та надійна. Вона використовується цілим рядом дуже вимогливих замовників, таких як Yahoo!, Finance, Siashdot та бюро перепису США (U.S. Census Bureau).

MySQL є чудовим інструментом вивчення баз даних, завдяки простоті її інсталяції та використання, а також виключно скромним вимогам у відношенні дискового простору та пам'яті.

Існує дуже багато систем управління базами даних, з якими можна зрівняти MySQL, але MySQL пропонує таку комбінацію продуктивності, ціни та можливостей, яку навряд чи зможуть запропонувати інші.

Продуктивність

Система MySQL забезпечує значну швидкодію виконання запитів.

На Web-вузлі MySQL представлені результати зрівняння MySQL з іншими системами управління базами даних. Ці результати демонструють помітну перевагу MySQL з продуктивності в порівнянні з конкурентами. Результати вказаних тестів можна знайти за наступною адресою: www. mysql.com/information/benchmarks.html

Контрольне тестування, виконане журналом eWeek в 2009 році, показало, що у підтримці Web- додатків на базі Java на машинах з чотирма процесорами та Windows MySQL не поступається за продуктивністю Oracle. Вказані дві системи управління базами даних працюють швидше DB2 від IBM, SQL Server від Microsoft та ASE від Sybase. Слід зазначити, що в даному випадку один з найкращих продуктів виявляється безкоштовним, тоді як інший має ціну біля 160 тис. доларів. Більш докладну інформацію можна знайти за адресою: www.eweek.com/article2/0,3959,293,00.asp.

Для розробників MySQL швидкість завжди є ключовим параметром. Нові можливості додавалися в пакет MySQL після того, як їх вдавалося реалізувати без шкоди для продуктивності. Іноді це означало, що деякі можливості додавалися не так швидко, як хотілося б користувачам, але завжди гарантувало швидку роботу MySQL.

Ціна

Ціна, мабуть, є самим простим критерієм для зрівняння. В більшості випадків MySQL взагалі є безкоштовним додатком. Ліцензія GPL дозволяє використовувати це програмне забезпечення, змінюючи вихідний код та

поширювати MySQL серед інших користувачів, які при цьому теж повинні підкорятися вимогам ліцензії GPL.

Головними конкурентами MySQL є комерційні продукти, для яких використовуються складні цінові схеми, залежні від запропонованого використання, числа процесорів в кожному сервері та числа користувачів, які будуть до них підключатися. Oracle, SQL Server від Microsoft, DB2 від IBM, та ASE від Sybase можуть коштувати від десятків тисяч доларів в звичайних умовах до сотень тисяч доларів на серверах з великою кількістю процесорів та багаточисельним підключенням клієнтів.

Іноді MySQL зрівнюють з іншими відкритими системами управління базами даних, наприклад, з PostgreSQL та Firebird. Із усіх таких систем тільки за MySQL стоїть єдина компанія, яка володіє усіма правами інтелектуальної власності та правом продажу комерційних ліцензій, включаючи обов'язки та компенсації, обумовлені для організацій з великою кількістю користувачів.

Другою категорією програмного забезпечення, в якій інколи проводять зрівняння з MySQL, являються недорогі системи управління базами даних, які не відносяться до типу «клієнт/сервер» і призначені для використання в домашніх умовах чи в малому бізнесі (наприклад, Microsoft Access та Filemarker Pro). Частіше такі системи можуть запропонувати користувачу простий та зручний графічний інтерфейс, але усі програми цієї категорії відрізняються недостатньою функціональністю, в них не достатньо стабільності, масштабованості, а також швидкості, необхідної для виконання важливих додатків.

Стабільність

Розробники MySQL завжди вважали стабільність предметом особливої важливості. Всі версії MySQL, які випускаються в бінарному вигляді – навіть альфа – версії, - повинні пройти систему тестування MySQL (MySQL Test Suite). В ході такого тестування перевіряються функції та інші можливості продукту, як і операції, в яких дефекти були виявлені та виправлені раніше, - таким чином, гарантується, що виявлені дефекти ніколи не будуть випадково повторені знову.

Розробники повинні вважати виправлення помилок задачею з більш високим пріоритетом по відношенню до інших задач розробки. Як правило, розробники припиняють виконання інших задач, поки не будуть виправлені усі помилки, які відносяться до їх сфери компетенції. Вважається правилом, що версії MySQL, повинні бути повністю вільні від відомих та відтворюючих помилок. Природно, деякі проблеми буває неможливо вирішити без того, щоб в результаті не з'явилася будь-яка інша проблема десь в іншому місті. Це відноситься і до розробки версій програмного забезпечення, які не містять кардинальних змін, які спроможні вплинути на стабільність продукту. Виявлені проблеми документуються та виправляються для всіх більш пізніх версій.

Контрольні питання

1. Які бази даних Вам відомі?
2. В чому полягають переваги застосування СУБД MySQL?

ПРАКТИЧНА РОБОТА №2

Тема: Ази роботи в MySQL
Мета: Закріпити навики роботи в програмі MySQL

Консольний клієнт mysql часто називають «термінальним монітором» або просто «монітором». Користувачі операційної системи Linux можуть набирати ім'я даної програми з параметрами з будь-якої точки системи. Власникам Windows для запуску mysql необхідно вибрати наступний пункт системного меню: **Пуск/Програми/ MySQL/ MySQL Server 5.0/ MySQL Command Line Client.** При такій формі запуску утиліта mysql не дозволяє вводити ніякі параметри, крім того, для роботи з іншими утилітами, які входять в поставку MySQL, необхідний доступ до командного рядка. Отримати його можна, звертаючись до наступного пункту: **Пуск/Програми/ Стандартні/Командний рядок.** В результаті буде відчинено вікно, яке представлене на рис.2.1

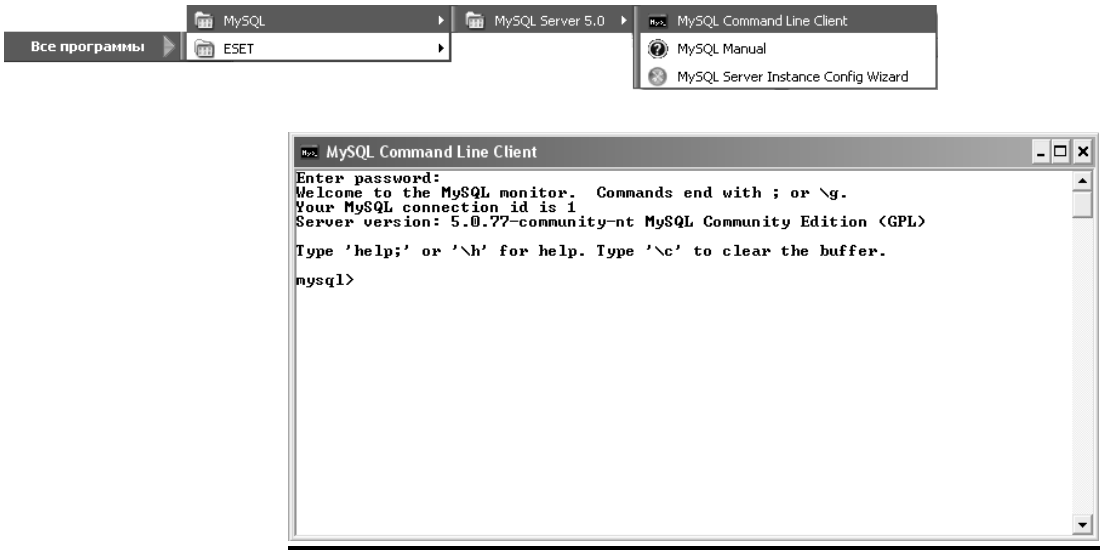

Рис 2.1 Вхід в MySQL

Зауваження. На даному рисунку колір вікна та його розміри вибрані відмінними від тих, що виставляються по замовчуванню системою. Можна самостійно змінювати кольорову схему командного рядка та його розміри у властивостях системного меню вікна (лівою клавішею миші по заголовку вікна рис 2.2) і у відкритому вікні поставити маркер, де потрібно.

Коли користувач вводить команду, вона надсилається на сервер для виконання, і якщо немає помилок у синтаксисі, на екран виводиться результат у вигляді таблиці, а на новому рядку – запрошення mysql >, після якого можна вводити команди.

На першому рядку таблиці з результатами містяться заголовки стовбців, а в наступних рядках – відповідь сервера на запит. Взагалі заголовками стовбців стають імена, які отримують з таблиць бази. Якщо ж вилучається не стовбець таблиці, а значення виразу (як це відбувається у приведеному вище прикладі) mysql дає стовбцю ім'я виразу, яке запитується. Після цього повідомляється

кількість рядків, які вертаються (1 row in set –один рядок в результаті) та час виконання запиту.

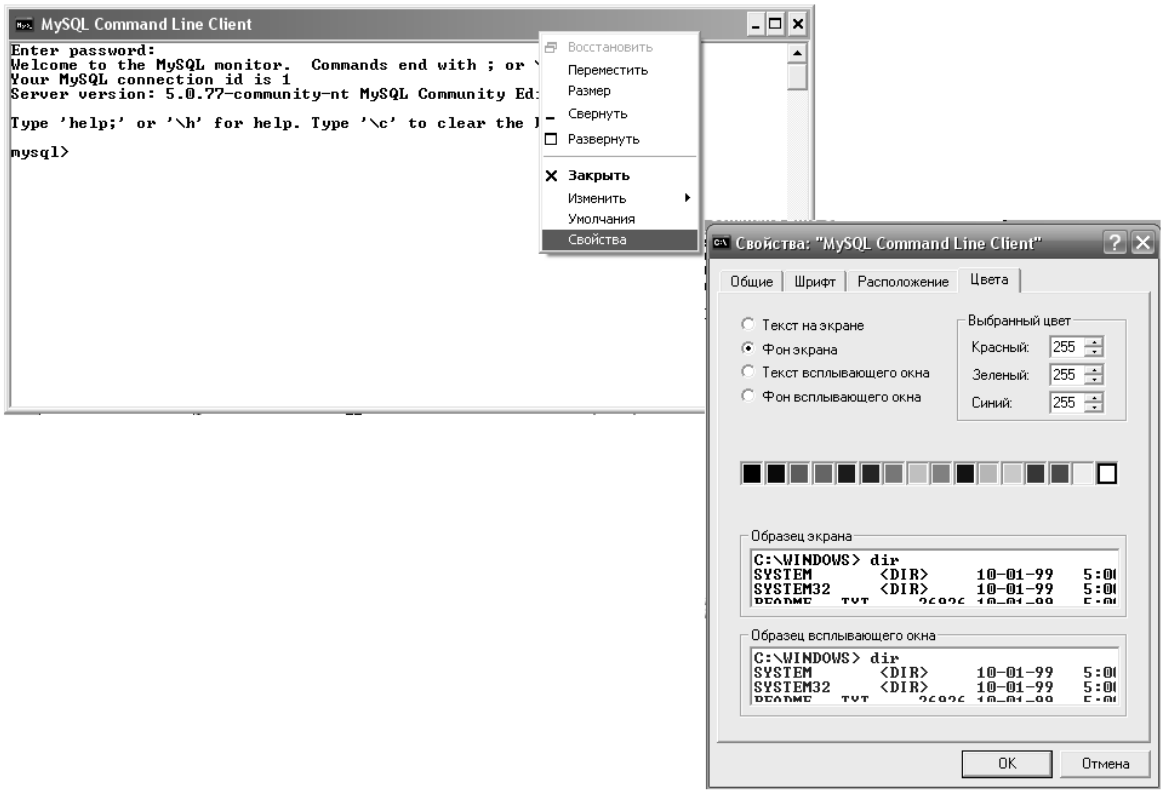

Рис 2.2 Зміна властивостей командного вікна в програмі MySQL

Для вводу ключових слів можна використовувати будь-який регістр символів. Так приведені на рис 2.3 запити абсолютно ідентичні.

Якщо команда не прибирається на одному рядку, можливий перехід на другий рядок, після натискання клавіші ‹Enter› - запит надсилається серверу тільки після того, як консольний клієнт mysql зустріне символ «;». Запрошення командного рядку після вводу першого рядку цього запиту змінюється з mysql › на -› (див. рис.2.4).

Таким чином програма mysql показує, що виразу, який завершився, вона поки що не отримала і чекає його повного вводу. Так утиліта mysql веде себе, коли чекає завершення рядку, який знаходиться у подвійних (“) чи одинарних (‘) лапках.

Вже введені раніше команди не обов’язково вводити знову, для цього достатньо їх викликати клавішами ‹↑› та ‹↓›, очистити рядок запиту можна за допомогою ‹ Esc›.

Параметри в утилітах MySQL мають дві форми: повну, яка починається з двох дефісів – user, та стислу, яка починається з одного дефісу –u. Можна використовувати обидва варіанти, але для ряду параметрів маємо лише повну форму.

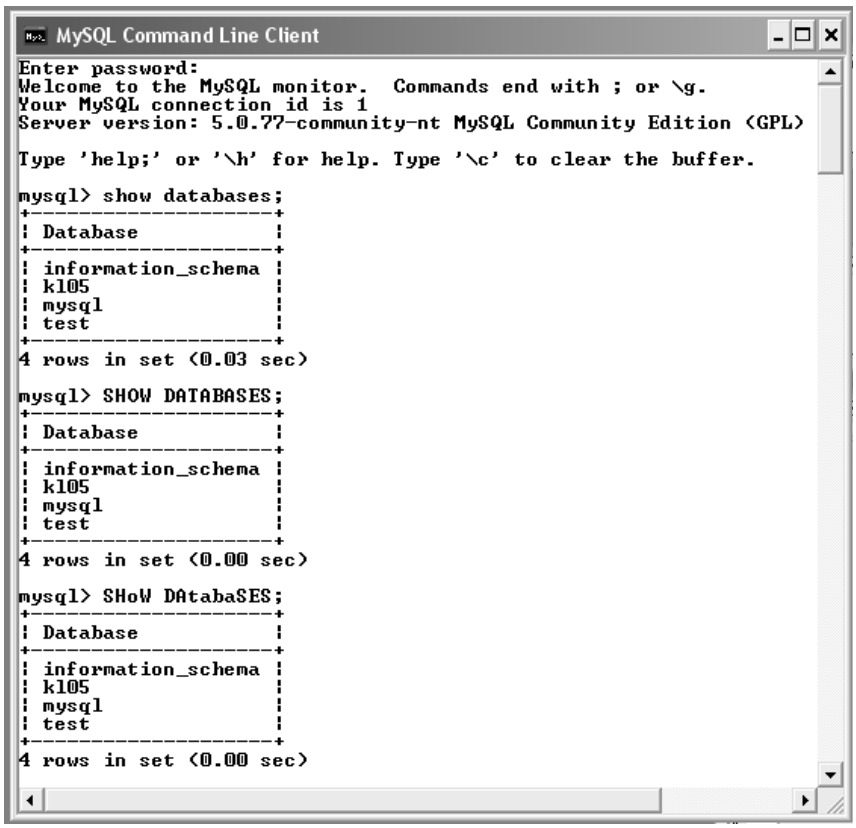

Рис 2.3. Імена інструкцій та ключові слова не залежать від регістра символів.

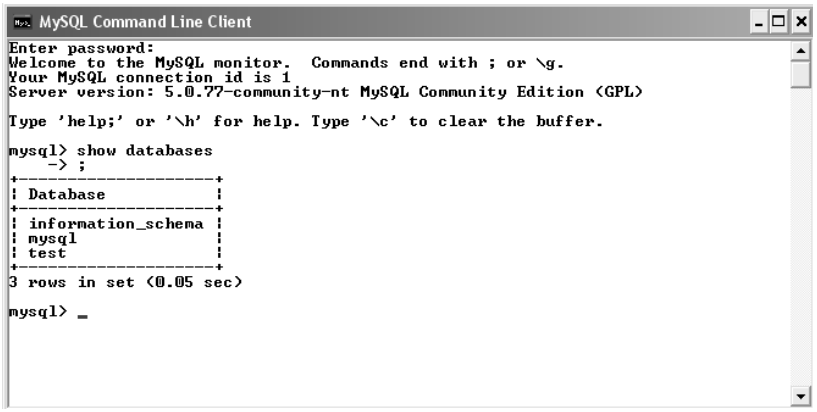

Рис 2.4 Недостає символа «;», програма MySQL не виконує попередню команду

Контрольні питання

1. Як увійти в програму MySQL?
2. як змінити колір фону монітора в MySQL?
3. Що потрібно ввести вкінці рядка для завершення запиту в MySQL?
4. Що означає символ «– >» в MySQL?
5. Чи є різниця в застосуванні регістра символів?
6. Для того, щоб попередньо набрану команду не вводити знову, потрібно натиснути…
7. Щоб швидко очистити рядок від команд, потрібно …
8. Параметри в утилітах MySQL мають … форми

ПРАКТИЧНА РОБОТА № 3

Тема: Створення баз даних та таблиць
Мета: Здобути навички створення баз даних та таблиць

Створення баз даних зводиться до створення нового підкаталогу в каталозі даних C:\mysql5\data. Створення баз даних засобами SQL відбувається за допомогою оператора CREATE DATABASE. У лістінгу 3.1 приводиться приклад створення бази даних KL05.

Лістінг 3.1. Створення бази даних KL05

```
mysql> create database KL05;
Query OK, 1 row affected (0.00 sec)
```

Після виконання запиту з лістінга 3.1, поглянувши у каталог C:\mysql5\data, можна виявити новий каталог **KL05**. Максимальна довжина імені бази даних складає 64 знаки і може включати літери, цифри та символи «_» та «$». Ім'я може починатися з цифри, але не повинно повністю складатися з цифр.

Контролювати створення бази даних, а також дізнаватися імена існуючих баз даних можна за допомогою оператора SHOW DATABASES (лістінг 3.2).

Лістінг 3.2. Використання оператора SHOW DATABASES.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| kl05               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)
```

Як видно з лістінга 3.2, оператор SHOW DATABASES повернув ім'я чотирьом базам даних. Бази даних information_schema та mysql є службовими та необхідні для підтримання сервера MySQL в працездатному стані – в них зберігається інформація про облікові записи, регіональних налаштуваннях та інш.

База даних kl05 була створена за допомогою SQL- запиту (див лістінг 3.1) База даних test являється порожньою і створюється при установці MySQL разом з системними базами даних information_schema та mysql.

Видалення баз даних можна здійснити за допомогою оператора DROP DATABASE, за яким слідує ім'я бази даних (лістінг 3.3).

Лістінг 3.3. Видалення бази даних kl05

```
mysql> drop database kl05;
Query OK, 1 row affected (0.06 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.00 sec)
```

Після виконання оператора DROP DATABASE можна переконатися, що таблиця kl05 була видалена. Якщо виконується спроба створення вже існуючої бази даних, повертається помилка (лістінг 3.4).

Лістінг 3.4. Неможливо створити базу даних kl05: база даних існує

```
mysql> create database KL05;
ERROR 1007 (HY000): Can't create database 'kl05'; database exists
```

Часто така поведінка неможлива, особливо в великих дампах, розташованих в sql – файлах, які виконуються в пакетному режимі. Для попередження помилок поряд з оператором CREATE DATABASE застосовується конструкція IF NOT EXISTS, при наявності якої база даних створюється, якщо вона ще не існує, якщо ж вона існує – ніяких змін не відбувається (лістінг 3.5).

Лістінг 3.5. Використання конструкції IF NOT EXISTS.

```
mysql> create database if not exists kl05;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Як бачимо з лістінга 3.5, після видалення бази даних kl05 перший оператор CREATE DATABASE повертає відповідь, в якому йде мова про те, що був задіяний один рядок (1 row affected), тобто база даних створюється. Другий оператор CREATE DATABASE вертає повідомлення, що запит не зачепив жодного рядка (0 row affected), тобто база даних не створюється, однак і помилка не видається.

Такий самий механізм розроблений і для оператора DROP DATABASE, додавання ключового слова IF EXISTS, видаляє базу даних, якщо вона існує, та не виконує ніяких дій, якщо база даних відсутня.

Спеціального SQL – оператора, призначеного для перейменування бази даних, не існує, але цю операцію легко виконати, перейменувавши каталог бази даних.

Перед тим як створити таблицю, слід вибрати базу даних, з якою буде виконуватися робота. Ця операція виконується за допомогою інструкції USE (лістінг 3.7).

Лістінг 3.7. Вибір бази даних.

```
mysql> use kl05;
Database changed
```

Зауваження. Перед тим як працювати з таблицями, завжди виконується вибір бази даних. Для створення таблиці використовують оператор CREATE TABLE, після якого слідує ім'я таблиці, яку створюють, та в дужках – структура таблиці (лістінг 3.8).

Лістінг 3.8. Створення таблиці student

```
mysql> create table student (numb int, famil text,
    -> adr text, born datetime, mark int);
Query OK, 0 rows affected (0.14 sec)
```

В круглих дужках через кому вказуються імена стовбців та їх тип. Так, створена таблиця **student** має п'ять полів: numb – порядковий номер студентів за списком групи (тип int – цілочислений), поле famil – прізвище студентів (тип text, при заповненні рядків пишеться в ‘ ’), адреса студентів вказується в стовпці adr, в полі born – день і рік народження, оцінки студента зводяться до стовпця mark, що має тип int. Максимальна довжина імен таблиць та стовбців складає 64 знака і може включати літери, цифри та символи «_» та «$». Ім'я може починатися з цифри, але не повинно повністю складатися з цифр.

Зауваження. В операційній системі Windows імена таблиць та баз даних не залежать від регістра, але тоді як в UNIX – подібних операційних системах - залежать. Це пов'язано з тим, що база даних являється директорією, а таблиці – файлами, імена яких співпадають з іменами баз даних та таблиць. В файловій системі Windows імена файлів не залежать від регістра, в той час як в UNIX імена, які відрізняються тільки регістром, вважаються різними файлами.

Проконтролювати створення таблиці **student** можна за допомогою оператора SHOW TABLES, який вертає перелік таблиць поточної бази даних, серед яких і повинна бути новостворена таблиця **student.**

Лістінг 3.9. Використання оператора SHOW TABLES.

```
mysql> show tables;
+----------------+
| Tables_in_kl05 |
+----------------+
| student        |
+----------------+
1 row in set (0.00 sec)
```

Для того, щоб продивитися зміст іншої бази даних, слід попередньо вибрати її з використанням оператора USE (лістінг 3.10).

Лістінг 3.10. Використання оператора USE.

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| kl05               |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.00 sec)

mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)
```

Як бачимо з лістінга 3.10, база даних test не має жодної таблиці.

Видалення таблиці виконується за допомогою оператора DROP TABLE (лістінг 3.11).

Лістінг 3.11. Використання оператора DROP TABLE.

```
mysql> use KL05;
Database changed
mysql> show tables;
+----------------+
| Tables_in_kl05 |
+----------------+
| student        |
+----------------+
1 row in set (0.00 sec)

mysql> drop table student;
Query OK, 0 rows affected (0.19 sec)

mysql> show tables;
Empty set (0.00 sec)
```

Зауваження. Разом зі звертанням до таблиці за її ім'ям, наприклад student, існує поширене звертання, яке включає в себе ім'я бази даних, наприклад, KL05.student. Будь-який оператор SQL може звертатися до бази даних за цим ім'ям. Так, оператор DROP TABLE з лістінгу 3.11 може виглядати наступним чином: DROP TABLE KL05.student. При використанні поширеного синтаксису дозволяється не виконувати вибір бази даних оператором USE, так як база даних явно вказується у запиті.

Лістінг 3.12 Використання розширеного формату при видаленні таблиці

```
mysql> drop table kl05.student;
Query OK, 0 rows affected (0.09 sec)

mysql> show tables;
Empty set (0.00 sec)
```

Для того, щоб отримати опис стовбців бази даних, можна використати оператор DESCRIBE.

Лістінг 3.13. Використання оператора DESCRIBE.

```
mysql> describe student;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| numb  | int(11)  | YES  |     | NULL    |       |
| famil | text     | YES  |     | NULL    |       |
| adr   | text     | YES  |     | NULL    |       |
| born  | datetime | YES  |     | NULL    |       |
| mark  | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Як видно з лістінгу 3.14, ім'я таблиці приведено в розширеному форматі kl05.student, однак досить допустимо вказати одне ім'я student, якщо в якості поточної бази даних вибрана kl05.

Лістінг 3.14 Використання розширеного формату при описі таблиці

```
mysql> describe kl05.student;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| numb  | int(11)  | YES  |     | NULL    |       |
| famil | text     | YES  |     | NULL    |       |
| adr   | text     | YES  |     | NULL    |       |
| born  | datetime | YES  |     | NULL    |       |
| mark  | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Оператор DESCRIBE повертає таблицю, кожен рядок якої відповідний стовбцю таблиці **student**. Імена стовбців та їх типи приводяться в першому та другому стовбцях відповідно.

Контрольні питання

1. Як створити базу даних?
2. Створення таблиці відбувається оператором…
3. Для видалення БД або таблиці застосовують оператор…
4. Оператором SHOW відбувається…
5. Щоб продивитись опис шапки таблиці потрібно…
6. Максимально можлива довжина імені таблиці/БД може складати…
7. Обрання БД відбувається оператором…
8. Як видалити таблицю з БД, яку не було обрано попередньо?
9. Як перейменувати базу даних?
10. Чи може ім'я БД/таблиці мати вигляд «1234567890»?

ПРАКТИЧНА РОБОТА №4

Тема: Типи даних
Мета: Отримати інформацію про типи даних програми MySQL

MySQL підтримує декілька типів даних.

- Числові дані – до них відносяться цілі числа, які не містять дробової частини, а також дійсні числа, які складаються із послідовності цифр, розділених крапкою (наприклад, 56.45).
- Строкові дані – послідовність символів, які знаходяться в одинарних або подвійних лапках : 'Hello world', '123', " MySQL". В якості стандарту в SQL визначаються одинарні лапки, тому для сумісності з іншими базами даних рекомендують використовувати саме їх.
- Календарні дані – спеціальний тип для позначення дати та часу, може приймати різну форму, наприклад строкову «2005-04-28» або числову 20050428. Основною характеристикою рисою цього типу даних є їх зберігання в єдиному внутрішньому форматі, який дозволяє виконувати операції складання та віднімання, незалежно від зовнішнього уявлення.
- NULL – спеціальний тип даних, який означає відсутність даних.

Ч И С Л О В І Д А Н І

Числові дані поділяються на крапкові (BOOLEAN, INTEGER, DECIMAL) та наближені (FLOAT, REAL, DOUBLE PRECISION).

Зауваження. Тут та надалі необов'язкові елементи синтаксису будуть знаходитися в квадратних дужках, тобто запис TINYINT [{M}] визначає, що елемент може бути записаний і як TINYINT, і як TINYINT (M).

Як видно з табл.1, СУБД MySQL має п'ять цілих типів: TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT. Відмінність між ними полягає в діапазоні величин, які можна зберігати в стовбцях такого типу. Чим більший діапазон значень у типі даних, тим більше пам'яті він потребує.

Зауваження. Тип INT має значення INTEGER.

Цілі типи даних можуть бути оголошені додатніми. Для цього після оголошення типу слід використовувати ключове слово UNSIGNED. В цьому випадку елементам даного стовбця не можна буде привласнити від'ємне значення, а припустимий діапазон, який може приймати тип, подвоюється. Так, тип TINYINT може приймати значення від [-128 до 127], а TINYINT UNSIGNED – від 0 до 255.

Таблиця 1. Точкові типи

Тип	Об'єм пам'яті	Діапазон
TINYINT [(M)]	1 байт	від -128 до 127 (від -2^7 до 2^7-1) від 0 до 255 (від 0 до 2^8-1)
SMALLINT [(M)]	2 байт	від -32768 до 32767 (від -2^{15} до $2^{15}-1$) від 0 до 65535 (від 0 до $2^{16}-1$)
MEDIUMINT [(M)]	3 байт	від –8388608 до 8388608 (від -2^{23} до $2^{23}-1$) від 0 до 16777215 (від 0 до $2^{24}-1$)
INT [(M)], INTEGER [(M)]	4 байт	від -2147683648 до 2147683648 (від-2^{31} до $2^{31}-1$) від 0 до 4294967295 (від 0 до $2^{32}-1$)
BIGINT [(M)]	8 байт	(від-2^{63} до $2^{63}-1$) (від 0 до 2^{64})
BIT [(M)]	(M+7)/8 байт	Від 1 до 64 бітів, в залежності від значення M
BOOL, BOOLEAN	1 байт	Або 1, або 0
DECIMAL [(M [,D])], DEC [(M [,D])], NUMERUC [(M [,D])]	M+2 байт	Підвищена точність; залежить від значення M і D

При оголошенні цілого типу задається кількість відведених під число символів M (від 1 до 255). Це необов'язкова вказівка кількості символів, які виводяться, використовується для доповнення пропусками зліва від значень символів, які виводяться, менших, ніж завдана ширина стовбця. Однак обмежень ні на діапазон, ні на кількість розрядів не накладається. Якщо число символів, необхідних для виводу числа, перевищує M, під стовбець буде виведено більше символів. Якщо додатково вказаних необов'язковий атрибут ZEROFILL, вільні позиції по замовчуванні заповнюються нулями зліва. Наприклад, для стовбця, оголошеного як INT(5) ZEROFILL, величина 4 відображається як 00004.

Тип BIT [(M)] призначений для зберігання бітових полів. Параметр M вказує число бітових значень, які може приймати поле (від 1 до 64). Якщо параметр M не вказаний, то по замовчуванню приймає значення 1.

Зауваження. Тип BIT доданий в MySQL, починаючи з версії 5.0.3

Тип BOOLEAN є синонімом для TINYINT (1). Значення 1 розглядається як істина (true), а 0 як неправда (false).

Тип DECIMAL, а також його синоніми DEC і NUMERUC, призначені для величин великої точності, наприклад для грошових даних. Необхідна точність задається при оголошенні стовпця даних одного з цих типів, наприклад:

salary DECIMAL(5,2)

В цьому прикладі цифра 5 визначає загальне число символів, які відводяться під число, а цифра 2 завдає кількість знаків після коми. Отже, в цьому випадку проміжок величин, які можуть зберігатися в стовбці salary, містить від -99.99 до 99.99 (в дійсності для даного стовбця MySQL забезпечує можливість зберігання чисел дуже близько до 999.99, оскільки допускається не зберігати знак для додатних чисел).

Зауваження. Перший параметр M може приймати максимальне значення, яке дорівнює 64, другий параметр D – 30.

Величини типів DECIMAL, DEC і NUMERUC зберігаються як рядки, а не як двійкові числа з плаваючою крапкою, щоб зберегти точність подання цих величин в десятковому вигляді. Якщо другий параметр дорівнює 0, то величини DECIMAL і NUMERUC не мають десяткового знаку чи дробової частини.

Для подання дійсних типів в СУБД MySQL є три типи: FLOAT, DOUBLE, DESIMAL.

Зауваження. Тип DOUBLE має два синоніми: PRECISION і REAL. Тип DECIMAL має синонім NUMERIC.

Характеристики наближених типів подані у таблиці 2.

Таблиця 2. Наближені типи

Тип	**Об'єм пам'яті**	**Діапазон**
FLOAT [(M, D)]	4 байта	Мінімальне значення $\pm 1.175494351 \cdot 10^{-39}$ Максимальне значення $\pm 3.402823466 \cdot 10^{38}$
DOUBLE (M, D)], REAL [(M, D)], DOUBLE PRECISION[(M, D)]	8 байт	Мінімальне значення $\pm 2.225073858507214 \cdot 10^{-308}$ Максимальне значення $\pm 1.797693134862315 \cdot 10^{308}$

Діапазон дійсних чисел крім максимального значення має також мінімальне значення, яке характеризує точність даного типу. Параметр M в табл.2 завдає число типів для відображення всього числа, а D – дробової чистини.

Числові типи даних з плаваючою крапкою також можуть мати параметр UNSIGNED. Як і в цілочисельних типах, цей атрибут запобігає зберігання в окремому стовбці від'ємних величин, але, на відміну від цілочисельних типів, максимальний проміжок для величин стовпця залишається попереднім.

Зауваження. Наближені числові дані можуть задаватися в звичайній формі, наприклад 45.67, і в так званій науковій нотації, наприклад 5.456E-02 або 4.674E+04. Символ E вказує на те, що число перед ним слід помножити на 10 в степені, яка вказана після E. Тобто приведені в прикладах числа можна записати як 0.05456 та 46740. Такий формат введений для зручності запису , так як числа в 300 степені незручно подавати в звичайному десятковому вигляді.

При виборі стовбців для формування структури таблиці необхідно спрямовувати увагу на розмір, який займає той чи інший тип даних: якщо значення, які розміщені в базі даних, ніколи не будуть виходити за межі 100, не слід обирати тип більше TINYINT. Якщо у полі стовпця пропонується зберігати тільки ціла чисельні дані, то використання атрибуту UNSIGNED дозволить збільшити діапазон у два рази.

СТРОКОВІ ДАНІ

Для строкових типів даних максимальний розмір та вимоги до пам'яті приведені в таблиці 3. Тут L – це довжина зберігаємого в комірці рядка, а байти, приплюсовані до L,- прикладні витрати для зберігання довжини рядка.

Таблиця 3.Строкові типи даних

ТИП	**Об'єм Пам'яті**	**Максимальний розмір**
CHAR (M)	M символів	M символів
VARCHAR (M)	L+1 символів	M символів
TINYBLOB, TINYTEXT	L+1 символів	2^8-1 символів
BLOB, TEXT	L+2 символів	2^{16}-1 символів
MEDIUMBLOB, MEDIUMTEXT	L+3 символів	2^{24}-1 символів
LONGBLOB, LONGTEXT	L+4 символів	2^{32}-1 символів
ENUM ('value1','value2',…)	1 або 2 байти	65535 елементів
SET ('value1','value2',…)	1,2,3,4 або 8 байтів	64 елементів

Тип CHAR дозволяє зберігати рядок фіксованої довжини M, його доповнює тип VARCHAR, який дозволяє зберігати змінні рядки довжиною L. Значення M може приймати значення від 0 до 65535.

Зауваження. До версії MySQL 5.0.3 значення типів CHAR та VARCHAR могло приймати значення тільки від 0 до 255.

При виборі строкового типу даних для стовпця слід мати на увазі, що для змінних рядків VARCHAR вимагається кількість символів, рівне довжині рядка плюс один байт, в той час як тип CHAR (M), незалежно від довжини рядка, використовує для її зберігання усі M символи. У той же час тип CHAR обробляється ефективніше змінних типів, так як завжди заздалегідь відомо, де закінчується черговий блок даних. Порівняння типів CHAR та VARCHAR приведено в таблиці 4.

Таблиця 4. Порівняння типів CHAR та VARCHAR

Значення	**CHAR (4)**		**VARCHAR (4)**	
	Представ-лення	Число байтів	Представ-лення	Число байтів
' '	' '	4 байта	' '	1 байт
'ab'	' ab'	4 байта	'ab'	3 байта
'abcd'	'abcd'	4 байта	'abcd'	5 байтів
'abcdefgh'	'abcd'	4 байта	'abcd'	5 байтів

При створенні таблиці не можна змішувати стовбці CHAR та VARCHAR. Якщо так стається СУБД MySQL змінить тип стовбців згідно з правилом: у випадку, коли в таблиці присутній хоч один стовбець змінної довжини, усі стовбці типу CHAR приводяться до типу VARCHAR.

Зауваження. Починаючи з версії 4.1.2, типи CHAR та VARCHAR розглядають рядки як послідовність символів. Це означає, що при використанні багатобайтних кодувань, наприклад UNICODE, розмір рядку в байтах буде більше, ніж у символах. Для сумісності зі старими версіями MySQL виведені

два спеціальних типи: BINARY та VARBINARY, які еквівалентні типам CHAR та VARCHAR, але рядок в них розглядається як послідовність байтів, а не символів. До BINARY – рядків не застосовується кодування та сортуються вони як звичайні послідовності байтів.

Типи BLOB та TEXT в СУБД MySQL в усьому аналогічні та розрізняються тільки в деталях. Наприклад, при виконанні операцій над стовбцями типу TEXT враховується кодування, а типу BLOB – ні.

Тип TEXT звичайно використовується для зберігання великих об'ємів тексту, в той час як BLOB – для великих двійкових об'єктів, таких як електронні документи, зображення, звуки та інші.

Зауваження. Основною відмінністю типу TEXT від CHAR та VARCHAR являється підтримка можливостей повнотекстового пошуку.

До особливих типів даних відносяться ENUM та SET. Рядки цих типів приймають значення із заздалегідь завданого переліку допустимих значень. Основна відмінність між ними полягає в тому, що значення типу ENUM повинно містити точно одне значення із вказаної множини, тоді як стовбці SET можуть містити будь-який або усі елементи заздалегідь завданої множини одночасно. Так , значення для стовбця, оголошеного як ENUM ('y','n'), можуть приймати тільки два значення : або 'y', або 'n'.

Для типу SET, також як і для типу ENUM, при оголошені задається перелік можливих значень, але комірка може приймати будь-яке значення із переліку, а порожній рядок означає, що ні один із елементів не вибраний. Наприклад, значення для стовбця SET('y', 'n') можуть приймати значення ('y', 'n'), ('y'), ('n') та порожню множину ( ).

Типи ENUM та SET можна назвати строковими лише частково, так як при оголошені вони завдаються переліком рядків, але у внутрішньому представленні бази даних елементи множин зберігаються у вигляді чисел. Елементи типу ENUM нумеруються послідовно, починаючи з 1. В залежності від кількості елементів в переліку під стовбець може відводитися 1 байт (до 256 елементів у переліку) чи 2 байти (від 256 до 65535 елементів у переліку).

Елементи множини SET відпрацьовуються як біти, розмір типу при цьому також визначається числом елементів у переліку : 1 байт (від 1 до 8 елементів) , 2 байт (від 9 до 16 елементів) , 3 байт (від 17 до 24 елементів), 4 байт (від 25 до 32

К А Л Е Н Д А Р Н І Д А Н І

СУБД MySQL має п'ять видів стовбців для зберігання календарних типів даних: DATE, DATETIME, TIME, TIMESTAMP та YEAR (табл..5). Тип DATE призначений для зберігання дати, TIME для часу доби, TIMESTAMP для представлення часу доби та дати. Тип TIMESTAMP призначений для представлення дати та часу доби у вигляді числа секунд, які пройшли з півночі 1 січня 1970. Тип даних YEAR дозволяє зберігати тільки рік.

Таблиця 5. Календарні типи даних

ТИП	**Об'єм пам'яті**	**Діапазон**
DATE	3 байти	від '1000-01-01' до '9999-12-31'
TIME	3 байти	від '-828:59:59' до '828:59:59'
DATETIME	8 байтів	від '1000-01-01 00:00:00' до '9999-12-31 00:00:00'
TIMESTAMP [(M)]	4 байти	від '1970-01-01 00:00:00' до '2038-12-31 59:59:59'
YEAR [(M)]	1 байт	від 1901 до 2155 для YEAR(4) від 1970 до 2069 для YEAR(2)

Для значень, які мають тип DATE та DATETIME, в якості першої цифри очікується рік або у форматі «YYYY», наприклад '2005-10-15', або у форматі «YY», наприклад '05-10-15'. Після року через дефіс вказується місяць у форматі «MM» (10), а потім день «DD» (15).

Зауваження. Будемо позначати формат дати як YYYY- MM- DD для дат виду 2005-10-15 і формат YY- MM- DD для дат виду 05-10-15. Для вказування часу буде використовуватись формат hh:mm:ss , де hh-години, mm-хвилини, ss-секунди, наприклад 10:48:56.

В типах TIME та DATETIME час приводиться в звичному форматі hh:mm:ss, де hh-години, mm-хвилини, ss-секунди. Дні, місяці, години, хвилини та секунда можна записувати як з ведучим нулем: 01, так і без нього: 1. Наприклад, всі слідуючи записи ідентичні:

2005-04-06 02:04:08
2005-4-06 02:04:08
2005-4-6 02:04:08
2005-4-6 2:04:08
2005-4-6 2:4:08
2005-4-6 2:4:8

В якості розділюю чого знаку між роками, місяцями, днями, годинами, хвилинами, секундами може виступати будь-який символ, відмінний від цифри. Так, наступні значення ідентичні:

05-12-31 11:30:45
05.12.31 11+30+45
05/12/31 11*30*45
05@12@31 11^30^45

Дата та час доби можуть також бути представлені в форматах 'YYYYMMDDhhmmss' та 'YYMMDDhhmmss'. Наприклад, рядки '2005091528' та '050523091528' аналогічні '2005-05-23 09:15:28', проте рядок '051122129015' вже не може розглядатися як дата та час доби, так як значення

для хвилин дорівнює 90 і виходить за допустимий інтервал. Замість рядків допустимі і цілочисельні значення, наприклад, 2005091528 та 0523091528 розглядаються як ‘2005-05-23 09:15:28’.

Починаючи з версії MySQL 4.1.1, при вказуванні часу доби після секунд через крапку можна також вказувати мікросекунди, тобто використовувати розширений формат виду hh:mm:ss.ffffff, наприклад, ’10:25:14.000001’. Крім розширеного формату, можна використовувати короткі формати ‘HH:MM’ та ‘HH’ – замість пропущених величин будуть підставлені нульові значення.

Якщо час задається в недопустимому форматі, то в поле записується нульове значення . Нульове значення привласнюється полям тимчасового типу по замовчуванню, коли їм не привласнюється ініціююче значення (табл.6).

Формат типу TIMESTAMP співпадає з DATETIME , але у внутрішньому представлені дата зберігається у вигляді секунд, які вже минули з опівночі 1 січня 1970 року.

Таблиця 6. Календарні типи.

ТИП	НУЛЬОВЕ ЗНАЧЕННЯ
DATE	‘0000-00-00’
TIME	’00:00:00’
DATETIME	‘0000-00-00 00:00:00’
TIMESTAMP	00000000000000
YEAR	0000

Зауваження. Час у даному форматі зберігається як кількість секунд, яке пройшло з 1 січня 1970 року. Таке обчислення прийнято в операційній системі UNIX, а дата 01.01.1970 часто називається «Початком епохи UNIX» та вважається днем народження даної операційної системи.

Крім того, поле TIMESTAMP може автоматично отримувати значення поточної дати та часу доби для створення нового запису та зміни вже існуючої оператором UPDATE.

Зауваження. У версіях MySQL 4.0 та більш ранніх формат типу TIMESTAMP представляв собою число YYYYMMDDhhmmss. Починаючи з версії 4.1, формат представлення був змінений YYYY- MM- DD hh:mm:ss.

Якщо в таблиці декілька стовбців TIMESTAMP, при модифікації запису поточний час буде записуватись тільки в один зі стовбців (по замовчуванню перший). Можна також вказати явно стовбець, якому необхідно призначати поточну дату при створенні нового запису чи зміні старої. Для того щоб поля стовбця приймали поточну дату при створенні нового запису, слід після визначення стовбця додати запис DEFAULT CURRENT_ TIMESTAMP. Якщо потрібно, щоб поточний час виставлявся при модифікації вже існуючого запису, при використанні оператора UPDATE слід додати конструкцію NO UPDATE CURRENT_ TIMESTAMP.

Стовбці типу YEAR призначені для зберігання тільки року, вказівка параметру M дозволяє створити формат року. Так, для YEAR(2) рік представляється у вигляді двох чисел ‘YY’, наприклад 05 або 79, з діапазоном даних від 1970 року до 2069, а тип YEAR(4) дозволяє завдати тип у вигляді чотирьох чисел ‘YYYY’, наприклад 2005 або 1997 з діапазоном від 1901 до

2155 року. Якщо параметр M не вказаний, по замовчуванню вважається, що він дорівнює 4.

ТИП ДАНИХ NULL

Реляційна база даних дозволяє об'єднати багаточисельні дані в одну таблицю та за допомогою SQL – запитів проводити в ній різні маніпуляції, отримуючи результат у вигляді чисел та рядків, а також нових таблиць. При створені таблиці неминучі випадки, коли інформації недостатньо та для часини даних неможна визначити, яке значення вони отримують. Такі дані позначаються спеціальним типом – NULL.

Зауваження. Коментар в SQL починається з двох дефісів "- -" , все що розташовано праворуч, вважається коментарем. Безпосередньо після "- -" повинен слідувати пропуск. Окрім стандартного коментарю, MySQL містить ряд власних коментарів. Shell – коментар # діє аналогічно "- -", все що розташовано праворуч нього, являється текстом коментарю. C – коментар /**/ являється багатостроковим - коментар починається з "/*" і закінчується тільки тоді, коли зустрічається завершення "*/".

Наприклад, користувач повинен вказати своє прізвище, ім'я, по-батькові, але не може вказувати свій e-mail та адресу домашньої сторінки, навіть не дивлячись не те, що вони в нього є в наявності. Таким чином, якщо для полів *email* та *url* немає інформації, це не значить, що її немає у природі, просто на даний час вона невідома. Такі поля приймають значення NULL – відсутність інформації, тобто невизначене значення. Виконання арифметичних операцій з даними типу NULL завжди дає NULL, передача NULL в якості аргументу функцій також завжди приводить до значення типу NULL. Любі дії, які робляться над невизначеним значенням, приводять знову до невизначеного значення.

Контрольні питання

1. Які типи даних MySQL Ви знаєте?
2. Чим відрізняється тип INT від BIGINT?
3. Яка відмінність типів DATETIME і DATE?
4. Відмінність типів CHAR, VARCHAR, TEXT?
5. Що означає тип даних NULL? Які операції можна з ним проводити?
6. Для чого використовують атрибут UNSIGNED?
7. Чи може бути в таблиці два поля різного типу (CHAR, VARCHAR)? Поясніть.
8. Відмінність типу BINARY і VARBINARY?
9. Відмінність типів TEXT та BLOB?
10. Типи ENUM і SET?
11. Що пов'язано з датою 01.01.1970 і типом TIMESTAMP?
12. Види коментарів та їх формат?

ПРАКТИЧНА РОБОТА №5

Тема: Додавання даних в таблицю
Мета: Навчитись заповнювати даними таблиці

Після вдалого створення бази даних та таблиць перед розробником постає задача заповнення таблиць даними. Традиційно в реляційних базах даних для здійснення цієї операції застосовують три підходи:

- однорядковий оператор INSERT – додає в таблицю даних новий запис;
- багаторядковий оператор INSERT – додає в таблицю даних декілька нових записів;
- пакетна загрузка даних – слугує для додавання в таблицю даних із зовнішнього файлу.

Одностроковий оператор INSERT

Одностроковий оператор INSERT може використовуватися в декількох формах. Спрощений синтаксис першої форми виглядає наступним чином:

INSERT [INTO] tbl [(col_name,…)] VALUES (expression,…)

Даний оператор вставляє новий запис в таблицю tbl, значення запису перераховуються в переліку expression, порядок слідування стовбців може задаватися переліком col_name. Значення передаються в переліку після ключового слова VALUES. Розглянемо процес вставки на прикладі таблиці **student**.

Зауваження. Перегляд даних, які містяться в таблиці **student**, здійснюється за допомогою запиту SELECT * FROM **student**.

Додати новий запис в таблицю **student** можна за допомогою запиту, який представлений у лістінгу 5.1.

Лістінг 5.1. Додавання нового запису в таблицю student

```
mysql> insert into student values (1, 'Skrynnik', 'Kirovograd',
    -> '1983-09-19',34);
Query OK, 1 row affected (0.20 sec)

mysql> select * from student;
+------+----------+------------+---------------------+------+
| numb | famil    | adr        | born                | mark |
+------+----------+------------+---------------------+------+
|    1 | Skrynnik | Kirovograd | 1983-09-19 00:00:00 |   34 |
+------+----------+------------+---------------------+------+
1 row in set (0.00 sec)
```

Як бачимо з лістінгу 5.1, в таблицю **student** додався новий запис. Строкові значення необхідно розміщувати в лапки, в той час як числові значення допускається використовувати без них.

Зауваження. Використання лапок з числовими значеннями призведе до зменшення продуктивності, так як на перетворення типу MySQL знадобиться витратити додатковий час.

Перелік стовбців *col_name,* який розміщений після імені таблиці, дозволяє змінити порядок слідування стовбців при додаванні.

Лістінг 5.2. Зміна порядку слідування стовбців.

```
mysql> insert into student(famil, adr, born, numb, mark)
    -> values ('Kalyta', 'Nova Pavlovka', '1982-12-19', 2, 56);
Query OK, 1 row affected (0.25 sec)

mysql> select * from student;
+------+----------+---------------+---------------------+------+
| numb | famil    | adr           | born                | mark |
+------+----------+---------------+---------------------+------+
|    1 | Skrynnik | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | Kalyta   | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
+------+----------+---------------+---------------------+------+
2 rows in set (0.00 sec)
```

При вставці оператором INSERT не обов'язково вказувати усі стовбці. Ті стовбці, які не вказані в переліку, отримують значення по замовчуванню.

Лістінг 5.7. Додавання порожніх рядків.

```
mysql> insert into student() values ();
Query OK, 1 row affected (0.01 sec)

mysql> select * from student;
+------+----------+---------------+---------------------+------+
| numb | famil    | adr           | born                | mark |
+------+----------+---------------+---------------------+------+
|    1 | Skrynnik | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | Kalyta   | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
| NULL | NULL     | NULL          | NULL                | NULL |
+------+----------+---------------+---------------------+------+
3 rows in set (0.00 sec)
```

Зауваження. Відмінною рисою MySQL є те, що при використанні функцій «пробел» між ім'ям функції та круглими дужками не прийнятні, тобто запис *LAST_INSERT_ID( )* є правильним, а *LAST_INSERT_ID ( )* вже ні.

Зауваження. Результат функції можна вивести на екран за допомогою SQL - оператора SELECT.

Крім констант, в переліку значень *expression* можуть виступати вирази з участю стовбців з переліку *col_name.*

Ще однією формою оператора INSERT є синтаксис з ключовим словом SET:

INSERT [INTO] tbl SET col_name = expr,…

Оператор заносить в таблицю *tbl* новий запис, стовбець *col_name* в якому отримує значення *expr.*

Лістінг 5.13. Використання альтернативної форми оператора INSERT.

```
mysql> insert into student set numb=4, famil='Krymskiy', adr='Lugansk',
    -> born='1990-02-08', mark=85;
Query OK, 1 row affected (0.20 sec)

mysql> select * from student;
+------+----------+---------------+---------------------+------+
| numb | famil    | adr           | born                | mark |
+------+----------+---------------+---------------------+------+
|    1 | Skrynnik | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta   | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy | Lugansk       | 1990-02-08 00:00:00 |   85 |
+------+----------+---------------+---------------------+------+
3 rows in set (0.00 sec)
```

Форми оператора INSERT з ключовим словом SET дозволяє більш гнучко додавати записи. Якщо ім'я стовбця після SET не зустрічається, полю привласнюється значення по замовчуванню.

Багаторядковий оператор INSERT

Багаторядковий оператор INSERT співпадає за формою з однорядковим оператором. В ньому використовується ключове слово VALUES, після якого додається не один, а декілька переліків *expression.* Так, у лістінгу 5.15 додається відразу п'ять записів за допомогою одного оператора INSERT.

Результат запиту SELECT це завжди таблиця, яка нічим не відрізняється від таблиць, розташованих в базі даних, та називається *результуючою таблицею*.

Лістінг 5.15. Багаторядковий оператор INSERT.

```
mysql> insert student values (09,'Ovcharenko', 'Doneck', '1991-05-24', 54),
    -> (05,'Lavrenenko', 'odessa', '1985-10-12', 85),
    -> (06, 'Kuznecov', 'uman', '1978-11-30', 10);
Query OK, 3 rows affected (0.19 sec)
Records: 3  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
+------+------------+---------------+---------------------+------+
6 rows in set (0.00 sec)
```

Таким же чином, як і у випадку однорядкового варіанту, дозволяється змінювати порядок та зміст переліку доданих значень.

Лістінг 5.16. Зміна змісту стовбців в багаторядковому операторі INSERT.

```
mysql> insert student(famil) values('popov'),
    -> ('levchenko');
Query OK, 2 rows affected (0.22 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)
```

Контрольні питання

1. Існують… методи заповнення таблиці
2. Відмінність однорядкового від багаторядкового оператора заповнення таблиці полягає у…
3. За допомогою якого запиту в MySQL виводиться результуюча таблиця?
4. Результуюча таблиця – це…
5. Альтернативна форма заповнення таблиці полягає у …
6. Яка повинна бути форма запиту, якщо потрібно змінити порядок слідування стовпців результуючої таблиці?

ПРАКТИЧНА РОБОТА № 6

Тема: Вибірка даних
Мета: Отримати навички вибірки даних з заповненої таблиці

В даній лабораторній роботі описується оператор SELECT, який дозволяє витягувати дані з таблиці бази даних. Спрощений синтаксис оператора SELECT виглядає наступним чином:

SELECT select_expr FROM tbl

Кожен запит починається з ключового слова SELECT, після якого слідує список полів, які розділені комами select_expr, які будуть повернуті в результаті запиту. Ключове слово FROM, яке вказує, з якої таблиці витягуються дані, є необов'язковим та може невживатися.

Результат запиту SELECT – це завжди таблиця, яка нічим не відрізняється від таблиць, розташованих в базі даних, та називається *результуючою таблицею*. Більше того, як буде показано далі, результати запиту, виконаного за допомогою оператора SELECT, можуть бути використані для створення нової таблиці. Якщо результати двох запитів різних таблиць мають однаковий формат, то їх можна об'єднати в одну таблицю.

Зауваження:

Таблиця, яка отримана в результаті запиту, може стати предметом подальших запитів.

Зміна кількості та порядку слідування стовпців

Розглянемо запити вибірки даних на прикладі таблиці **student:** numb – порядковий номер студентів за списком групи (тип int – цілочислений), поле famil – прізвище студентів (тип text, при заповненні рядків пишеться в ‘ ’), адреса студентів вказується в стовпці adr, в полі born – день і рік народження, оцінки студента зводяться до стовпця mark, що має тип int.

Лістинг 6.1. Виборка даних за допомогою оператора SELECT

```
mysql> select famil from student;
+------------+
| famil      |
+------------+
| Skrynnik   |
| kalyta     |
| Krymskiy   |
| Ovcharenko |
| Lavrenenko |
| Kuznecov   |
| popov      |
| levchenko  |
+------------+
8 rows in set (0.00 sec)
```

Якщо потрібно вивести всі стовпці таблиці, необов'язково перераховувати їх імена після ключового слова SELECT, достатньо замінити цей список символом «*» (всі стовпці).

Лістинг 6.2. Використання символу «*» (всі стовпці)

```
mysql> select * from student;
+------+------------+--------------+---------------------+------+
| numb | famil      | adr          | born                | mark |
+------+------------+--------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd   | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk      | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck       | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa       | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman         | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL         | NULL                | NULL |
| NULL | levchenko  | NULL         | NULL                | NULL |
+------+------------+--------------+---------------------+------+
8 rows in set (0.00 sec)
```

До списку стовпців в операторі SELECT приходять в тому випадку, якщо необхідно змінити порядок слідування стовпців в результуючій таблиці або вибрати тільки частину стовпців.

Лістинг 6.3. Зміна числа та порядку слідування стовпців таблиці, що виводяться (результуюча таблиця)

```
mysql> select mark, famil from student;
+------+------------+
| mark | famil      |
+------+------------+
|   34 | Skrynnik   |
|   56 | kalyta     |
|   85 | Krymskiy   |
|   54 | Ovcharenko |
|   85 | Lavrenenko |
|   10 | Kuznecov   |
| NULL | popov      |
| NULL | levchenko  |
+------+------------+
8 rows in set (0.00 sec)
```

В лістингу 6.3 в запиті до таблиці **student** змінюється порядок слідування стовпців, це зручно для сприймання в тій формі, що задається користувачем.

Умови

Для виводу потрібних рядків, в яких містяться конкретні дані, потрібні користувачу, в програмі MySQL застосовуються різні форми оператора SELECT та спеціальне ключове слово WHERE, після якого слідує логічна умова. Якщо запис задовольняє такій умові, вона попадає до результату виборки, в іншому випадку такий запис відкидається.

Так, у лістингу 6.5 приводиться приклад запиту, в результаті якого виводяться *всі відомі дані* про тих студентів, які мають бал менше за 50.

Лістинг 6.5. Використання конструкції WHERE

```
mysql> select * from student where mark<50;
+------+----------+------------+---------------------+------+
| numb | famil    | adr        | born                | mark |
+------+----------+------------+---------------------+------+
|    1 | Skrynnik | Kirovograd | 1983-09-19 00:00:00 |   34 |
|    6 | Kuznecov | uman       | 1978-11-30 00:00:00 |   10 |
+------+----------+------------+---------------------+------+
2 rows in set (0.00 sec)
```

Умова може бути складною та об'єднуватися за допомогою логічних операторів. В лістингу 6.6 використовується складна умова: потрібно вивести всі відомі дані про того студента, у якого оцінка менше 50 балів, але й більше 30. Для об'єднання цих двох умов використовується оператор AND (І).

Зауваження:

Крім оператора AND (І), для об'єднання логічних виразів може використовуватися оператор OR (АБО).

Лістинг 6.6. Використання складної логічної умови

```
mysql> select * from student where mark<50 and mark>30;
+------+----------+------------+---------------------+------+
| numb | famil    | adr        | born                | mark |
+------+----------+------------+---------------------+------+
|    1 | Skrynnik | Kirovograd | 1983-09-19 00:00:00 |   34 |
+------+----------+------------+---------------------+------+
1 row in set (0.00 sec)
```

Як видно з попереднього лістингу, в результуючій таблиці були виведені дані лише того студента, який має бал і менше 50 і водночас більше 30.

Такого ефекту можна досягти застосуванням конструкції BETWEEN.

Лістинг 6.7. Використання конструкції BETWEEN

```
mysql> select * from student where mark between 30 and 50;
+------+----------+------------+---------------------+------+
| numb | famil    | adr        | born                | mark |
+------+----------+------------+---------------------+------+
|    1 | Skrynnik | Kirovograd | 1983-09-19 00:00:00 |   34 |
+------+----------+------------+---------------------+------+
1 row in set (0.00 sec)
```

Існує конструкція, протилежна конструкції BETWEEN – NOT BETWEEN, яка повертає записи, які не потрапляють до інтервалу між даними конкретного діапазону (лістинг 6.8).

Лістинг 6.8. Використання конструкції NOT BETWEEN

```
mysql> select * from student where mark not between 30 and 50;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
+------+------------+---------------+---------------------+------+
5 rows in set (0.00 sec)
```

Інколи потрібно витягувати записи, що задовольняють не діапазону, а списку, наприклад, потрібно вивести всі відомі дані про студентів, номер за списком яких 2, 5, 9. Для цього призначена конструкція IN.

Лістинг 6.9. Використання оператора IN

```
mysql> select * from student where numb in (2, 5, 9);
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
+------+------------+---------------+---------------------+------+
3 rows in set (0.02 sec)
```

Зауваження:

По аналогії з конструкцією BETWEEN, для IN існує зворотня конструкція NOT IN, яка дозволяє витягти з таблиці записи, які не задовольняють списку.

Лістинг 6.9а. Використання оператора NOT IN

```
mysql> select * from student where numb not in (2, 5, 9);
+------+----------+------------+---------------------+------+
| numb | famil    | adr        | born                | mark |
+------+----------+------------+---------------------+------+
|    1 | Skrynnik | Kirovograd | 1983-09-19 00:00:00 |   34 |
|    4 | Krymskiy | Lugansk    | 1990-02-08 00:00:00 |   85 |
|    6 | Kuznecov | uman       | 1978-11-30 00:00:00 |   10 |
+------+----------+------------+---------------------+------+
3 rows in set (0.00 sec)
```

Конструкції WHERE можуть використовуватися не тільки числові стовпці. В лістингу 6.10 показано виведення всіх відомих даних студента з Донецька. Дану конструкцію можна застосовувати у якості пошуку інформації за заданим виразом.

Лістинг 6.10. Робота з текстовими полями

```
mysql> select * from student where adr='Doneck';
+------+------------+--------+---------------------+------+
| numb | famil      | adr    | born                | mark |
+------+------------+--------+---------------------+------+
|    9 | Ovcharenko | Doneck | 1991-05-24 00:00:00 |   54 |
+------+------------+--------+---------------------+------+
1 row in set (0.00 sec)
```

У лістингу 6.11 з таблиці student виводяться *прізвища* лише тих студентів, які народилися до 1990 року.

Лістинг 6.11. Робота з датою

```
mysql> select famil from student where born<'1990-00-00';
+------------+
| famil      |
+------------+
| Skrynnik   |
| kalyta     |
| Lavrenenko |
| Kuznecov   |
+------------+
4 rows in set (0.01 sec)
```

Сортування

Як видно з лістингів, результат вибірки представляє собою записи, які розташовуються в порядку, в якому вони зберігаються в базі даних. Проте, часто потрібно відсортовувати значення по одному з стовпців. Це здійснюється за допомогою конструкції ORDER BY, яка слідує після виразу SELECT. Після цієї конструкції вказується стовпчик, за яким слід сортувати дані. Наприклад, необхідно відсортувати прізвища студентів за алфавітом.

Лістинг 6.13. Використання конструкції ORDER BY

```
mysql> select * from student order by famil;
+------+------------+--------------+---------------------+------+
| numb | famil      | adr          | born                | mark |
+------+------------+--------------+---------------------+------+
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk      | 1990-02-08 00:00:00 |   85 |
|    6 | Kuznecov   | uman         | 1978-11-30 00:00:00 |   10 |
|    5 | Lavrenenko | odessa       | 1985-10-12 00:00:00 |   85 |
| NULL | levchenko  | NULL         | NULL                | NULL |
|    9 | Ovcharenko | Doneck       | 1991-05-24 00:00:00 |   54 |
| NULL | popov      | NULL         | NULL                | NULL |
|    1 | Skrynnik   | Kirovograd   | 1983-09-19 00:00:00 |   34 |
+------+------------+--------------+---------------------+------+
8 rows in set (0.02 sec)
```

Сортування записів можна виконувати і за декількома стовпчиками. Нехай потрібно вивести в алфавітному порядку прізвища тих студентів, які мають бал, вище 50-ти.

Лістинг 6.14. Сортування прізвищ студентів, бал яких вище 50-ти

```
mysql> select famil, mark from student WHERE  mark>50 order by famil;
+------------+------+
| famil      | mark |
+------------+------+
| kalyta     |   56 |
| Krymskiy   |   85 |
| Lavrenenko |   85 |
| Ovcharenko |   54 |
+------------+------+
4 rows in set (0.00 sec)
```

За замовчуванням сортування проводиться в прямому порядку, записи розташовуються починаючи з найменшого значення і закінчуються найбільшим.

Лістинг 6.16. Сортування за замовчуванням

```
mysql> select famil, born, mark from student order by born;
+------------+---------------------+------+
| famil      | born                | mark |
+------------+---------------------+------+
| popov      | NULL                | NULL |
| levchenko  | NULL                | NULL |
| Kuznecov   | 1978-11-30 00:00:00 |   10 |
| kalyta     | 1982-12-19 00:00:00 |   56 |
| Skrynnik   | 1983-09-19 00:00:00 |   34 |
| Lavrenenko | 1985-10-12 00:00:00 |   85 |
| Krymskiy   | 1990-02-08 00:00:00 |   85 |
| Ovcharenko | 1991-05-24 00:00:00 |   54 |
+------------+---------------------+------+
8 rows in set (0.00 sec)
```

Змінити порядок сортування на зворотній можна за допомогою ключового слова DESC (лістинг 6.17).

Лістинг 6.17. Зворотнє сортування

```
mysql> select famil, born, mark from student order by born DESC;
+------------+---------------------+------+
| famil      | born                | mark |
+------------+---------------------+------+
| Ovcharenko | 1991-05-24 00:00:00 |   54 |
| Krymskiy   | 1990-02-08 00:00:00 |   85 |
| Lavrenenko | 1985-10-12 00:00:00 |   85 |
| Skrynnik   | 1983-09-19 00:00:00 |   34 |
| kalyta     | 1982-12-19 00:00:00 |   56 |
| Kuznecov   | 1978-11-30 00:00:00 |   10 |
| popov      | NULL                | NULL |
| levchenko  | NULL                | NULL |
+------------+---------------------+------+
8 rows in set (0.00 sec)

mysql> select famil, born, mark from student order by born ASC;
+------------+---------------------+------+
| famil      | born                | mark |
+------------+---------------------+------+
| popov      | NULL                | NULL |
| levchenko  | NULL                | NULL |
| Kuznecov   | 1978-11-30 00:00:00 |   10 |
| kalyta     | 1982-12-19 00:00:00 |   56 |
| Skrynnik   | 1983-09-19 00:00:00 |   34 |
| Lavrenenko | 1985-10-12 00:00:00 |   85 |
| Krymskiy   | 1990-02-08 00:00:00 |   85 |
| Ovcharenko | 1991-05-24 00:00:00 |   54 |
+------------+---------------------+------+
8 rows in set (0.00 sec)
```

Для прямого сортування також існує ключове слово ASC, оскільки за замовчуванням запису сортуються в прямому порядку, дане ключове слово часто не використовують.

Обмеження вибірки

Результат вибірки може містити сотні та тисячі записів. Їх виведення та обробка займають значний час і серйозно завантажують сервер бази даних. Тому інформацію часто розбивають на сторінки та пропонують її користувачеві. Витягування тільки частини запиту потребує менше часу та розрахунків, крім того, користувачеві часто буває вдосталь проглянути перші декілька записів. Посторінкова навігація використовується за допомогою ключового слова LIMIT, потім пишеться число записів, що виводяться. В лістигну 6.18 витягуються перші три записи, при цьому одночасно здійснюється зворотне сортування за полем famil.

Лістинг 6.18. Використання ключового слова LIMIT

```
mysql> select famil, born, adr from student order by famil desc
    -> limit 3;
+------------+---------------------+------------+
| famil      | born                | adr        |
+------------+---------------------+------------+
| Skrynnik   | 1983-09-19 00:00:00 | Kirovograd |
| popov      | NULL                | NULL       |
| Ovcharenko | 1991-05-24 00:00:00 | Doneck     |
+------------+---------------------+------------+
3 rows in set (0.02 sec)
```

Виконання попередніх виразів призводить до того, що виводиться інформація останніх трьох студентів, адже сортування виконувалось в черзі збування.

Для того щоб витягти наступні записи, використовують ключове слово LIMIT з двома цифрами, перша вказує позицію, починаючи з якої необхідно повернути результат, а друга цифра – це число виведених записів (лістинг 6.19).

Лістинг 6.19. Витягування записів починаючи с позицій після 3

```
mysql> select * from student order by famil;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
| NULL | levchenko  | NULL          | NULL                | NULL |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
| NULL | popov      | NULL          | NULL                | NULL |
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)

mysql> select famil, born, adr from student order by famil ASC
    -> limit 3, 3;
+------------+---------------------+--------+
| famil      | born                | adr    |
+------------+---------------------+--------+
| Lavrenenko | 1985-10-12 00:00:00 | odessa |
| levchenko  | NULL                | NULL   |
| Ovcharenko | 1991-05-24 00:00:00 | Doneck |
+------------+---------------------+--------+
3 rows in set (0.00 sec)
```

Використання функцій

При виконанні вибірки з бази даних часто потрібно виконувати специфічні задачі, для рішення яких зручно скористатися вбудованими функціями MySQL. Кожна функція має унікальне ім'я та може мати декілька аргументів (в тому числі жодного), які перераховуються через кому в круглих дужках разом після назви. Якщо аргументи функцій відсутні, круглі дужки все одно потрібно вказати, наприклад, NOW(). Відмінною рисою MySQL є те, що при використанні функцій пробіл між ім'ям функцій та круглими дужками використовувати непотрібно, тобто написання NOW() вірне, а вже NOW () невірне. Результат функцій підставляється на місце виклику функції.

В лістингу 6.20 приведений приклад використання функції VERSION(), яка повертає версію сервера MySQL.

Лістинг 6.20. Використання функції VERSION()

```
mysql> select version();
+---------------------+
| version()           |
+---------------------+
| 5.0.77-community-nt |
+---------------------+
1 row in set (0.00 sec)
```

Як видно з лістингу 6.20, використання функції VERSION() не потрібно застосовувати ключове слово FROM, тобто параметри таблиці непотрібні. Проте використання його не забороняється (лістинг 6.21).

Лістинг 6.21. Використання функції VERSION()

```
mysql> select version() from student;
+---------------------+
| version()           |
+---------------------+
| 5.0.77-community-nt |
| 5.0.77-community-nt |
| 5.0.77-community-nt |
| 5.0.77-community-nt |
| 5.0.77-community-nt |
| 5.0.77-community-nt |
| 5.0.77-community-nt |
| 5.0.77-community-nt |
+---------------------+
8 rows in set (0.00 sec)
```

Частіше всього функція застосовує параметри, наприклад, імена стовпців. Функція COUNT() повертає число рядків в таблиці та приймає у якості аргументу ім'я стовпчика. Функція повертає число рядків в таблиці, значення стовпчика для яких відмінні від NULL.

Лістинг 6.23. Використання функції COUNT()

```
mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)

mysql> select count(numb) from student;
+-------------+
| count(numb) |
+-------------+
|           6 |
+-------------+
1 row in set (0.00 sec)
```

У якості параметра функції поряд з іменами стовпців може виступати символ зірочка «*». Однією з особливостей використання функції є те, позначення стовпчика в результуючій таблиці співпадає з назвою функції та її параметрами. Часто це не зручно, особливо в прикладних програмах, де після виконання запиту звернення до результату виконується за іменем стовпця. В SELECT запиті стовпчика можна назначити нове ім’я. В лістингу 6.24 результату функції COUNT() присвоюється новий псевдонім total.

Лістинг 6.24. Використання оператора AS

```
mysql> select count(numb) as TOTAL from student;
+-------+
| TOTAL |
+-------+
|     6 |
+-------+
1 row in set (0.00 sec)
```

Нове ім’я можна використовувати в інших частинах SQL – запита, в виразах ORDER BY. Даний приклад часто використовується при виконанні багатотабличних запитів.

У якості додаткового прикладу можна привести виконання функцій MIN та MAX повертаючи мінімальне і максимальне значення стовпця, ім’я якого застосовується у якості прикладу (знаходження мінімального і максимального балу в стовпці mark-оцінка)

Лістинг 6.25. Використання функцій MIN та MAX

```
mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)

mysql> select MIN(mark) as minimum,
    -> MAX(mark) as maximum from student;
+---------+---------+
| minimum | maximum |
+---------+---------+
|      10 |      85 |
+---------+---------+
1 row in set (0.00 sec)
```

Проте використання функцій MIN() та MAX() у виразі WHERE призведе до помилки. У лістингу 6.26 показано намагання витягнути з таблиці **student** запис з максимальним значення каталогу mark.

Лістинг 6.26. Використання функцій MIN() та MAX() в WHERE неможливо

```
mysql> select * from student WHERE mark=max(mark);
ERROR 1111 (HY000): Invalid use of group function
mysql> _
```

Вирішення поставленої задачі слід шукати з використанням виразу ORDER BY. У лістингу 6.27 перший раз витягується запис з найменшим значенням поля, а другий - з найбільшим.

Лістинг 6.27 Використання функцій MIN() та MAX() спільно з ORDER BY

```
mysql> select * from student ORDER BY mark limit 1;
+------+-------+------+------+------+
| numb | famil | adr  | born | mark |
+------+-------+------+------+------+
| NULL | popov | NULL | NULL | NULL |
+------+-------+------+------+------+
1 row in set (0.02 sec)

mysql> select * from student ORDER BY mark desc limit 1;
+------+----------+---------+---------------------+------+
| numb | famil    | adr     | born                | mark |
+------+----------+---------+---------------------+------+
|    4 | Krymskiy | Lugansk | 1990-02-08 00:00:00 |   85 |
+------+----------+---------+---------------------+------+
1 row in set (0.00 sec)
```

СУБД MySQL має велику кількість вбудованих функцій.

Групування записів

Найкраще, коли запит повертає унікальне значення, наприклад стовпчика numb. Для цього перед іменем стовпчика можна використовувати ключове слово DISTINCT, в результаті чого відбувається виведення всіх існуючих елементів стовпця **numb**, що неповторюються

Зауваження:

Ключове слово DISTINCT має синонім – DISTINCTROW.

Лістинг 6.29. Виборка унікальних значень

```
mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)

mysql> select distinct numb from student ORDER BY numb;
+------+
| numb |
+------+
| NULL |
|    1 |
|    2 |
|    4 |
|    5 |
|    6 |
|    9 |
+------+
7 rows in set (0.02 sec)
```

Як показано в лістингу 6.29, результат запиту не містить жодного значення, що повторюється. Використання ключового слова DISTINCT допускається разом з функцією COUNT(). У лістингу 6.30 перший запит повертає загальне число записів в таблиці student в стовпці mark, відмінних від NULL, а другий – кількість унікальних (неповторюючихся) значень елементів стовпця mark.

Лістинг 6.30. Сумісне використання ключового слова DISTINCT в функції COUNT()

```
mysql> select count(mark) from student;
+-------------+
| count(mark) |
+-------------+
|           6 |
+-------------+
1 row in set (0.00 sec)

mysql> select count(distinct mark) from student;
+----------------------+
| count(distinct mark) |
+----------------------+
|                    5 |
+----------------------+
1 row in set (0.02 sec)
```

Для ключового слова DISTINCT є протилежним слово ALL, яке приписує витягування всіх значень стовпця, в тому числі і тих що повторюються. Оскільки така поведінка встановлена по замовчуванні, ключовим словом ALL часто нехтують.

Для витягання унікальних записів частіше вживають конструкції GROUP BY, за якою вказується ім'я стовпчика, по якому групується результат.

Зауваження:

Конструкція GROUP BY вживається в SELECT – запиті перед ORDER BY та LIMIT.

Лістинг 6.31. Використання конструкції GROUP BY

```
mysql> select numb from student
    -> GROUP BY numb ORDER BY numb;
+------+
| numb |
+------+
| NULL |
|    1 |
|    2 |
|    4 |
|    5 |
|    6 |
|    9 |
+------+
7 rows in set (0.00 sec)
```

Проте, відмінність від ключового слова DISTINCT, використання функції COUNT() сумісно з GROUP BY призводить не до підрахунку унікальних значень mark, а до виведення числа записів, які відповідають кожному з унікальних значень mark.

Як видно з лістингу 6.32, під номер списку №1 в стовпці numb зустрічається один раз, так як і всі інші номери списку, на відмінну від підрахунку повторювальних елементів стовпця mark.

Лістинг 6.32. Сумісне використання GROUP BY а функції COUNT()

```
mysql> select numb, count(numb) from student
    -> GROUP BY numb ORDER BY numb;
+------+-------------+
| numb | count(numb) |
+------+-------------+
| NULL |           0 |
|    1 |           1 |
|    2 |           1 |
|    4 |           1 |
|    5 |           1 |
|    6 |           1 |
|    9 |           1 |
+------+-------------+
7 rows in set (0.00 sec)

mysql> select mark, count(mark) from student
    -> GROUP BY mark ORDER BY mark;
+------+-------------+
| mark | count(mark) |
+------+-------------+
| NULL |           0 |
|   10 |           1 |
|   34 |           1 |
|   54 |           1 |
|   56 |           1 |
|   85 |           2 |
+------+-------------+
6 rows in set (0.02 sec)
```

Об’єднання таблиць

Як було сказано вище, оператор SELECT повертає результат у вигляді таблиці. Якщо формат результуючих таблиць (кількість, порядок слідування та тип стовпців) співпадає, то можливо об’єднати результати виконання двох операторів SELECT та одну результуючу таблицю. Це досягається використанням оператора UNION.

Нехай маємо дві таблиці **student** та **student2**, створені раніше, і вигляд яких наведений нижче у лістингу:

Лістинг 6.38. Одиничні SELECT – запити

```
mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)

mysql> select * from student2;
+------+---------------+----------------+---------------------+------+
| numb | famil         | adr            | born                | mark |
+------+---------------+----------------+---------------------+------+
|   55 | klichko V     | KIEV           | 1975-08-29 00:00:00 |  100 |
|   23 | YUsCHenkO     | KIEV           | 1975-08-29 00:00:00 |  100 |
|    3 | Konotopskiy L | CHYGYRYN       | 1955-11-09 00:00:00 |   91 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
+------+---------------+----------------+---------------------+------+
5 rows in set (0.00 sec)
```

Об’єднати дві таблиці можна, об’єднавши два запити SELECT з допомогою ключового слова UNION, так як це показано в лістингу 6.39.

Лістинг 6.39. Використання ключового слова UNION

```
mysql> select * from student
    -> UNION
    -> select * from student2;
+------+---------------+----------------+---------------------+------+
| numb | famil         | adr            | born                | mark |
+------+---------------+----------------+---------------------+------+
|    1 | Skrynnik      | Kirovograd     | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta        | Nova Pavlovka  | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy      | Lugansk        | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko    | Doneck         | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko    | odessa         | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov      | uman           | 1978-11-30 00:00:00 |   10 |
| NULL | popov         | NULL           | NULL                | NULL |
| NULL | levchenko     | NULL           | NULL                | NULL |
|   55 | klichko V     | KIEV           | 1975-08-29 00:00:00 |  100 |
|   23 | YUsCHenkO     | KIEV           | 1975-08-29 00:00:00 |  100 |
|    3 | Konotopskiy L | CHYGYRYN       | 1955-11-09 00:00:00 |   91 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
+------+---------------+----------------+---------------------+------+
12 rows in set (0.03 sec)
```

Перенести вміст стовпця mark таблиці student в стовпець numb таблиці student2 виконується за рахунок виконання наступних запитів, показаних в лістінгу 6.40

Лістинг 6.40. Об’єднання елементів двох стовпців різних таблиць.

```
mysql> select numb from student2
    -> union
    -> select mark from student
    -> ;
+------+
| numb |
+------+
|   55 |
|   23 |
|    3 |
|    4 |
|   34 |
|   56 |
|   85 |
|   54 |
|   10 |
| NULL |
+------+
10 rows in set (0.00 sec)
```

Контрольні питання

1. Формат застосування команди WHERE
2. Команда BETWEEN застосовується у випадку…
3. Відмінність BETWEEN і NOT BETWEEN полягає у…
4. Випадок застосування команди NOT IN…
5. Формат застосування команди ORDER BY…
6. Якщо вкінці запиту використовується DESC, то сортування відбувається за…
7. Обмеження вибірки відбувається за допомогою команди…
8. Особливість застосування функцій в MySQL полягає у …
9. Команда COUNT() застосовується для …
10. Формат застосування запитів MIN(), MAX()
11. Унікальні дані (неповторювані) виводяться за допомогою команди…
12. Для чого застосовується команда GROUP BY?
13. Об’єднання таблиць виконується командою…

ПРАКТИЧНА РОБОТА №7

Тема: Видалення та оновлення даних
Мета: Отримати та засвоїти навички редагування таблиці

Видалення даних

Час від часу виникає задача видалення записів з бази даних, наприклад, якогось студента було відраховано в результаті невдалого складання екзамену з дисципліни «БФСІД» ☺. Для реалізації даного факту потрібно застосувати видалення відповідного запису у таблиці student.

Для видалення записів з таблиць запропоновано два оператори: DELETE та TRUNCATE TABLE.

Оператор DELETE

Оператор DELETE має наступний синтаксис:

DELETE FROM tb1 WHERE where_definition ORDER BY …LIMIT rows

Оператор видаляє з таблиці tb1 записи, які задовольняють умову where_definition. В лістингу 7.1 з таблиці student видаляється вся інформація про тих студентів, котрі мають записи, мають .

Лістинг 7.1. Видалення даних з таблиці student

```
mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    1 | Skrynnik   | Kirovograd    | 1983-09-19 00:00:00 |   34 |
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
|    6 | Kuznecov   | uman          | 1978-11-30 00:00:00 |   10 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
8 rows in set (0.00 sec)
mysql> delete from student WHERE mark<50;
Query OK, 2 rows affected (0.03 sec)

mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
6 rows in set (0.00 sec)
```

Якщо в операторі DELETE відсутня умова WHERE, з таблиці видаляються всі записи.

Створимо спочатку нову таблицю inv, а потім видалемо всі записи

Лістинг 7.2. Видалення всіх записів таблиці INV

```
mysql> select * from inv;
+------+-------------+--------+
| numb | nazv        | kolich |
+------+-------------+--------+
| 2323 | стол и стул |     22 |
|  201 | кресла      |     12 |
| 5800 | монитор LG  |      5 |
+------+-------------+--------+
3 rows in set (0.00 sec)
mysql> delete from inv;
Query OK, 3 rows affected (0.02 sec)

mysql> select * from inv;
Empty set (0.00 sec)
```

Як видно з попереднього лістингу таблиця не була видалена, а очистились лише всі її рядки.

Змінна обмеження LIMIT дозволяє задати максимальну кількість записів, які можуть бути знищеними. В лістингу 7.3 видаляються записи таблиці **student2**, але не більше 3 записів.

Лістинг 7.3. Видалення перших двох записів з таблиці **student2**

```
mysql> select * from student2;
+------+--------------+----------------+---------------------+------+
| numb | famil        | adr            | born                | mark |
+------+--------------+----------------+---------------------+------+
|   55 | klichko V    | KIEV           | 1975-08-29 00:00:00 |  100 |
|   23 | YUsCHenkO    | KIEV           | 1975-08-29 00:00:00 |  100 |
|    3 | Konotopskiy L | CHYGYRYN       | 1955-11-09 00:00:00 |   91 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
+------+--------------+----------------+---------------------+------+
5 rows in set (0.00 sec)

mysql> delete from student2 limit 2;
Query OK, 2 rows affected (0.01 sec)

mysql> select * from student2;
+------+--------------+----------------+---------------------+------+
| numb | famil        | adr            | born                | mark |
+------+--------------+----------------+---------------------+------+
|    3 | Konotopskiy L | CHYGYRYN       | 1955-11-09 00:00:00 |   91 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
+------+--------------+----------------+---------------------+------+
3 rows in set (0.00 sec)
```

Так як таблиця **student2** містить 5 записів, то в результаті в таблиці залишаються три записи. Інструкція ORDER BY звичайно застосовується разом з ключовим словом LIMIT. Наприклад, якщо необхідно видалити 20 перших записів з таблиці, то виконується сортування за полем DATETIME – це гарантує, що видаляються будуть в першу чергу найстаріші таблиці.

Оператор TRUNCATE TABLE

Оператор TRUNCATE TABLE, на відміну від оператора DELETE, повністю очищує таблицю і не допускає умовного видалення. Тобто оператор TRUNCATE TABLE аналогічний оператору DELETE без умови WHERE та обмеження LIMIT. На відмінну від DELETE, видалення виконується швидше, тобто здійснюється на перебір кожного запису, а повне очищення таблиці.

Зауваження:

Оптимізатор запитів СУБД MySQL автоматично використовує оператор TRUNCATE TABLE, якщо оператор DELETE не містить умову WHERE або конструкції LIMIT.

Лістинг 7.4. Видалення всих записів з таблиці student2

```
mysql> select * from student2;
+------+---------------+----------------+---------------------+------+
| numb | famil         | adr            | born                | mark |
+------+---------------+----------------+---------------------+------+
|    3 | Konotopskiy L | CHYGYRYN       | 1955-11-09 00:00:00 |   91 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
|    4 | Deshevenko SV | Zeleni Koshary | 1982-05-11 00:00:00 |    9 |
+------+---------------+----------------+---------------------+------+
3 rows in set (0.00 sec)

mysql> truncate table student2;
Query OK, 3 rows affected (0.03 sec)
mysql> select * from student2;
Empty set (0.00 sec)
mysql> describe student2;
+-------+----------+------+-----+---------+-------+
| Field | Type     | Null | Key | Default | Extra |
+-------+----------+------+-----+---------+-------+
| numb  | int(11)  | YES  |     | NULL    |       |
| famil | text     | YES  |     | NULL    |       |
| adr   | text     | YES  |     | NULL    |       |
| born  | datetime | YES  |     | NULL    |       |
| mark  | int(11)  | YES  |     | NULL    |       |
+-------+----------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Нагадаємо, що повне видалення таблиці відбувається при наступному запиті:

Лістінг 7.5 Повне видалення таблиці student2

```
mysql> drop table student2;
Query OK, 0 rows affected (0.03 sec)

mysql> describe student2;
ERROR 1146 (42S02): Table 'kl05.student2' doesn't exist
```

Оновлення даних

Операція оновлення даних дозволяє змінювати значення полів, які вже існують в записах, це може знадобитися при зміні балів студента, при зміні адреси проживання і т.ін Для оновлення даних призначені оператори UPDATE та REPLACE. Перший дозволяє оновити окремі поля, які існують в записах, тоді як оператор REPLACE більш схожий на INSERT.

Оператор UPDATE

Оператор UPDATE має наступний синтаксис:

UPDATE [IGNORE] tb1 SET coll=expr1 [, col2=expr2 …] [WHERE where_definition] [ORDER BY …] [LIMIT rows]

В інструкції, відразу після ключового слова UPDATE, вказується таблиця tb1, яка підлягає зміні. В пропозиції SET вказується, які стовбці підлягають оновленню та встановлюються їх нові значення. Необов'язкова умова WHERE дозволяє задати критерій відбору рядків – оновленню будуть підлягати тільки ті рядки, які задовольняють умові where_definition.

В лістингу 7.5 запис у таблиці **student** в стовпці adr 'Nova Pavlovka' замінемо на вираз 'Ulyanovka'

Лістинг 7.5. Оновлення таблиці **student**

```
mysql> select * from student;
+------+------------+---------------+---------------------+------+
| numb | famil      | adr           | born                | mark |
+------+------------+---------------+---------------------+------+
|    2 | kalyta     | Nova Pavlovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk       | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck        | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa        | 1985-10-12 00:00:00 |   85 |
| NULL | popov      | NULL          | NULL                | NULL |
| NULL | levchenko  | NULL          | NULL                | NULL |
+------+------------+---------------+---------------------+------+
6 rows in set (0.00 sec)

mysql> UPDATE student SET adr='Ulyanovka'
    -> WHERE adr='Nova Pavlovka';
Query OK, 1 row affected (0.03 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from student;
+------+------------+-----------+---------------------+------+
| numb | famil      | adr       | born                | mark |
+------+------------+-----------+---------------------+------+
|    2 | kalyta     | Ulyanovka | 1982-12-19 00:00:00 |   56 |
|    4 | Krymskiy   | Lugansk   | 1990-02-08 00:00:00 |   85 |
|    9 | Ovcharenko | Doneck    | 1991-05-24 00:00:00 |   54 |
|    5 | Lavrenenko | odessa    | 1985-10-12 00:00:00 |   85 |
| NULL | popov      | NULL      | NULL                | NULL |
| NULL | levchenko  | NULL      | NULL                | NULL |
+------+------------+-----------+---------------------+------+
6 rows in set (0.00 sec)
```

Зміни можна проводити не тільки над вибірковими записами, але і над всією таблицею. Нехай викладач подвоїв в результаті якісного виконання ОДЗ бали всім студентам, що зведені до списку **student**. Зміна даних в стовпці mark виконаємо наступним чином:

Лістинг 7.6. Зміна балів студентів

```
mysql> UPDATE student SET mark=(mark*2);
Query OK, 4 rows affected (0.02 sec)
Rows matched: 6  Changed: 4  Warnings: 0

mysql> select famil,mark FROM student;
+------------+------+
| famil      | mark |
+------------+------+
| kalyta     |  112 |
| Krymskiy   |  170 |
| Ovcharenko |  108 |
| Lavrenenko |  170 |
| popov      | NULL |
| levchenko  | NULL |
+------------+------+
6 rows in set (0.00 sec)
```

Також як і у випадку оператора DELETE, конструкція LIMIT та ORDER BY дозволяють обмежити число записів, які підлягають змінам. Нехай потрібно двом першим за списком студентам зменшити оцінку на 10 балів, тоді

Лістинг 7.7. Змінені оцінки перших двох студентів

```
mysql> UPDATE student SET mark=(mark-10)
    -> ORDER BY famil LIMIT 2;
Query OK, 2 rows affected (0.02 sec)
Rows matched: 2  Changed: 2  Warnings: 0

mysql> select famil,mark FROM student;
+------------+------+
| famil      | mark |
+------------+------+
| kalyta     |  102 |
| Krymskiy   |  160 |
| Ovcharenko |  108 |
| Lavrenenko |  170 |
| popov      | NULL |
| levchenko  | NULL |
+------------+------+
6 rows in set (0.00 sec)
```

Оператор REPLACE

Оператор **REPLACE** працює як і оператор INSERT, за винятком того, що якщо минулий запис в даній таблиці має теж саме значення індексу UNIQUE або PRIMARY KEY, що і нова, то минулий запис перед занесенням нового буде знищено. Слід враховувати, що якщо не використовуються індекси UNIQUE чи PRIMARY KEY, то застосування команди **REPLACE** не має сенсу, він працює як оператор INSERT:

Синтаксис оператора RENAME аналогічний оператору INSERT:

REPLACE [INTO] tb1 [(col_name,…)] VALUES (expression,…),(…),…

В таблицю tb1 вставляють значення, які визначаються в списку після ключового слова VALUES. Задати порядок стовпців можливо за допомогою необов'язкового списку col_name, який йде після ім'я таблиці tb1. Також, як і у випадку оператора INSERT, оператор REPLASE допускає багаторядковий формат. Приклад використання оператора наведений в лістингу 7.9, де в таблицю student заносять додатково ще 3 записи, які додаються вкінець таблиці.

Лістинг 7.9. Використання оператора REPLACE

```
mysql> select * from student;
+------+------------+-----------+---------------------+------+
| numb | famil      | adr       | born                | mark |
+------+------------+-----------+---------------------+------+
|    2 | kalyta     | Ulyanovka | 1982-12-19 00:00:00 |  102 |
|    4 | Krymskiy   | Lugansk   | 1990-02-08 00:00:00 |  160 |
|    9 | Ovcharenko | Doneck    | 1991-05-24 00:00:00 |  108 |
|    5 | Lavrenenko | odessa    | 1985-10-12 00:00:00 |  170 |
| NULL | popov      | NULL      | NULL                | NULL |
| NULL | levchenko  | NULL      | NULL                | NULL |
+------+------------+-----------+---------------------+------+
6 rows in set (0.00 sec)

mysql> REPLACE into student values
    -> (Null, 'POPOV LL','Nikolaev','2002-12-02',69),
    -> (Null, 'Desevenko','Pervomaysk','1981-05-11',96);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------+------------+------------+---------------------+------+
| numb | famil      | adr        | born                | mark |
+------+------------+------------+---------------------+------+
|    2 | kalyta     | Ulyanovka  | 1982-12-19 00:00:00 |  102 |
|    4 | Krymskiy   | Lugansk    | 1990-02-08 00:00:00 |  160 |
|    9 | Ovcharenko | Doneck     | 1991-05-24 00:00:00 |  108 |
|    5 | Lavrenenko | odessa     | 1985-10-12 00:00:00 |  170 |
| NULL | popov      | NULL       | NULL                | NULL |
| NULL | levchenko  | NULL       | NULL                | NULL |
| NULL | POPOV LL   | Nikolaev   | 2002-12-02 00:00:00 |   69 |
| NULL | Desevenko  | Pervomaysk | 1981-05-11 00:00:00 |   96 |
+------+------------+------------+---------------------+------+
8 rows in set (0.02 sec)
```

Контрольні питання

1. Відмінність оператора DELETE і TRUNCATE…
2. Формат застосування команди DELETE з LIMIT…
3. Якими операторами виконується оновлення даних таблиці?
4. Чим оператор REPLACE подібний до INSERT?

ПРАКТИЧНА РОБОТА №8

Тема: Редагування структури таблиці
Мета: Отримати навички редагування та форматування таблиці

Після того, як таблиці бази даних створенні, заповнені та введені в експлуатацію, може виникнути питання про зміну структури, з метою оптимізації роботи бази даних та додатків які вони використовують.

Найбільш розповсюджені завдання для зміни структури таблиці включають зміну порядку проходження стовпців, їх назв, типів, та додавання нових індексів. Всі ці операції виконуються за допомогою оператора **ALTER TABLE**.

Оператор ALTER TABLE має такий синтаксис:

ALTER TABLE tb1 alter_specification [, alter_specification]…

Відразу після оператора ALTER TABLE потрібно ввести ім'я таблиці tb1, яка підлягає зміні, що виконується змінами alter_specification, яких може бути декілька. Більш детальний опис оператора виконується далі.

Зауваження:
Оператор ALTER TABLE під час роботи створює тимчасову копію вихідної таблиці. Потрібні зміни виконуються на копії, потім вихідна таблиця видаляється, а новою замінюється її значення. Це потрібно робити для того, щоб у нову таблицю автоматично потрапляли всі оновлення, окрім невдалих. Під час виконання оператора ALTER TABLE вихідна таблиця доступна для читання іншими.

Додавання та видалення стовпців

Для додавання стовпця використовується форма оператора ALTER TABLE – ADD [COLUMN], який запропонований в лістингу 8.1, де в таблицю **student** KL05 вводиться новий стовпець **propuski**, куди заноситься кількість пропущених навчальних годин.

Лістинг 8.1. Додавання нового стовпця propuski в таблицю student

```
mysql> alter table student ADD COLUMN propuski int(3);
Query OK, 8 rows affected (0.14 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> describe student;
+----------+----------+------+-----+---------+-------+
| Field    | Type     | Null | Key | Default | Extra |
+----------+----------+------+-----+---------+-------+
| numb     | int(11)  | YES  |     | NULL    |       |
| famil    | text     | YES  |     | NULL    |       |
| adr      | text     | YES  |     | NULL    |       |
| born     | datetime | YES  |     | NULL    |       |
| mark     | int(11)  | YES  |     | NULL    |       |
| propuski | int(3)   | YES  |     | NULL    |       |
+----------+----------+------+-----+---------+-------+
6 rows in set (0.02 sec)

mysql> select * from student;
+------+------------+------------+---------------------+------+----------+
| numb | famil      | adr        | born                | mark | propuski |
+------+------------+------------+---------------------+------+----------+
|    2 | kalyta     | Ulyanovka  | 1982-12-19 00:00:00 |  102 |     NULL |
|    4 | Krymskiy   | Lugansk    | 1990-02-08 00:00:00 |  160 |     NULL |
|    9 | Ovcharenko | Doneck     | 1991-05-24 00:00:00 |  108 |     NULL |
|    5 | Lavrenenko | odessa     | 1985-10-12 00:00:00 |  170 |     NULL |
| NULL | popov      | NULL       | NULL                | NULL |     NULL |
| NULL | levchenko  | NULL       | NULL                | NULL |     NULL |
| NULL | POPOV LL   | Nikolaev   | 2002-12-02 00:00:00 |   69 |     NULL |
| NULL | Desevenko  | Pervomaysk | 1981-05-11 00:00:00 |   96 |     NULL |
+------+------------+------------+---------------------+------+----------+
8 rows in set (0.00 sec)
```

Видалення стовпця виконується за допомогою оператора ALTER TABLE – DROP [COLUMN]. В лістингу 8.2 демонструється видалення новоствореного стовпця **propuski**.

Лістинг 8.2. Видалення стовпця propuski з таблиці student

```
mysql> alter table student drop column propuski;
Query OK, 8 rows affected (0.14 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> select * from student;
+------+------------+------------+---------------------+------+
| numb | famil      | adr        | born                | mark |
+------+------------+------------+---------------------+------+
|    2 | kalyta     | Ulyanovka  | 1982-12-19 00:00:00 |  102 |
|    4 | Krymskiy   | Lugansk    | 1990-02-08 00:00:00 |  160 |
|    9 | Ovcharenko | Doneck     | 1991-05-24 00:00:00 |  108 |
|    5 | Lavrenenko | odessa     | 1985-10-12 00:00:00 |  170 |
| NULL | popov      | NULL       | NULL                | NULL |
| NULL | levchenko  | NULL       | NULL                | NULL |
| NULL | POPOV LL   | Nikolaev   | 2002-12-02 00:00:00 |   69 |
| NULL | Desevenko  | Pervomaysk | 1981-05-11 00:00:00 |   96 |
+------+------------+------------+---------------------+------+
8 rows in set (0.00 sec)
```

Зауваження:

Ключове слово COLUMN є необов’язковим і може не вживатися, як при додаванні так і при видаленні стовпця.

Зауваження:

Якщо таблиця містить тільки один стовпець, то цей стовпець не може бути видаленим. Замість цього можна видалити дану таблицю, використовуючи команду DROP TABLE.

Як видно з лістингу 8.1, новий стовпець propuski був доданий у кінець таблиці. Для того, щоб змінити позицію, в яку буде поміщений стовпець, разом з ALTER TABLE – ADD [COLUMN] використовуються ключові слова **FIRST** та **AFTER**. Ключове слово FIRST потребує, щоб новий стовпець був розміщений першим, а ALTER дозволяє вказати, після якого стовпця слід помістити новий. В лістингу 8.3 демонструється додавання стовпця **Ak_zaborg** (академічна заборгованність) на початок таблиці та стовпця **propuski** – після стовпця born.

Лістинг 8.3. Додавання стовпців

```
mysql> alter table student ADD Ak_zaborg int first,
    -> ADD propuski int AFTER born;
mysql> describe student;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| Ak_zaborg | int(11)  | YES  |     | NULL    |       |
| numb      | int(11)  | YES  |     | NULL    |       |
| famil     | text     | YES  |     | NULL    |       |
| adr       | text     | YES  |     | NULL    |       |
| born      | datetime | YES  |     | NULL    |       |
| propuski  | int(11)  | YES  |     | NULL    |       |
| mark      | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
7 rows in set (0.02 sec)
```

Зміна вже існуючих стовпців

Окрім вводу в таблицю нових стовпців, часто постає задача зміни типу або назви вже існуючого стовпця. Для зміни типу вже існуючого стовпця використовується ключове слово **MODIFY**, після якого вказується ім’я модифікованого стовпця та його новий тип. В лістингу 8.4 цілочисленний тип стовпця Ak_zaborg, добавленого в лістингу 8.3, змінюється на текстовий.

Лістинг 8.4. Зміна типу стовпця Ak_zaborg

```
mysql> alter table student MODIFY Ak_zaborg TEXT;
Query OK, 8 rows affected (0.14 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> describe student;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| Ak_zaborg | text     | YES  |     | NULL    |       |
| numb      | int(11)  | YES  |     | NULL    |       |
| famil     | text     | YES  |     | NULL    |       |
| adr       | text     | YES  |     | NULL    |       |
| born      | datetime | YES  |     | NULL    |       |
| propuski  | int(11)  | YES  |     | NULL    |       |
| mark      | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Порядок слідування стовпців змінюється використанні ключових слів FIRST та ALTER дозволяє змінити порядок слідування стовпців.

Лістинг 8.5. Зміна порядку слідування стовпців

```
mysql> alter table student MODIFY famil TEXT FIRST;
Query OK, 8 rows affected (0.16 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> alter table student MODIFY mark INT AFTER famil;
Query OK, 8 rows affected (0.14 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> alter table student MODIFY numb INT FIRST;
Query OK, 8 rows affected (0.13 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> describe student;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| numb      | int(11)  | YES  |     | NULL    |       |
| famil     | text     | YES  |     | NULL    |       |
| mark      | int(11)  | YES  |     | NULL    |       |
| Ak_zaborg | text     | YES  |     | NULL    |       |
| adr       | text     | YES  |     | NULL    |       |
| born      | datetime | YES  |     | NULL    |       |
| propuski  | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Ключове слово MODIFY в операторі ALTER TABLE не дозволяє змінити ім'я стовпця, для цього призначене ключове слово **CHANGE**. Після даного ключового слова вказується ім'я стовпця, потім слідує нове ім'я та новий тип. В лістингу 8.6 стовпець **mark** змінює своє ім'я на **OCINKI**.

Лістинг 8.6. Перейменування стовпця **mark** на **OCINKI**

```
mysql> alter table student CHANGE mark OCINKI INT;
Query OK, 8 rows affected (0.13 sec)
Records: 8  Duplicates: 0  Warnings: 0

mysql> describe student;
+-----------+----------+------+-----+---------+-------+
| Field     | Type     | Null | Key | Default | Extra |
+-----------+----------+------+-----+---------+-------+
| numb      | int(11)  | YES  |     | NULL    |       |
| famil     | text     | YES  |     | NULL    |       |
| OCINKI    | int(11)  | YES  |     | NULL    |       |
| Ak_zaborg | text     | YES  |     | NULL    |       |
| adr       | text     | YES  |     | NULL    |       |
| born      | datetime | YES  |     | NULL    |       |
| propuski  | int(11)  | YES  |     | NULL    |       |
+-----------+----------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

Перетворення таблиці

Окрім зміни структури таблиці, оператор ALTER TABLE дозволяє змінювати параметри таблиці, наприклад, її назву. Для цього використовується конструкція RENAME [TO] new_table, в результаті застосування чого таблиця отримує нове ім'я new_table. В лістингу 8.10 таблиця student бази даних KL05 змінює своє ім'я на STUDENT_kl05.

Зауваження:

Ключове слово TO в операції RENAME [TO] може не вживатися.

Лістинг 8.10. Перейменування таблиці student на STUDENT_kl05.

```
mysql> show tables;
+----------------+
| Tables_in_kl05 |
+----------------+
| inv            |
| student        |
+----------------+
2 rows in set (0.02 sec)

mysql> alter table student RENAME STUDENT_kl05;
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+----------------+
| Tables_in_kl05 |
+----------------+
| inv            |
| student_kl05   |
+----------------+
2 rows in set (0.00 sec)
```

Для перейменування таблиць існує окремий оператор RENAME TABLE, яка має наступний синтаксис:

RENAME TABLE tb1_name TO new_tb1_name [, tb1_name2 TO new_tb1_name2,…]

Результатом роботи оператора є перейменування таблиці tb1_name в new_tb1_name. В одному операторі можливо перейменувати відразу декілька таблиць.

Лістинг 8.11. Зміна імені двох таблиць

```
mysql> show tables;
+----------------+
| Tables_in_kl05 |
+----------------+
| inv            |
| student_kl05   |
+----------------+
2 rows in set (0.00 sec)

mysql> RENAME TABLES inv TO inv_nomer,student_kl05 TO student;
Query OK, 0 rows affected (0.06 sec)

mysql> show tables;
+----------------+
| Tables_in_kl05 |
+----------------+
| inv_nomer      |
| student        |
+----------------+
2 rows in set (0.00 sec)
```

Контрольні питання

1. Редагування структури таблиці відбувається за рахунок використання оператора…
2. Додавання і видалення стовпців відбувається у форматі…
3. Який формат застосування команди ADD COLUMN при додаванні стовпця вкінці/на початку таблиці?
4. Зміна порядку слідування вже існуючих стовпців таблиці проводиться за рахунок оператора…
5. Перейменування стовпця виконується запитом…
6. Перейменувати таблицю можна, використавши оператор …
7. Формат застосування команди RENAME при зміні імен декількох таблиць одночасно…
8. Відмінність операторів CHANGE і MODIFY

ПРАКТИЧНАРОБОТА №9

Тема: Оператори та математичні функції

Мета: Отримати навички виконання математичних дій в СУБД MySQL

В даній лабораторній роботі будуть розглянуті основні функції для роботи з числовими даними. Також описані оператори, які можна використовувати при постановці запитів MySQL.

Оператори

Під операторами мають на увазі конструкції мови, які виконують перетворення даних, наприклад, операція додавання «+», віднімання «-» і т.д. Дані над якими здійснюється операція, називають *операндами*.

Арифметичні оператори

До арифметичних операторів відносять додавання «+», віднімання «-», множення «*», ділення «/» та цілочисельне ділення DIV. Якщо у якості аргументів використовуються числа INT, то результатом є число типу BIGINT. В той же час, якщо аргументи мають тип UNSIGNED INT, то результат буде також мати тип UNSIGNED INT.

Зауваження:

Операція цілочисельного ділення DIV введена в СУБД починаючи з версії 4.1.0.

Застосування операції додавання «+» демонструється в лістингу 9.1.

Лістинг 9.1. Операція додавання

```
mysql> select 22+3;
+------+
| 22+3 |
+------+
|   25 |
+------+
1 row in set (0.00 sec)
```

Важливо відмітити, що операція додавання числа з NULL дає NULL.

Лістинг 9.3. Операція додавання з NULL

```
mysql> select 3+null;
+--------+
| 3+null |
+--------+
|   NULL |
+--------+
1 row in set (0.00 sec)
```

Така поведінка в певній мірі вірна. NULL позначає дані, для яких значення невідоме, тому додавання до такого значення числа призводить до невідомого значення, тобто до NULL.

Лістинг 9.6. Операція віднімання

```
mysql> select 22-3;
+------+
| 22-3 |
+------+
|   19 |
+------+
1 row in set (0.01 sec)
```

Окрім бінарного оператора «-», який виконує віднімання, існує унарний оператор, який змінює знак операнда

На відміну від інших мов програмування, ділення на нуль не викликає помилки синтаксису та зупинки обчислень. У якості результату повертається NULL.

Лістинг 9.10. Ділення на нуль

```
mysql> select 889/0;
+-------+
| 889/0 |
+-------+
|  NULL |
+-------+
1 row in set (0.00 sec)
```

Крім звичайного ділення, починаючи з версії 4.1.0 введений оператор цілочисельного ділення DIV, робота якого запропонована в лістингу 9.11.

Лістинг 9.11. Цілочисельне ділення за допомогою DIV

```
mysql> select 133 DIV 2,133/2;
+-----------+---------+
| 133 DIV 2 | 133/2   |
+-----------+---------+
|        66 | 66.5000 |
+-----------+---------+
1 row in set (0.00 sec)
```

Як видно з лістинга 9.11, результат ділення за допомогою DIV є цілочисельним. Дробова частина відкидається та округлення результату не виконується. Для того, щоб отримати залишок від ділення, необхідно скористатися оператором «%».

Оператори порівняння

Ця група операторів використовується для створення умовних запитів із застосуванням конструкції WHERE. Результатом операції порівняння може бути 0 (хибне), та 1 (істина) та NULL. Порівняти можливо як числа, так і рядки.

Зауваження:

При порівнянні із значенням NULL результат завжди дорівнює NULL. Винятком є оператор «<=>», який спеціально введений для порівняння NULL.

Зауваження:

Якщо один з операндів має тип TIMESTAMP або DATETIME, а інший є константою, операнди порівнюються як TIMESTAMP – типи.

Рівність =

Результат рівності дорівнює 1, якщо операнди рівні, чи дорівнює 0 в протилежному випадку. В лістингу 9.14. запропоноване порівняння рядків та чисел.

Лістинг 9.14. Порівняння рядків та чисел

```
mysql> select 2=1,1=55,'text'='test','kl05'='KL05';
+-----+------+---------------+---------------+
| 2=1 | 1=55 | 'text'='test' | 'kl05'='KL05' |
+-----+------+---------------+---------------+
|   0 |    0 |             0 |             1 |
+-----+------+---------------+---------------+
1 row in set (0.00 sec)
```

Як видно з лістинга 9.14, при порівнянні рядків з числами останні автоматично перетворюються в числа, якщо таке перетворення виконати неможливо, то рядок буде мати значення 0

Порівняння рядків виконується без врахування регістру. Якщо необхідно, щоб регістр враховувався, рядки слід порівняти як бінарні (порівняння виконуються не за символами, а за байтами). Для цього рядок перевіряється словом BINARY.

Лістинг 9.15. Бінарне порівняння рядків

```
mysql> select BINARY 'KL05'='kl05';
+----------------------+
| BINARY 'KL05'='kl05' |
+----------------------+
|                    0 |
+----------------------+
1 row in set (0.00 sec)
```

Порівняти можна не лише цілочисельні значення і рядки, а й рядки з плаваючою комою.

Лістинг 9.16. Порівняння чисел з плаваючою комою

```
mysql> select 0.1=0.01,0.1=.1;
+----------+--------+
| 0.1=0.01 | 0.1=.1 |
+----------+--------+
|        0 |      1 |
+----------+--------+
1 row in set (0.00 sec)
```

Як видно з лістингу 9.16. при написанні чисел з плаваючою комою нуль можна не вказувати.

Оператор <=>

Оператор еквівалентності <=>, подібний до оператора рівності «=», але повертає він лише два значення: 1 (істина) та 0 (хибне). Результат NULL він не повертає.

Лістинг 9.13. Порівняння з використанням оператора еквівалентності <=>

```
mysql> select 1<=>1, 1<=>NULL, null<=>null;
+-------+----------+-------------+
| 1<=>1 | 1<=>NULL | null<=>null |
+-------+----------+-------------+
|     1 |        0 |           1 |
+-------+----------+-------------+
1 row in set (0.02 sec)
```

Як видно з лістингу 9.17, порівняння NULL з NULL дає 1(істину), на відміну від оператора рівності «=», який дав би в цьому випадку NULL.

Оператор < >

Оператор нерівності «< >» дорівнює 1(істина), якщо операнди не рівні, та 0 (хиба) в протилежному випадку.

Зауваження:

Окрім запису < > , даний оператор може приймати форму «!=».

Лістинг 9.18. Використання оператора нерівності «< >»

```
mysql> select 0.01<>.01, 2.23<>3.22, binary 'YURA'!='yura';
+-----------+------------+-----------------------+
| 0.01<>.01 | 2.23<>3.22 | binary 'YURA'!='yura' |
+-----------+------------+-----------------------+
|         0 |          1 |                     1 |
+-----------+------------+-----------------------+
1 row in set (0.00 sec)
```

Даний оператор повертає значення, яке протилежне оператору рівності «=».

Оператор <

Оператор «меньше» «<» повертає 1 (істина), якщо лівий операнд менше правого, в противному випадку 0 (хиба).

Лістинг 9.19. Використання оператора «<»

```
mysql> select 11<2, 56<89;
+------+-------+
| 11<2 | 56<89 |
+------+-------+
|    0 |     1 |
+------+-------+
1 row in set (0.00 sec)
```

Оператор <=

Оператор «менше чи дорівнює» «<=» повертає 1(істина), якщо лівих операнд менше правого чи обидва оператори рівні, інакше повертається 0 (хиба).

Лістинг 9.21. Використання оператора «<=»

```
mysql> select 45.5<=36.7,666<=666;
+------------+----------+
| 45.5<=36.7 | 666<=666 |
+------------+----------+
|          0 |        1 |
+------------+----------+
1 row in set (0.00 sec)
```

Функція GREATEST

Ця функція повертає максимальне значення зі списку.

Лістинг 9.33. Використання функції GREATEST()

```
mysql> select GREATEST(-55, 0, 158);
+----------------------+
| GREATEST(-55, 0, 158) |
+----------------------+
|                  158 |
+----------------------+
1 row in set (0.00 sec)
```

Функція LEAST

Функція LEAST() є протилежною до GREATEST() та повертає мінімальне значення зі списку.

Лістинг 9.34. Використання функції LEAST()

```
mysql> select LEAST(-55, 0, 158);
+--------------------+
| LEAST(-55, 0, 158) |
+--------------------+
|                -55 |
+--------------------+
1 row in set (0.00 sec)
```

Функція INTERVAL

Функція INTERVAL (N,N1,N2,N3,…) повертає 0, якщо N < N1, та 1, якщо N < N2,… Всі аргументи як цілі числа (лістинг 9.40).

Лістинг 9.40. Використання функції INTERVAL()

```
mysql> select interval(23,1,15,17,30,44,200);
+---------------------------------+
| interval(23,1,15,17,30,44,200) |
+---------------------------------+
|                               3 |
+---------------------------------+
1 row in set (0.00 sec)
```

Математичні функції

Усі числові функції повертають NULL, якщо при їх роботі виникає помилка, наприклад, якщо аргументи виходять за межі допустимих значень або в якості аргументів передається NULL.

MySQL, як і будь-яка база даних, володіє великою кількістю вбудованих функцій, які будуть розглядатися далі. Функції характеризуються ім’ям та аргументами, які перераховуються через кому після імені та в круглих дужках. Якщо аргументи та функції відсутні, круглі дужки всеодно слід вказати, наприклад, NOW(). Відмінною рисою MySQL є те, що при використанні функції пробіли між ім’ям та круглими скобками неможливі, тобто написання функції NOW() вірне, а NOW () вже ні. Результат функції підставляється замість виклику функції.

Функція ABS

Функція ABS (X) повертає абсолютне значення аргументу X

Лістинг 9.54. Використання функції ABS()

```
mysql> select ABS(-10.122), abs(43.2), Abs(null);
+--------------+-----------+-----------+
| ABS(-10.122) | abs(43.2) | Abs(null) |
+--------------+-----------+-----------+
|       10.122 |      43.2 |      NULL |
+--------------+-----------+-----------+
1 row in set (0.02 sec)
```

Функція ACOS

Функція ACOS(X) повертає арккосинус числа X або NULL, якщо значення X не знаходиться в діапазоні від -1 до 1

Лістинг 9.55. Використання функції ACOS()

```
mysql> select acos(-1),acos(0),acos(0.5);
+-----------------+-----------------+-----------------+
| acos(-1)        | acos(0)         | acos(0.5)       |
+-----------------+-----------------+-----------------+
| 3.1415926535898 | 1.5707963267949 | 1.0471975511966 |
+-----------------+-----------------+-----------------+
1 row in set (0.00 sec)
```

Функція ASIN

Функція ASIN(X) повертає арксинус числа X або NULL, якщо значення X не знаходиться в діапазоні від -1 до 1

Лістинг 9.56. Використання функції ASIN()

```
mysql> select asin(-1),asin(0),asin(0.5);
+------------------+---------+-----------------+
| asin(-1)         | asin(0) | asin(0.5)       |
+------------------+---------+-----------------+
| -1.5707963267949 |       0 | 0.5235987755983 |
+------------------+---------+-----------------+
1 row in set (0.00 sec)
```

Функція ATAN

Функція ATAN(X) повертає арктангенс числа X

Лістинг 9.57. Використання функції ATAN()

```
mysql> select atan(-1),atan(0),atan(null);
+-------------------+---------+------------+
| atan(-1)          | atan(0) | atan(null) |
+-------------------+---------+------------+
| -0.78539816339745 |       0 |       NULL |
+-------------------+---------+------------+
1 row in set (0.00 sec)
```

Функція COS

Функція COS(X) обчислює косинус кута X, який задано в радіанах

Лістинг 9.61. Використання функції COS()

```
mysql> select cos(0),cos(1), cos(0.5);
+--------+------------------+------------------+
| cos(0) | cos(1)           | cos(0.5)         |
+--------+------------------+------------------+
|      1 | 0.54030230586814 | 0.87758256189037 |
+--------+------------------+------------------+
1 row in set (0.00 sec)
```

Функція COT

Функція COT(X) обчислює котангенс кута X, який задано в радіанах

Лістинг 9.62. Використання функції COT()

```
mysql> select cot(0),cot(2),cot(0.3);
+--------+-------------------+-----------------+
| cot(0) | cot(2)            | cot(0.3)        |
+--------+-------------------+-----------------+
|   NULL | -0.45765755436029 | 3.2327281437658 |
+--------+-------------------+-----------------+
1 row in set (0.01 sec)
```

Функція DEGREES

Функція DEGREES(X) повертає значення кута X, перетвореного з радіан в градуси

Лістинг 9.64. Використання функції DEGREES()

```
mysql> select DEGREES(pi()), degrees(pi()/2),degrees(pi()/3);
+---------------+-----------------+-----------------+
| DEGREES(pi()) | degrees(pi()/2) | degrees(pi()/3) |
+---------------+-----------------+-----------------+
|           180 |              90 |              60 |
+---------------+-----------------+-----------------+
1 row in set (0.00 sec)
```

Функція EXP

Функція EXP(X) повертає значення степеня числа X: e^x, де e – основа натурального логарифму

Лістинг 9.65. Використання функції EXP()

```
mysql> select exp(0),exp(1),exp(null);
+--------+----------------+-----------+
| exp(0) | exp(1)         | exp(null) |
+--------+----------------+-----------+
|      1 | 2.718281828459 |      NULL |
+--------+----------------+-----------+
1 row in set (0.00 sec)
```

Функція LOG

Функція LOG(X) повертає натуральний логарифм (з основою e) числа X.

Зауваження:

Крім LOG(X) допускається запис LN(X)

Лістинг 9.67. Використання функції LOG(X)

```
mysql> select LOG(1), LOg(0),ln(exp(1));
+--------+--------+------------+
| LOG(1) | LOg(0) | ln(exp(1)) |
+--------+--------+------------+
|      0 |   NULL |          1 |
+--------+--------+------------+
1 row in set (0.00 sec)
```

Крім формату з одним аргументом, показано в лістингу 9.67, функція допускає передачу їй двох аргументів - LOG(B,X), в цьому випадку функція повертає логарифм числа X з основою B.

Лістинг 9.68. Використання функції LOG(B,X)

```
mysql> select LOG(2,8), log(10,100), log(3,27);
+----------+-------------+-----------+
| LOG(2,8) | log(10,100) | log(3,27) |
+----------+-------------+-----------+
|        3 |           2 |         3 |
+----------+-------------+-----------+
1 row in set (0.00 sec)
```

Функція LOG2

Функція LOG2(X), приклад якої запропоновано в лістингу 9.69, повертає логарифм числа X з основою 2.

Лістинг 9.69. Використання функції LOG2()

```
mysql> select LOG2(16), log2(124);
+----------+-----------------+
| LOG2(16) | log2(124)       |
+----------+-----------------+
|        4 | 6.9541963103869 |
+----------+-----------------+
1 row in set (0.00 sec)
```

Функція LOG10

Функція LOG10(X) повертає логарифм числа X з основою 10.

Лістинг 9.70. Використання функції LOG10

```
mysql> select log10(0.01), log10(1000);
+-------------+-------------+
| log10(0.01) | log10(1000) |
+-------------+-------------+
|          -2 |           3 |
+-------------+-------------+
1 row in set (0.00 sec)
```

Функція PI

Функція PI() не приймає аргументів та повертає значення числа π.

Лістинг 9.71. Використання функції PI()

```
mysql> select pi(), pi()/2, pi()/4;
+----------+--------------+--------------+
| pi()     | pi()/2       | pi()/4       |
+----------+--------------+--------------+
| 3.141593 | 1.5707963268 | 0.7853981634 |
+----------+--------------+--------------+
1 row in set (0.00 sec)
```

Функція POW

Функція POW(X, Y) повертає значення числа X, яке піднесене до степеня Y - X^Y.

Зауваження:

Для функції POW() існує синонім POWER()

Лістинг 9.72. Використання функції POW()

```
mysql> select POW(2,3), POW(10,0), POW(pi(),1);
+----------+-----------+-----------------+
| POW(2,3) | POW(10,0) | POW(pi(),1)     |
+----------+-----------+-----------------+
|        8 |         1 | 3.1415926535898 |
+----------+-----------+-----------------+
1 row in set (0.00 sec)
```

Функція RADIANS

Функція RADIANS(X) повертає значення кута X, перетворення з градусів в радіани.

Лістинг 9.73. Використання функції RADIANS()

```
mysql> select RADIANS(0), raDIANs(180), radiaNS(60);
+------------+-----------------+-----------------+
| RADIANS(0) | raDIANs(180)    | radiaNS(60)     |
+------------+-----------------+-----------------+
|          0 | 3.1415926535898 | 1.0471975511966 |
+------------+-----------------+-----------------+
1 row in set (0.00 sec)
```

Функція RAND

Функція RAND() повертає випадкове значення з плаваючою крапкою в діапазоні від 0.0 до 1.0. Ця функція може приймати у якості аргумента ціле число N, яким ініціалізується генератор випадкових чисел.

Кожен раз, коли функція приймає такий самий параметр N, вона повертає такий самий результат. Звичайно функція перший раз викликається з параметром, а всі наступні рази без параметрів. Приклад застосування функції RAND() запропоновано в лістингу 9.74.

Лістинг 9.74. Використання функції RAND()

```
mysql> select RAND(2), rand(689);
+------------------+------------------+
| RAND(2)          | rand(689)        |
+------------------+------------------+
| 0.65558664654902 | 0.53138266273903 |
+------------------+------------------+
1 row in set (0.00 sec)
```

Завдання початкового значення для всіх користувачів виконується незалежно один від одного та в наступному сеансі при такому ж ініціалізуючому значенні буде отримано інший результат.

Функцію RAND() можливо використовувати після ключового слова ORDER BY для вибірки записів у випадковому порядку. В якості прикладу виберемо з таблиці **student** бази даних KL05 стовпця **famil** 5 випадкових прізвищ.

Лістинг 9.75. Випадкова вибірка з таблиці з використанням функції RAND()

```
mysql> use KL05;
Database changed
mysql> select famil from student order by RAND() limit 5;
+-----------+
| famil     |
+-----------+
| popov     |
| kalyta    |
| POPOV LL  |
| Desevenko |
| levchenko |
+-----------+
5 rows in set (0.08 sec)
```

Найчастіше потрібно здійснювати вибірку одного випадкового запису, для цього достатньо вказати після ключового слова LIMIT цифру 1.

Лістинг 9.76. Вибірка одного випадкового запису

```
mysql> select famil from student order by RAND() limit 1;
+-----------+
| famil     |
+-----------+
| levchenko |
+-----------+
1 row in set (0.00 sec)
```

Функція ROUND

Функція ROUND(X) повертає закруглене до найближчого цілого значення числа X. Ця функція може приймати два аргументи ROUND(X, D), в цьому випадку округлення виконується до числа, яке має D знаків після коми.

Лістинг 9.77. Використання функцій ROUND()

```
mysql> select ROUND(1.026589,3), round(-.256,1);
+-------------------+----------------+
| ROUND(1.026589,3) | round(-.256,1) |
+-------------------+----------------+
|             1.027 |           -0.3 |
+-------------------+----------------+
1 row in set (0.00 sec)
```

Функція SIGN

Функція SIGN(X) дозволяє визначити знак числа X та повертає -1, 0 або 1, якщо X від'ємне, дорівнює 0 або додатне відповідно.

Лістинг 9.79. Використання функції SIGN()

```
mysql> select sign(-5.012), sign(0), sign(.022);
+--------------+---------+------------+
| sign(-5.012) | sign(0) | sign(.022) |
+--------------+---------+------------+
|           -1 |       0 |          1 |
+--------------+---------+------------+
1 row in set (0.00 sec)
```

Функція SQRT

Функція SQRT(X) обчислює квадратний корінь числа X.

Лістинг 9.81. Використання функцій SQRT()

```
mysql> select sqrt(144), SQRT(9), sqrt(0.01);
+-----------+---------+------------+
| sqrt(144) | SQRT(9) | sqrt(0.01) |
+-----------+---------+------------+
|        12 |       3 |        0.1 |
+-----------+---------+------------+
1 row in set (0.00 sec)
```

Функція TRUNCATE

Функція TRUNCATE(X, D) повертає число X з дробовою частиною, яка має D знаків після коми. Якщо кількість знаків після коми в числі X більше D, то зайві рядки зменшуються, якщо менше, то в кінець числа додаються нулі.

Лістинг 9.83. Використання функції TRUNCATE()

```
mysql> select truncate(-28.1201,2), TRUNCATE(.011,0), truncate(87.565,1);
+----------------------+-----------------+--------------------+
| truncate(-28.1201,2) | TRUNCATE(.011,0) | truncate(87.565,1) |
+----------------------+-----------------+--------------------+
|               -28.12 |               0 |               87.5 |
+----------------------+-----------------+--------------------+
1 row in set (0.00 sec)
```

Контрольні питання

1. Основні математичні оператори в СУБД MySQL
2. Загальний формат застосування математичних операторів
3. При діленні числа на «0» MySQL видає…
4. Чи можна в MySQL одночасно вирішувати більше одного математичного виразу?
5. Оператор DIV застосовується для …
6. Відмінність застосування операторів «/» і DIV …
7. Щоб порівняти вирази «ГРУПА» і «група» застосовують оператор…
8. Чи сприйме MySQL правильним запис числа 0,245 у вигляді ,245?
9. Що повинен видати MySQL при виконанні запиту SELECT 333.3<=322.2?
10. Функція GREATEST застосовується для …
11. Антонім попередньої функції є оператор…
12. Для знаходження модуля числа потрібно використати оператор …
13. Який формат застосування тригонометричних функцій?
14. За допомогою якої функції відбувається перетворення радіан в градуси і навпаки?
15. Щоб визначити X^y використовуємо …
16. Суть функції RAND…
17. Функція SQRT визначає…
18. Оператор TRUNCATE застосовується у випадку…

ПРАКТИЧНА РОБОТА №10

Тема: Функції дати та часу

Мета: Отримати та закріпити навички виконання операцій з даними календарного типу

Ця група функцій призначена для роботи з даними часового типу: TIME, DATE, TIMESTAMP, TIME та YEAR.

Зауваження:

Відмінною рисою MySQL є те, що при використанні функцій пробіли між іменем функції та круглими дужками ставити недопустимо, тобто написання NOW() вірне, а NOW () невірне. Результат функції підставляється замість виклику функції.

Функції дати та часу можуть приймати дані в різноманітних форматах:

- у вигляді рядка, який містить повну дату та час ‘2005-12-27 08:30:00’;
- рядки, які містять тільки дату ‘2005-12-27’;
- у вигляді числа 20051227083000 або 2005 1227.

MySQL коректно розпізнає час у всіх форматах та переводить до внутрішнього представлення.

Функції дозволяють повернути поточну дату та час доби. До них відносяться CURDATE(), CURTIME(), UTC_DATE(), UTC_TIME(), UTC_TIMESTAMP(). Обчислення поточного часу в рамках одного SQL запиту виконується тільки один раз, як би вони не викликалося протягом даного запиту. Це призведе до того, що час в межах всього запиту залишається постійним.

Функція ADDDATE

Функція ADDDATE(date, INTERVAL expr type) повертає час date, за яким доданий часовий інтервал, який визначається другим параметром.

Зауваження:

Функція ADDDATE() має синонім DATE_ADD().

Другий параметр починається з ключового слова INTERVAL, за яким слідує число одиниці виміру type. Значення, які може приймати параметр type, які запропоновані в першому стовпці таблиці 10.1. Функція ADDDATE() допускає також і від’ємні значення. В цьому випадку виконується віднімання часового інтервалу.

Лістинг 10.1. Використання функції ADDDATE()

```
mysql> select ADDDATE('2010-01-01',INTERVAL 30 day);
+---------------------------------------+
| ADDDATE('2010-01-01',INTERVAL 30 day) |
+---------------------------------------+
| 2010-01-31                            |
+---------------------------------------+
1 row in set (0.03 sec)

mysql> select ADDDATE('2010-01-01',INTERVAL -30 day);
+----------------------------------------+
| ADDDATE('2010-01-01',INTERVAL -30 day) |
+----------------------------------------+
| 2009-12-02                             |
+----------------------------------------+
1 row in set (0.00 sec)
```

В лістингу 10.1 запропонований приклад використання функції ADDDATE(). До дати 01 січня 2010 року додається 30 днів в одному випадку, а в іншому – віднімається 30 днів. Отримана дата виводиться в результаті виконання введених команд.

Зупинимося докладніше на таблиці 10.1. Замість одиниць виміру DAY, які використовувалися в лістингах 10.1 та 10.2, можна застосувати будь-який параметр з першого стовпця таблиці. Вираз expr, можна задавати як у вигляді числа, так і у вигляді рядка, expr містить також нечислові значення (пробіл, дефіс, двокрапка). Замість пробілу, двокрапки або дефісу можна встановити будь-який роздільник символів.

Таблиця 10.1. Типи часових інтервалів функції ADDDATE()

Тип	Опис	Формат значення
MICROSECOND	Мікросекунди	xxxxxx
SECOND	Секунди	Ss
MINUTE	Хвилини	mm
HOUR	Години	hh
DAY	Дні	DD
WEEK	Тижні	Ww
MONTH	Місяці	MM
QUARTER	Квартали	QQ
YEAR	Рік	YY
SECOND_MICROSECOND	Секунди та мікросекунди	‘ss.xxxxxx’
MINUTE_MICROSECOND	Хвилини, секунди та мікросекунди	‘mm:ss.xxxxxx’
MINUTE_SECOND	Хвилини та секунди	‘mm:ss’
HOUR_MICROSECOND	Години, хвилини, секунди та мікросекунди	‘hh:mm:ss.xxxxxx’
HOUR_SECOND	Години, хвилини та секунди	‘hh:mm:ss’
HOUR_MINUTE	Години та хвилини	‘hh:mm’
DAY_MICROSECOND	Дні, години, хвилини, секунди та мікросекунди	‘DD hh:mm:ss.xxxxxx’
DAY_SECOND	Дні, години, хвилини та секунди	‘DD hh:mm:ss’
DAY_MINUTE	Дні, години та хвилини	‘DD hh:mm’
DAY_HOUR	Дні та години	‘DD-hh’
YEAR_MONTH	Роки та місяці	‘YY-MM’

Зауваження:

Типи часових інтервалів DAY_MICROSECOND, HOUR_MICROSECOND,MINUTE_MICROSECOND,SECOND_MICROSECOND, MICROSECOND були введені в MySQL 4.1.1, а QUARTER, WEEK – починаючи з 5.0.0.

Робота з типом часового інтервалу MICROSECOND запропонована в лістингу 10.3.

Лістинг 10.3. Використання типу MICROSECOND

```
mysql> select ADDDATE('2010-02-14 03-03-03',interval 100 microsecond);
+-----------------------------------------------------+
| ADDDATE('2010-02-14 03-03-03',interval 100 microsecond) |
+-----------------------------------------------------+
| 2010-02-14 03:03:03.000100                          |
+-----------------------------------------------------+
1 row in set (0.00 sec)
```

Як видно з лістинга 10.4, якщо формат часу, що ми використовуємо, недостатній для відображення змін, отриманих в результаті складання, то MySQL автоматично розширює тип даних.

Лістинг 10.4. Використання типу MINUTE_MICROSECOND

```
mysql> select adddate('19830919',interval '22-10-100'minute_microsecond);
+-----------------------------------------------------------+
| adddate('19830919',interval '22-10-100'minute_microsecond) |
+-----------------------------------------------------------+
| 1983-09-19 00:22:10.100000                                |
+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

При використанні типу WEEK (тиждень), який запропоновано в лістингу 10.5, та QUARTER (квартал), приклад якого наведений в лістингу 10.6, виконується збільшення дати на квартали.

Лістинг 10.5. Використання типу WEEK

```
mysql> select adddate('2010/03/08',interval 54 week);
+----------------------------------------+
| adddate('2010/03/08',interval 54 week) |
+----------------------------------------+
| 2011-03-21                             |
+----------------------------------------+
1 row in set (0.00 sec)
```

Лістинг 10.6. Використання типу QUARTER (квартал)

```
mysql> select adddate('2010/03/08',interval 1 QUARTER);
+----------------------------------------+
| adddate('2010/03/08',interval 1 QUARTER) |
+----------------------------------------+
| 2010-06-08                             |
+----------------------------------------+
1 row in set (0.00 sec)
```

Окрім функції ADDDATE(), для додавання дати з часовим інтервалом можливе використання оператора «+», який діє аналогічно до функції ADDDATE().

Лістинг 10.7. Використання оператора «+» з числовими типами даних

```
mysql> select '2014-03-20'+interval '10 12/100/78.12' day_microsecond;
+-----------------------------------------------------------+
| '2014-03-20'+interval '10 12/100/78.12' day_microsecond |
+-----------------------------------------------------------+
| 2014-03-30 13:41:18.120000                                |
+-----------------------------------------------------------+
1 row in set (0.00 sec)

mysql> select '2014-03-20'+interval 30 month;
+--------------------------------+
| '2014-03-20'+interval 30 month |
+--------------------------------+
| 2016-09-20                     |
+--------------------------------+
1 row in set (0.00 sec)
```

При використанні складних форматів, якщо значення одного з параметрів виходить за межі, наприклад, вказується 100 хвилин, залишок автоматично додається до старшого розряду (1 година 40 хвилин).

Зауваження:

Поряд з оператором «+» можливе використання оператора «-» для віднімання часового інтервалу з дати.

Функція CURDATE

Функція CURDATE(), запропонована в лістингу 10.12, повертає поточну дату у форматі ‘YYYY-MM-DD’ або ‘YYYYDDMM’ в залежності від того, викликається функція в текстовому або в числовому контексті.

Лістинг 10.12. Використання функції CURDATE()

```
mysql> select CURdate(), CURdate()+0, Current_date(),CURRENT_DATE;
+------------+-------------+----------------+--------------+
| CURdate()  | CURdate()+0 | Current_date() | CURRENT_DATE |
+------------+-------------+----------------+--------------+
| 2009-11-21 |    20091121 | 2009-11-21     | 2009-11-21   |
+------------+-------------+----------------+--------------+
1 row in set (0.02 sec)
```

Зауваження:

Функція CURDATE() має два синоніма: CURRENT_DATE() та CURRENT_DATE. Тобто є варіант написання CURRENT_DATE без круглих дужок.

Функція CURTIME

Функція CURTIME() повертає поточний час доби у вигляді ‘hh-mm-ss’ або ‘hhmmss’ в залежності від того, викликається функція в текстовову чи числовому контексті.

Лістинг 10.14. Використання функції CURTIME()

```
mysql> select CURtime(), CURTIME()+0, Current_time(),CURRENT_TIME;
+-----------+---------------+----------------+--------------+
| CURtime() | CURTIME()+0   | Current_time() | CURRENT_TIME |
+-----------+---------------+----------------+--------------+
| 10:57:14  | 105714.000000 | 10:57:14       | 10:57:14     |
+-----------+---------------+----------------+--------------+
1 row in set (0.02 sec)
```

Зауваження:

Функції CURTIME() відповідають два синоніма: CURRENT_TIME() та CURRENT_TIME. Тобто є варіант написання CURRENT_TIME без круглих дужок.

Функція DATE

Функція DATE() витягує із значення дати та години доби у форматі DATE ‘2005-04-17 00:26:08’ дату, відкидаючи гадини, хвилини та секунди.

Лістинг 10.15. Використання функції DATE()

```
mysql> select DATE('2009-11-21 11/00/01');
+-----------------------------+
| DATE('2009-11-21 11/00/01') |
+-----------------------------+
| 2009-11-21                  |
+-----------------------------+
1 row in set (0.00 sec)
```

Зауваження:

Функція DATE() з’явилася в MySQL в версії 4.1.1.

Функція DATEDIFF

Функція DATEDIFF(begin, end) обчислює різницю в днях між датами begin, end. Аргументи функції можуть мати тип DATE або DATETIME, проте при обчисленні різниці використовується тільки DATE частина.

Лістинг 10.16. Використання функції DATEDIFF().

```
mysql> select datediff('2009/11/21','1983/09/19');
+-------------------------------------+
| datediff('2009/11/21','1983/09/19') |
+-------------------------------------+
|                                9560 |
+-------------------------------------+
1 row in set (0.02 sec)

mysql> select 9560/365;
+----------+
| 9560/365 |
+----------+
|  26.1918 |
+----------+
1 row in set (0.00 sec)
```

В лістингу 10.16 показано, як вираховується кількість днів з дня народження 19 вересня 1983 року і до вказаного дня 21 листопада 2009 року, для визначення кількості років ділимо отримане значення на середню кількість днів в році.

Зауваження:

Функція DATEDIFF з’явилася в MySQL в версії 4.1.1.

Функція DATE_FORMAT

Функція DATE_FORMAT(date, format) форматує час date у відповідності до рядку format. В рядку format можуть використовуватися визначники, які запропоновано в таблиці 10.2.

Всі інші символи, які не входять до таблиці 10.2, вводяться без змін. Приклад використання функції DATE_FORMAT() запропонований в лістингу 10.17.

В СУБД MySQL по замовчуванню в представленні дати спочатку потрібно вказати рік, потім місяць та день, тобто в якому порядку зручніше сортувати результат. Це не завжди є зручним, оскільки частіше потрібно спочатку розмістити дату, потім місяць, а лише потім рік. Саме для таких задач призначена функція DATE_FORMAT().

Лістинг 10.17. Використання функції DATE_FORMAT(), та перетворення дати з формату YYYY-MM-DD в DD:MM:YYYY

```
mysql> select date_FORMAT('2010/12/09','%Y:%M:%d');
+--------------------------------------+
| date_FORMAT('2010/12/09','%Y:%M:%d') |
+--------------------------------------+
| 2010:December:09                     |
+--------------------------------------+
1 row in set (0.01 sec)

mysql> select date_FORMAT('2010/12/09','%Y:%m:%d');
+--------------------------------------+
| date_FORMAT('2010/12/09','%Y:%m:%d') |
+--------------------------------------+
| 2010:12:09                           |
+--------------------------------------+
1 row in set (0.00 sec)

mysql> select date_FORMAT('2010/12/09','%y:%m:%D');
+--------------------------------------+
| date_FORMAT('2010/12/09','%y:%m:%D') |
+--------------------------------------+
| 10:12:9th                            |
+--------------------------------------+
1 row in set (0.00 sec)
```

Таблиця 10.2. Визначники рядка форматування функції DATE_FORMAT()

Визначник	**Опис**
%a	Скорочене найменування дня тижня (Sun…Sat)
%b	Скорочене найменування місяця (Jan…Dec)
%c	Місяць в числовій формі (1.12)
%D	День місяця з англійським суфіксом (1 st, 2 nd, 3 rd …)
%d	День місяця в числовій формі з керуючим нулем (01…31)
%e	День місяця в числовій формі (1..31)
%f	Мікросекунди (000000..999999)
%H	Час з керуючим нулем (00..23)
%h	Час з керуючим нулем (01..12)
%I	Час з керуючим нулем (01..12)
%i	Хвилини з керуючим нулем (00..59)
%j	День року (001..366)
%k	Час з керуючим нулем (0..23)
%l	Час без керуючого нуля (1..12)
%M	Назва місяця (January…December)
%m	Місяць в чмсловій формі з керуючим нулем (01..12)
%p	AM або PM (для 12-годинкового формату)
%r	Час, 12-годинковий формат (hh:mm:ss AM|hh:mm:ss PM)
%S	Секунди (00..59)
%s	Секунди (00..59)
%T	Час, 24-часовий формат (hh:mm:ss)
%U	Тиждень (00..52), де неділя вважається першим днем тижня
%u	Тиждень (00..52), де понеділок вважається першим днем тижня
%V	Тиждень (01..52), де неділя вважається першим днем тижня
%v	Тиждень (00..52), де понеділок вважається першим днем тижня, використовується з ‘%x’

Продовження таблиці 10.2

%W	Назва для тижня (Sunday…Saturday)
%w	День тижня (0…6) 0 – неділя, 6 - субота
%X	Рік для тижня, де неділя вважається першим днем тижня 4 розряди, використовується з ‘%V’
%x	Рік для тижня, де неділя вважається першим днем тижня 4 розряди, використовується з ‘%v’
%Y	Рік, 4 розряди (YYYY)
%y	Рід 2 розряди (YY)
%%	Літерал ‘%’

Лістинг 10.18.

```
mysql> select date_FORMAT('2010/12/09','%D:%M:%Y');
+--------------------------------------+
| date_FORMAT('2010/12/09','%D:%M:%Y') |
+--------------------------------------+
| 9th:December:2010                    |
+--------------------------------------+
1 row in set (0.00 sec)
```

Тут показано як стандартну дану для американців (рік/міс/день) перетворено в нашу стандартну (день:міс:рік).

Зауваження:

Визначник %f для мікросекунд з’явився в MySQL, починаючи з версії 4.1.0.

За допомогою функції DATE_FORMAT() час та добу з одного формату можна перетворити в будь-який інший формат. Якщо вихідний аргумент date містить зайву інформацію, він буде відкинутим, якщо інформації недостатньо, то ті елементи яких не вистачає будуть додані з присвоєнням їм нульових значень.

Лістинг 10.19. Перетворення формату за допомогою функції DATE_FORMAT()

```
mysql> select NOW(), date_format(NOW(),'%T');
+---------------------+-------------------------+
| NOW()               | date_format(NOW(),'%T') |
+---------------------+-------------------------+
| 2009-11-21 11:16:48 | 11:16:48                |
+---------------------+-------------------------+
1 row in set (0.00 sec)

mysql> select NOW(), date_format(NOW(),'%D');
+---------------------+-------------------------+
| NOW()               | date_format(NOW(),'%D') |
+---------------------+-------------------------+
| 2009-11-21 11:16:57 | 21st                    |
+---------------------+-------------------------+
1 row in set (0.00 sec)

mysql> select NOW(), date_format(NOW(),'%Y');
+---------------------+-------------------------+
| NOW()               | date_format(NOW(),'%Y') |
+---------------------+-------------------------+
| 2009-11-21 11:17:03 | 2009                    |
+---------------------+-------------------------+
1 row in set (0.00 sec)
```

Функція DAY

Функція DAY(date) є синонімом функції DAYOFMONTH(date), приймає у якості аргументу дату date та повертає порядковий номер дня в місяці.

Лістинг 10.20. Використання функції DAY()

```
mysql> select DAY('1983/09/19');
+-------------------+
| DAY('1983/09/19') |
+-------------------+
|                19 |
+-------------------+
1 row in set (0.00 sec)
```

Зауваження:

Функція DAY() введена в MySQL, починаючи з версії 4.1.1. В більш раніших версіях MySQL слід використовувати DAYOFMONTH()

Функція DAYNAME

Функція DAYNAME(date) приймає у якості аргументу дату date та повертає день тижня у вигляді повної англійської назви. Наприклад ви хочете взнати в який день тижня ви народились, наберіть вказані команди і взнаєте.

Лістинг 10.21. Використання функцій DAYNAME()

```
mysql> select DAYNAME('1983/09/19');
+-----------------------+
| DAYNAME('1983/09/19') |
+-----------------------+
| Monday                |
+-----------------------+
1 row in set (0.00 sec)
```

Зазвичай переклад англійських назв виконують в прикладній програмі. Відповідність англійських та українських назв днів тижня приведено в таблиці 10.3.

Таблиця 10.3. Відповідність англійських та українських днів тижня.

Номер	Англійська назва	Скорочений варіант	Українська назва
1	**Sunday**	Sun	Неділя
2	**Monday**	Mon	Понеділок
3	**Tuesday**	Tue	Вівторок
4	**Wednesday**	Wed	Середа
5	**Thursday**	Thu	Червер
6	**Friday**	Fri	П’ятниця
7	**Saturday**	Sat	Субота

Слід пам’ятати, що в західних країнах тиждень починається з неділі (Sunday) або з понеділка (Monday), як в Україні, тому нумерація днів введена в MySQL починаючи з неділі.

Функція DAYOFMONTH

Функція DAYOFMONTH(date), яка запропонована в лістингу 10.22, приймає в якості аргументу дату date та повертає порядковий номер дня в місяці (від 1 до 31).

Лістинг 10.22. Використання функція DAYOFMONTH()

```
mysql> select DAYofMONTH('2010/05/11');
+--------------------------+
| DAYofMONTH('2010/05/11') |
+--------------------------+
|                       11 |
+--------------------------+
1 row in set (0.00 sec)
```

Функція DAYOFWEEK

Функція DAYOFWEEK(date) приймає у якості аргументу дату date та повертає порядковий номер дня тижня. Слід пам'ятати, що в західних країнах тиждень починається з неділі, для якого функція повертає 1, для останнього дня тижня, суботи, повертається 7.

Лістинг 10.23. Використання функції DAYOFWEEK()

```
mysql> select DAYofweek('1983/09/19');
+--------------------------+
| DAYofweek('1983/09/19') |
+--------------------------+
|                        2 |
+--------------------------+
1 row in set (0.00 sec)

mysql> select DAYNAME('1983/09/19');
+------------------------+
| DAYNAME('1983/09/19') |
+------------------------+
| Monday                 |
+------------------------+
1 row in set (0.00 sec)
```

В лістингу 10.23 результат відповідає понеділку (див табл. 10.3)

Функція DAYOFYEAR

Функція DAYOFYEAR(date) приймає у якості аргументу дату date та повертає порядковий номер дня в році (від 1 до 366). Приклад застосування функції DAYOFYEAR() запропоновано в лістингу 10.24.

Лістинг 10.24. Використання функції DAYOFYEAR()

```
mysql> select DAYofyear('1983/09/19'), dayofYEAR('1983/12/19');
+-------------------------+------------------------+
| DAYofyear('1983/09/19') | dayofYEAR('1983/12/19') |
+-------------------------+------------------------+
|                     262 |                    353 |
+-------------------------+------------------------+
1 row in set (0.00 sec)
```

Функція EXTRACT

Функція EXTRACT(type FROM datetime) приймає дату та годину доби datetime та повертає частину, яка визначається параметром type, який може приймати значення з першого стовпця.

Лістинг 10.25. Використання функції EXTRACT()

```
mysql> select EXTRACT(Day from '1982/05/11');
+--------------------------------+
| EXTRACT(Day from '1982/05/11') |
+--------------------------------+
|                             11 |
+--------------------------------+
1 row in set (0.00 sec)
```

Окрім повної форми часу ‘YYYY-MM-DD hh:mm:ss’, допускається використання скороченої форми, яка містить тільки дату ‘YYYY-MM-DD’.

Витягування за допомогою функції EXTRACT() мікросекунд, секунд, хвилин та годин з скороченого варіанту часу, який містить тільки дату ‘YYYY-MM-DD’, призводить до непередбаченого результату.

Функція FROM_DAYS

Функція FROM_DAYS(N) приймає число днів N, які минули з нульового року, та повертає дату у форматі ‘YYYY-MM-DD’.

Лістинг 10.27. Використання функції FROM_DAYS()

```
mysql> select from_days(0),from_days(567894);
+--------------+-------------------+
| from_days(0) | from_days(567894) |
+--------------+-------------------+
| 0000-00-00   | 1554-11-04        |
+--------------+-------------------+
1 row in set (0.00 sec)
```

Звичайно функція FROM_DAYS() використовується разом з функцією TO_DAYS(date), яка вирішує зворотню задачу – перетворює дату date з формату ‘YYYY-MM-DD’ в кількість днів, які минули з нульового року.

Функція GET_FORMAT

Ця функція повертає рядок форматування, яка відповідає одному з п’яти стандартів часу. Отриманий рядок можна потім використати в функціях DATE_FORMAT() та SPR_TO_DATE(). Функція має наступний синтаксис:

GET_FORMAT (DATE|TIME|DATETIME, ‘EUR’ | ‘USA’ | ‘JIS’ | ‘ISO’ | ‘INTERVAL’)

У якості першого параметру виступає одне з ключових слів:

DATE – отримати формат ля дати;

TIME – отримати формат для часу;

DATETIME – отримати формат для часу та дати.

Другий параметр задає назву формату, згідно до якого слід від форматувати час, та може прийняти один з рядків:

EUR – європейський стандарт IBM;

USA – американський стандарт IBM;

JIS – японський індустріальний стандарт;

ISO – стандарт ISO (міжнародна організація стандартів);

INTERNAL – інтернаціональний стандарт.

Приклад використання функції GET_FORMAT() наведений в лістингу 10.32.

Зауваження:

Функція GET_FORMAT() введена в MySQL починаючи з версії 4.1.1.

Лістинг 10.32. Використання функції GET_FORMAT ()

```
mysql> select GET_format(date,'EUR');
+------------------------+
| GET_format(date,'EUR') |
+------------------------+
| %d.%m.%Y               |
+------------------------+
1 row in set (0.00 sec)
```

Представлення часу у всіх п’яти стандартах представлено в таблиці 10.4.

Таблиця 10.4. Представлення часу у всіх п’яти стандартах

Виклик функції	Результат
GET_FORMAT (DATE, ‘USA’)	‘%m.%d.%Y’
GET_FORMAT (DATE, ‘JIS’)	‘%Y-%m-%d’
GET_FORMAT (DATE, ‘ISO’)	‘%Y-%m-%d’
GET_FORMAT (DATE, ‘EUR’)	‘%d.%m.%Y’
GET_FORMAT (DATE, ‘INTERVAL’)	‘%Y%m%d’
GET_FORMAT (DATETIME, ‘USA’)	‘%Y-%m-%d-%H.%i.%s’
GET_FORMAT (DATETIME, ‘JIS’)	‘%Y-%m-%d%H:%i:%s’
GET_FORMAT (DATETIME, ‘ISO’)	‘%Y-%m-%d%H:%i:%s’
GET_FORMAT (DATETIME, ‘EUR’)	‘%Y-%m-%d-%H.%i.%s’
GET_FORMAT (DATETIME, ‘INTERVAL’)	‘%Y%m%d%H%i%s’
GET_FORMAT (TIME, ‘USA’)	‘%h:%i:%s%p’
GET_FORMAT (TIME, ‘JIS’)	‘%H:%i:%s’
GET_FORMAT (TIME, ‘ISO’)	‘%H:%i:%s’
GET_FORMAT (TIME, ‘EUR’)	‘%H.%i.%S’
GET_FORMAT (TIME, ‘INTERVAL’)	‘%H%i%s’

Приклад використання функції GET_FORMAT() разом з DATE_FORMAT() наведено в лістингу 10.33.

Лістинг 10.33. Сумісне використання GET_FORMAT() та DATE_FORMAT()

```
mysql> select date_format('2005/10/21', GET_format(date,'internal'));
+-----------------------------------------------------------+
| date_format('2005/10/21', GET_format(date,'internal')) |
+-----------------------------------------------------------+
| 20051021                                                  |
+-----------------------------------------------------------+
```

Функція HOUR

Функція HOUR(datetime) повертає значення числа (від 0 до 23) для часу datetime.

Лістинг 10.34. Використання функції HOUR

```
mysql> select hour('06:00:45');
+------------------+
| hour('06:00:45') |
+------------------+
|                6 |
+------------------+
1 row in set (0.00 sec)
```

Однією з особливостей функції HOUR() є те, що якщо час доби виходить за межі 24 години, функція поверне більше число.

Функція LAST_DAY

Функція LAST_DAY(datetime) приймає у якості параметру значення дати datetime в скороченому ‘YYYY-MM-DD’ чи в розширеному форматі ‘YYYY-MM-DD hh:mm:ss’ та повертає дату в скороченому форматі ‘YYYY-MM-DD’, останній день вказаного місяця.

Лістинг 10.36. Використання функції LAST_DAY()

```
mysql> select last_day('2009/11/21'), last_DAY('1980/04/25');
+------------------------+-----------------------+
| last_day('2009/11/21') | last_DAY('1980/04/25') |
+------------------------+-----------------------+
| 2009-11-30             | 1980-04-30            |
+------------------------+-----------------------+
```

Зауваження:

Функція LAST_DAY() введена в MySQL починаючи з версії 4.1.1.

Функція MAKEDATE

Функція MAKEDATE(year, dayofyear) приймає у якості параметрів рік year, номер дня в році dayofyear та повертає дату у форматі 'YYYY-MM-DD'. Наприклад, якщо ви раптом захотіли взнати яке число буде мати день, якщо від початку року пройшло 325 днів.

Лістинг 10.37. Використання функції MAKEDATE()

```
mysql> select makedate(2009,325);
+--------------------+
| makedate(2009,325) |
+--------------------+
| 2009-11-21         |
+--------------------+
1 row in set (0.00 sec)
```

Параметр dayofyear повинен обов'язково бути більшим від нуля, в протилежному випадку повертається NULL.

Зауваження:

Функція MAKEDATE() введена в MySQL починаючи з версії 4.1.1.

Функція MONTHNAME

Функція MONTHNAME(datetime) повертає рядок з назвою місяця для дати datetime.

Лістинг 10.42. Використання функції MONTHNAME()

```
mysql> select MONTHname('2009/11/21'), monthName('1983%09%19');
+-------------------------+-------------------------+
| MONTHname('2009/11/21') | monthName('1983%09%19') |
+-------------------------+-------------------------+
| November                | September               |
+-------------------------+-------------------------+
1 row in set (0.00 sec)
```

Звичайно переклад англійських назв виконують в прикладній програмі. Відповідність англійських та українських місяців наведено в таблиці 10.5.

Таблиця 10.5. Відповідність англійських та українських назв місяців

Номер	Англійська назва	Скорочений варіант	Українська назва
1	**January**	Jan	Січень
2	**February**	Feb	Лютий
3	**March**	Mar	Березень
4	**April**	Apr	Квітень
5	**May**	May	Травень
6	**June**	Jun	Червень
7	**Jule**	Jul	Липень
8	**August**	Aug	Серпень
9	**September**	Sep	Вересень
10	**October**	Oct	Жовтень
11	**November**	Nov	Листопад
12	**December**	Dec	Грудень

Функція NOW

Функція NOW() повертає поточну дату та час у вигляді рядка у формі 'YYYY-MM-DD hh:mm:ss' або YYYYMMDDhhmmss, в залежності від того, викликається функція в рядковому чи числовому контексті.

Зауваження:

Функція NOW() має п'ять синонімів: LOCALTIME(), LOCALTIME, LOCALTIMESTAMPS(), LOCALTIMESTAMPS, SYSDATE().

Лістинг 10.43. Використання функції NOW()

```
mysql> select now();
+---------------------+
| now()               |
+---------------------+
| 2009-11-21 12:08:04 |
+---------------------+
1 row in set (0.00 sec)
```

Функція PERIOD_ADD

Функція PERIOD_ADD(period, N) додає N місяців до закінчення дати period. Аргумент period повинен бути представлений у числовому форматі 'YYYYMM' або YYMM. Передача у якості аргументу дати в будь-якому іншому форматі призводить до непередбаченого результату (результат в другому стовпцю).

Лістинг 10.44. Використання функції PERIOD_ADD()

```
mysql> select PERIOD_add(200911,6),period_ADD(2010/11,7);
+----------------------+----------------------+
| PERIOD_add(200911,6) | period_ADD(2010/11,7) |
+----------------------+----------------------+
|               201005 |                200806 |
+----------------------+----------------------+
1 row in set (0.00 sec)
```

Результат представляє собою число у форматі YYYYMM.

Функція PERIOD_DIFF

Функція PERIOD_DIFF(period1, period2) обчислює різницю в місяцях між двома датами period1 та period2, які представлені у числовому форматі 'YYYYMM' або 'YYMM'. Передача у якості аргументів дати в будь-якому іншому форматі призводить до непередбаченого результату.

Лістинг 10.45. Використання функції PERIOD_DIFF()

```
mysql> select period_diff(198212,198309);
+----------------------------+
| period_diff(198212,198309) |
+----------------------------+
|                         -9 |
+----------------------------+
1 row in set (0.00 sec)
```

Результатом є цілочисельне число: додатнє, якщо перший аргумент більше другого, та від'ємне, якщо перший аргумент менше другого.

Контрольні питання

1. Основні функції дати та часу.
2. Суть функції CURDATE, CURTIME
3. Щоб взнати кількість днів, які минули з вказаного часу до заданої дати, потрібно використати оператор …
4. Суть оператора DATE_format…
5. Для визначення дня тижня заданої дати потрібно використати запит у вигляді…
6. Відмінність оператора DAYNAME і DAYofWEEK
7. Формат функції DAYofYEAR…

Які існують стандартні формати представлення дати в MySQL

Список літератури

1. Верьовкіна Г.В. Система управління базами даних Access: Навчальний посібник з дисципліни "СУБД" / Г.В. Верьовкіна. – Київ: Київський національний університет імені Тараса Шевченка, 2022. – 71 с.

2. Гайна Г.А. Основи проектування баз даних: навчальний посібник / Г.А. Гайна. - Київ: КНУБА, 2005. 204 с.

3. Доценко С.І. Організація та системи керування базами даних: Навчальний посібник / С.І. Доценко - Харків: УкрДУЗТ, 2023. - 117 с.

4. Пасічник В.В., Реаніченко В.А. Організація баз даних та знань. Київ: Видавнича група ВНУ, 2006. - 384 с.

5. Руденко В.Д. Інформатика: бази даних / В.Д. Руденко. - Харків: Вид-во «Ранок» , 2019. - 112 с.

З М І С Т

Навчально-методичне видання

С И С Т Е М И Б А З И Д А Н И Х І З Н А Н Ь
Частина ІІ
(SQL)

Методичні вказівки до виконання практичних робіт з курсу “Системи бази даних і знань” (SQL) ч. ІІ для студентів спеціальності G7 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка», 76 с.

Укладачі: М.О. Федотова, канд. техн. наук, асистент
Д.В. Трушаков, канд. техн. наук, доцент

Формат 60х84 1/16 Ум. друк. арк. 5. 2025рік.