МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ

НАЦІОНАЛЬНА АКАДЕМІЯ ВНУТРІШНІХ СПРАВ
КАФЕДРА КРИМІНОЛОГІЇ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ВИКОРИСТАННЯ ТЕХНОЛОГІЙ SQL ТА NoSQL БАЗ ДАНИХ ДЛЯ ВИРІШЕННЯ ЗАВДАНЬ КРИМІНАЛЬНОГО АНАЛІЗУ

Навчальний посібник



КИЇВ-2024

УДК 343.1.85 (477)
М11

Авторський колектив:

Школьніков Владислав, доктор філософії (Національна академія внутрішніх справ);

Корнейко Олександр, кандидат технічних наук, професор (Національна академія внутрішніх справ).

Рецензенти:

Звєрєв Володимир, кандидат технічних наук, старший науковий співробітник (Апарат Ради національної безпеки та оборони України);

Орлов Юрій, доктор юридичних наук, старший науковий співробітник (Національна академія внутрішніх справ).

Схвалено та рекомендовано до друку Вченою радою Національної академії внутрішніх справ 24 грудня 2024 року (протокол № 25/3-8).

М 11 Використання технологій SQL та NoSQL баз даних для вирішення завдань кримінального аналізу [Текст] : навч. посібн. / [В. Школьніков, О. Корнейко]. К. : Вид-во Нац. акад. внутр. справ, 2024. 70 с.

У навчальному посібнику висвітлено загальний порядок використання можливостей технологій SQL та NoSQL баз даних при роботі з інформацією в електронній (цифровій) формі під час здійснення кримінального аналізу.

Посібник розрахований на працівників оперативних та аналітичних підрозділів Національної поліції та інших правоохоронних органів України, викладачів, курсантів та слухачів закладів вищої освіти зі специфічними умовами навчання.

УДК 343.1.85 (477)

© Національна академія внутрішніх справ

© Школьніков В., Корнейко О., 2024

ЗМІСТ

Стор.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ....5
ВСТУП....6
1. ЗАГАЛЬНІ ПОНЯТТЯ ПРО ТЕХНОЛОГІЇ БАЗ ДАНИХ....9
1.1. Дані та програми....9
1.2. Історія розвитку СКБД....10
1.3. Основи реляційних баз даних....11
1.4. Принцип ACID....14
1.5. Теорема CAP....15
Завдання для самоконтролю....18
2. РЕЛЯЦІЙНА БАЗА ДАНИХ MYSQL....19
2.1. Архітектура MySQL....19
2.2. Клієнти MySQL....19
2.3. Керування базою даних....28
2.4. Порядок створення таблиць в MySQL....31
Завдання для самоконтролю....33
3. МОВА ЗАПИТІВ SQL....35
3.1. Поняття мови запитів SQL....35
3.2. Коментарі в мові запитів SQL....37
3.3. DDL та DML....38
3.4. Структура запиту SQL....38
3.5. Типи даних в мові запитів SQL....40
3.5.1. Числовий тип даних....41
3.5.2. Текстовий тип даних....43
3.5.3. Тип NULL....45
3.5.4. Календарний тип даних....47
3.5.5. Колекційні типи даних....49
Завдання для самоконтролю....51
4. Індекси та ключи....52
4.1. Загальне поняття індексів та ключів....52
4.2. Види індексів....52
Завдання для самоконтролю....58
5. CRUD операції....60
5.1. Загальне поняття....60
5.2. INSERT....60

5.3. SELECT..... 61
Завдання для самоконтролю..... 61
6. АГРЕГАЦІЯ ДАНИХ..... 63
6.1. Групування даних..... 63
Завдання для самоконтролю..... 66
КВЕСТОВІ ЗАВДАННЯ..... 67
ВИСНОВКИ..... 72
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СКБД	система керування базами даних
КА	кримінальний аналіз
ПК	персональний комп’ютер

ВСТУП

На даний час методи та технології кримінального аналізу (КА) широко застосовуються Національною поліцією та іншими правоохоронними органами України, правоохоронцями країн Євросоюзу, США та інших зарубіжних країн під час розкриття кримінальних правопорушень та прогнозування злочинності.

При цьому великі об’єми оперативної інформації, а також їхня неструктурованість, вимагають від правоохоронців при здійсненні КА наявності відповідних технічних вмінь, знань та навичок з обробки та аналізу такої інформації за допомогою спеціалізованих програмних інформаційних технологій.

На даний час однією із технологій, що застосовується аналітиками правоохоронних органів і дозволяє ефективно проаналізувати інформацію в електронній (цифровій) формі, є технології SQL та NoSQL баз даних (*далі – технології баз даних*).

Технології баз даних дозволяють максимально ефективно обробляти та аналізувати інформацію.

Під час виконання завдань КА за допомогою технологій баз даних можливо виконувати різноманітні задачі, в тому числі обробляти та аналізувати:

1) інформацію з електронних комунікаційних мереж:

– з’єднання абонентів мобільного зв’язку;

– історії використання мобільних терміналів та телефонних сім-карток;

2) рух товарів (надання послуг), грошових коштів та віртуальних активів:

– рух товарів (надання послуг) з Єдиного реєстру податкових накладних (ЄРПН) та автоматизованої системи митного оформлення (АСМО) “Інспектор”;

– рух грошових коштів за банківськими рахунками;

– рух грошових коштів за допомогою електронних платіжних систем;

– рух віртуальних активів;

3) інформацію з державних інформаційних ресурсів:

– з системи “Аркан”;

– з Єдиного реєстру довіреностей;

– з Державного реєстру речових прав на нерухоме майно;

– з інформаційно-аналітичного комплексу “Безпечне місто” та інформаційної підсистеми “Гарпун” “Інформаційного порталу Національної поліції України” про маршрути пересування транспортних засобів;

4) інформацію з комп’ютерів та мобільних терміналів зв’язку;

5) відкриті дані:

– з Єдиного державного веб-порталу відкритих даних;

– з Єдиного веб-порталу використання публічних коштів;

– з електронної системи публічних закупівель Prozorro;

– з системи онлайн аукціонів з продажу та здачі в оренду майна ProZorro.Продажі тощо.

Але використання технологій баз даних при здійсненні КА потребує від правоохоронців відповідних знань, умінь та навичок роботи з даними за допомогою вищевказаної технології.

Тому авторами підготовлений цей посібник, який розрахований на працівників оперативних та аналітичних підрозділів Національної поліції та інших правоохоронних органів України, викладачів, курсантів, студентів, слухачів закладів вищої освіти зі специфічними умовами навчання.

Метою даного навчального посібнику є надання базових навичок користування технологіями бази даних під час обробки та аналізу даних при вирішенні типових завдань КА.

У даному посібнику демонструється порядок роботи з реляційною базою даних MySQL.

Розділ 1 даного навчального посібника описує загальні поняття про технології баз даних.

Розділ 2 описує порядок роботи з реляційною базою даних MySQL.

Розділ 3 містить навчальний матеріал щодо мови запитів SQL.

Розділ 4 стосується порядку використання індексів та ключів.

Розділ 5 розкриває читачу особливості роботи з CRUD операціями.

Розділ 6 описує порядок агрегації даних.

Наведені у посібнику методи застосування інструментів обробки та аналізу за допомогою технологій баз даних доповнені прикладами їх використання у реальних завданнях здійснення КА.

Остання версія посібника **із змінами та доповненнями** доступна за посиланням: docs.google.com/document/d/1rXja_yGbZlpZEwimzQO5p_pBSIL3fEKTua_6VfrP5tc (**станом на 24.12.2024**).

PDF версія посібника для завантаження доступна за посиланням: drive.google.com/drive/folders/1kDA_dbkBEbGXfTsfeDU2mS91TewX9E3X.

1. ЗАГАЛЬНІ ПОНЯТТЯ ПРО ТЕХНОЛОГІЇ БАЗ ДАНИХ

1.1. Дані та програми

Дані зберігаються довше, ніж час функціонування комп'ютерних програми. Це обумовлено тим, що програми постійно видозмінюються, вдосконалюються або зникають (не підтримуються). Також одні і ті самі дані можуть використовуватися декількома комп'ютерними програмами.

Тому в програмуванні прийнято відокремлювати дані від коду та тримати дані у спеціалізованому сховищі – базі даних.

База даних – це іменована сукупність даних, що відображає стан об'єктів та їх відношень у визначеній предметній області. Бази даних створюються за допомогою сучасних інформаційних технологій.

Обіг великих об'ємів інформації (Big Data) зумовлює необхідність зберігання таких масивів у бінарному форматі із використанням механізмів стиснення об'єму. У даному випадку використання звичайних текстових форматів є неефективним способом.

У разі необхідності зі спеціалізованого сховища – бази даних – можно отримати дані у форматах, які прийнятні для звичайного користувача. Окрім цього, бази даних можуть забезпечувати безперебійний процес зчитування та запису даних двома і більше користувачами.

З появою операційних систем та баз даних почали з'являтися спеціалізовані комп'ютерні програми, які дозволяють керувати, шукати, зберігати та редагувати дані всередині баз даних. Назва таких спеціалізованих комп'ютерних програм – системи керування базами даних (СКБД).

1.2. Історія розвитку СКБД

Сучасні бази даних протягом останніх 60 років пройшли тривалий шлях розвитку.

Перші бази даних були ієрархічними. Ієрархія – це дерево, що складається з вузлів, у яких може бути кілька нащадків. За допомогою такої структури добре описуються ієрархічні структури організацій та виробництв.

Головна їхня перевага – висока швидкість обробки операцій. Перші комп'ютери не відрізнялися високою продуктивністю – чим простіше організована база даних, тим швидше вона працює.

Основний недолік ієрархічної структури бази даних – неможливість реалізувати відносини "багато-до-багатьох".

Тому, було розроблено нову модель даних – мережева. Вона розширила можливості ієрархічної модель, дозволяючи одному запису брати участь у кількох відносинах "предок-нащадок".

У зв'язку з розвитком соціальних мереж ці бази даних отримали друге дихання у вигляді графових СКБД, які належать до сучасних технологій NoSQL баз даних.

Теоретичні основи реляційної моделі даних вперше були описані британським вченим, співробітником компанії IBM Едгаром Франком Коддом (E. F. Codd) у 1970 році в статті "A Relational Model of Data for Large Shared Data Banks". Спочатку його робота викликала лише академічний інтерес, однак через 10 років такі компанії як Oracle, IBM та безліч інших компаній на основі статті Е.Ф. Кодда почали створювати реляційні бази даних.

Реляційні бази даних найпоширеніший вид баз даних, які на сьогодні актуальні.

Довгий час на зміну реляційним базам даних пророкували прихід XML та об'єктно-орієнтованих баз даних.

Однак ці технології так і не стали масовими. XML не популярний через свою надмірність. ООП-бази даних, які вирішують проблему несумісності

реляційних баз даних, заснованих на множинах, та ООП-програм, заснованих на деревах, теж не стали надто популярними.

Найбільш популярні СКБД на сьогоднішній день – Oracle, MS SQL і DB2 серед комерційного програмного забезпечення; MySQL, PostgreSQL та Firebird серед вільного програмного забезпечення.

Слідкувати за індексом популярності баз даних можна на сайті https://db-engines.com/en/.

З розвитком сучасних технологій та збільшенням даних по експоненті змінилися підходи до організації баз даних – використання розподілених сховищ та більшого об'єму оперативної пам'яті. За таких умов стало складно реалізовувати стандартну мову структурованих запитів SQL.

У результаті почали з'являтися спеціалізовані СКБД, орієнтовані під вирішення спеціалізованих завдань:

Redis – розподілене сховище, побудоване за принципом "ключ-значення", які зберігаються в оперативній пам'яті.

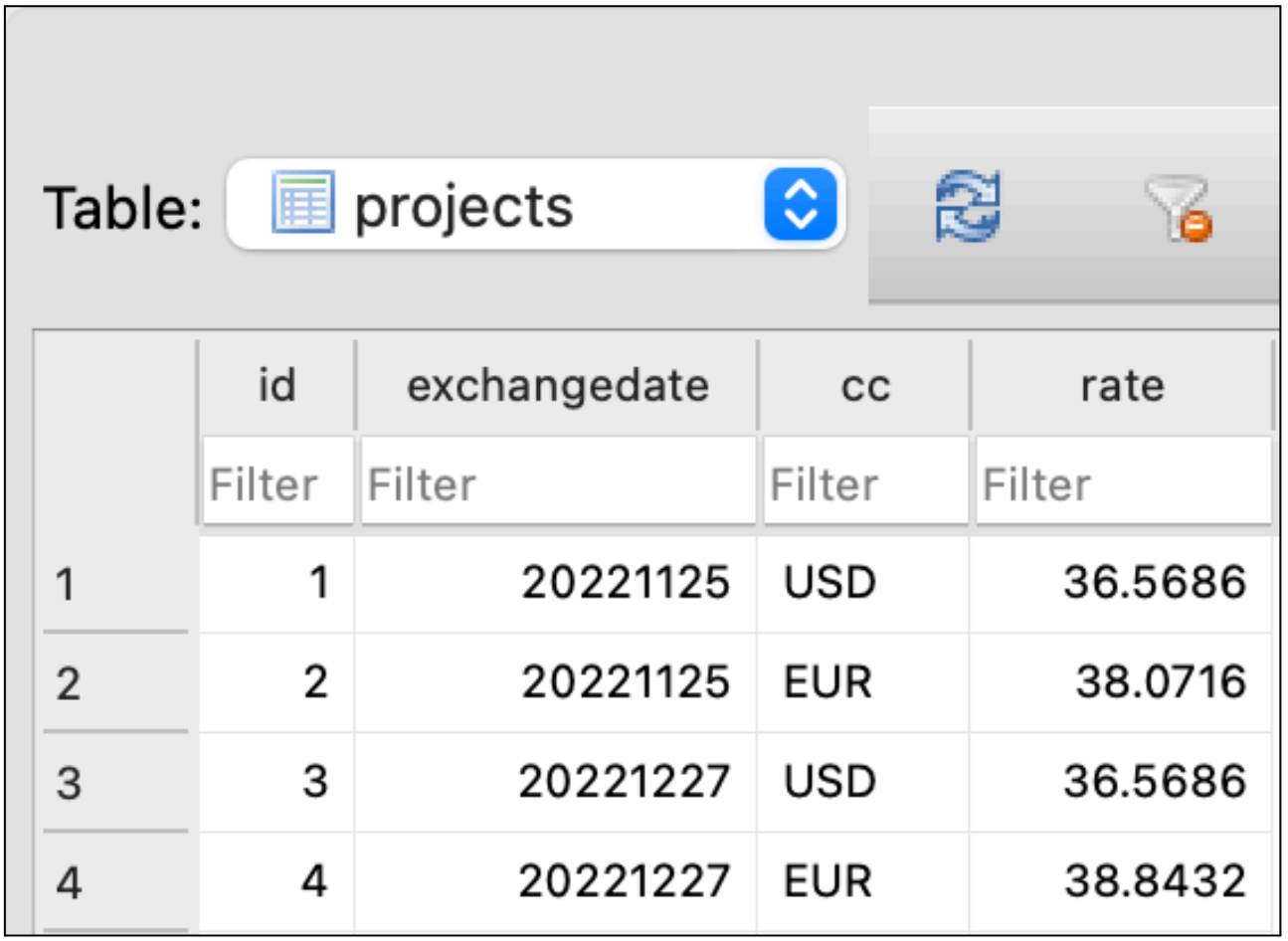
MongoDB – документо-орієнтована база даних з відкритим вихідним кодом, яка не потребує опису схеми таблиць.

ElasticSearch – пошуковий сервер, розроблений на базі Lucene, який надає можливість повнотекстового пошуку.

1.3. Основи реляційних баз даних

У реляційних базах даних інформація організована у вигляді прямокутних таблиць, розділених на рядки та стовпці, на перетині яких містяться значення.

База даних може складатися з кількох таблиць. Кожна таблиця має унікальне ім'я, яке описує її вміст. На рис. 1 відображено вміст таблиці `projects`.



	id	exchangedate	cc	rate
	Filter	Filter	Filter	Filter
1	1	20221125	USD	36.5686
2	2	20221125	EUR	38.0716
3	3	20221227	USD	36.5686
4	4	20221227	EUR	38.8432

Рис. 1.

Дані, що містяться в одному стовпці, мають однаковий тип. Наприклад, у стовпці `cc` містяться лише текстові дані, а в стовпці `exchangedate` – числові. У кожного стовпця в таблиці є своє ім'я, яке зазвичай слугує його заголовком. Усі стовпці в одній таблиці повинні мати унікальні імена. Інші таблиці можуть також містити ті самі назви стовпців. Наприклад, такі імена стовпців як `name`, `id`, `description` тощо часто зустрічаються у різних таблицях однієї бази даних.

Стовпці таблиці упорядковані зліва направо, порядок яких визначається під час створення таблиці. У будь-якій таблиці завжди є як мінімум один стовпець.

На відміну від стовпців, у рядків таблиці немає певного порядку. Це означає, що якщо послідовно виконати два однакові запити для відображення вмісту таблиці, то немає гарантії, що кожного разу рядки будуть відображені в однаковій послідовності.

Звичайно можна передбачити сортування рядків в SQL запиті.

У таблиці може бути будь-яка кількість рядків. Якщо кількість рядків дорівнює нулю (0), то у цьому випадку таблиця називається порожньою.

Порожня таблиця зберігає структуру, визначену її стовпцями, просто у ній не містяться дані.

Оскільки рядки в реляційній таблиці не впорядковані, не можна вибрати рядок за номером в таблиці. У таблиці немає першого, останнього чи тринадцятого рядка.

У правильно побудованій реляційній базі даних у кожній таблиці є стовпець (або комбінація стовпців), для якого значення у всіх рядках різні. Цей стовпець (або стовпці) називається первинним ключем (`primary key`) таблиці.

Первинний ключ кожного рядка унікальний. У таблиці з первинним ключем немає двох зовсім однакових рядків.

Таблиця, в якій всі рядки відрізняються один від одного, називається відношенням (`relation`). Саме від цього терміну походить термін “реляційні бази даних”. Тобто в основі таких баз даних лежать відношення, тобто таблиці з різними один від одного рядками.

Стовпець однієї таблиці, значення в якому збігаються зі значеннями стовпця, що є первинним ключем іншої таблиці, називається зовнішнім ключем (`foreign key`).

Також однією із головних концепцій баз даних є транзакція. У програмному забезпеченні можуть виникати різноманітні помилки або відбутися відмова апаратного забезпечення, тому за допомогою транзакцій можно успішно виконати всі операції запису та читання даних або завершити такі операції із перериванням та відкатом до вихідного стану в безпечному режимі.

Транзакції значно спрощують для програми обробку помилок, так як немає потреби піклуватися про часткові відмови, коли частина операцій завершилася успішно, а частина – ні.

1.4. Принцип ACID

Практично всі реляційні бази даних підтримують транзакції.

NoSQL бази даних, що з'явилися наприкінці 2000-х років, ставили за мету покращити ефективність роботи баз даних, використовуючи нові моделі даних відмінні від реляційної моделі. Наслідком таких нововведень стало зниження надійності транзакцій.

Багато баз даних нового покоління повністю відмовилися від транзакцій або змінили набір гарантій безпеки надійної роботи транзакцій.

Набір властивостей, що гарантують надійну роботу транзакцій бази даних позначати абревіатурою ACID (англ. Atomicity, Consistency, Isolation, Durability) – атомарність, узгодженість, ізольованість та довговічність.

Atomicity **(атомарність)** гарантує, що у разі виникнення помилки (внаслідок відключення електроенергії, помилок програмного забезпечення або іншого роду збоїв) транзакція не буде виконана частково. Тобто виконані мають бути всі операції в межах транзакції або у разі виникнення помилки – жодна операція.

Прикладом атомарної транзакції є переказ грошей з рахунку на рахунок, який проходить двома операціями: списання грошей з першого рахунку та зарахування їх на другий рахунок. Виконання цих операцій в атомарній транзакції забезпечує узгодженість даних бази, тобто гроші не будуть списані та не будуть зараховані, якщо одна з цих двох операцій зазнає невдачі [1].

Consistency **(узгодженість)** гарантує, що будь-які дані, які записуються в базу даних, мають відповідати усім визначеним правилам, що встановлені організацією. Якщо транзакція починається зі стану бази даних, який задовольняє цим правилам, то після виконання транзакції цей стан не має порушуватись.

Isolation **(ізольованість)** означає, що паралельно виконувані транзакції ізольовані один від одного. Виконання операцій в цих транзакціях не маю

заважати один одному. База даних гарантує, що результат паралельного виконання транзакцій має бути такий самий, якщо б вони виконувались послідовно.

`Durability` **(довговічність)** гарантує, що дані не будуть втрачені у разі виникнення помилки (внаслідок відключення електроенергії, помилок програмного забезпечення або іншого роду збоїв).

NoSQL бази даних не завжди дотримуються `ACID` у зв'язку з тим, що з розвитком мережі Інтернет одночасно до бази даних можуть звертатися велика кількість клієнтів. Це призводить до проблеми масштабованості баз даних. Навіть у невеликих реляційних базах даних буде виникати проблема уповільнення операцій із-за використання комп'ютерних мереж багатьма клієнтами. Окрім того, вертикальне масштабування (збільшення потужностей серверів, у яких розташована база даних) ефективніше замінити горизонтальним масштабуванням, коли база даних розподілена на декількох пристроях (серверах).

Як тільки база даних стає розподіленою, виникає необхідність забезпечувати її стійкість до мережевих збоїв.

1.5. Теорема CAP

Протиріччя забезпечення узгодженості розподілених комп'ютерних систем описує теорема CAP, яка у 2000 році була сформульована Еріком Брюером. Теорема стосується трьох взаємопов'язаних вимог, що існують у великій розподіленій комп'ютерній системі:

1) `Consistency` **(узгодженість)** означає, що всі клієнти повинні отримати ті самі дані у відповідь на один і той самий запит, навіть якщо одночасно з цим виконуються оновлення системи.

2) `Availability` **(доступність)** відповідає тому, що всі клієнти бази даних завжди мають можливість читати та записувати дані.

3) Partition tolerance **(стійкість до розподілу)** означає, що базу даних можна розподілити між різними пристроями (серверами) та у разі відмови роботи одного сегмента база даних продовжить роботу.

Теорема CAP стверджує, що у будь-якій системі можна гарантовано забезпечити виконання лише двох із вищезазначених трьох вимог – “програма може бути гарною, швидкою та дешевою – обирай будь-які дві властивості”.

Тому можно розподілені системи класифікувати на наступні категорії:

CA – забезпечується доступності (availability) та узгодженості (consistency), але відсутня стійкості до розподілу (partition tolerance).

AP – не гарантується цілісність даних (consistency), але наявний високий рівень доступності (availability) та стійкості до розподілу (partition tolerance).

CP – забезпечується цілісність даних (consistency) та стійкості до розподілу (partition tolerance), але не гарантується доступність (availability).

У розподіленій системі мережеві збої дуже ймовірні. Неможливо уникнути затримок і втрати даних, що передаються по мережі. Будь-яка база даних може розташовуватися лише на одній грані трикутника теореми CAP (рис. 2).

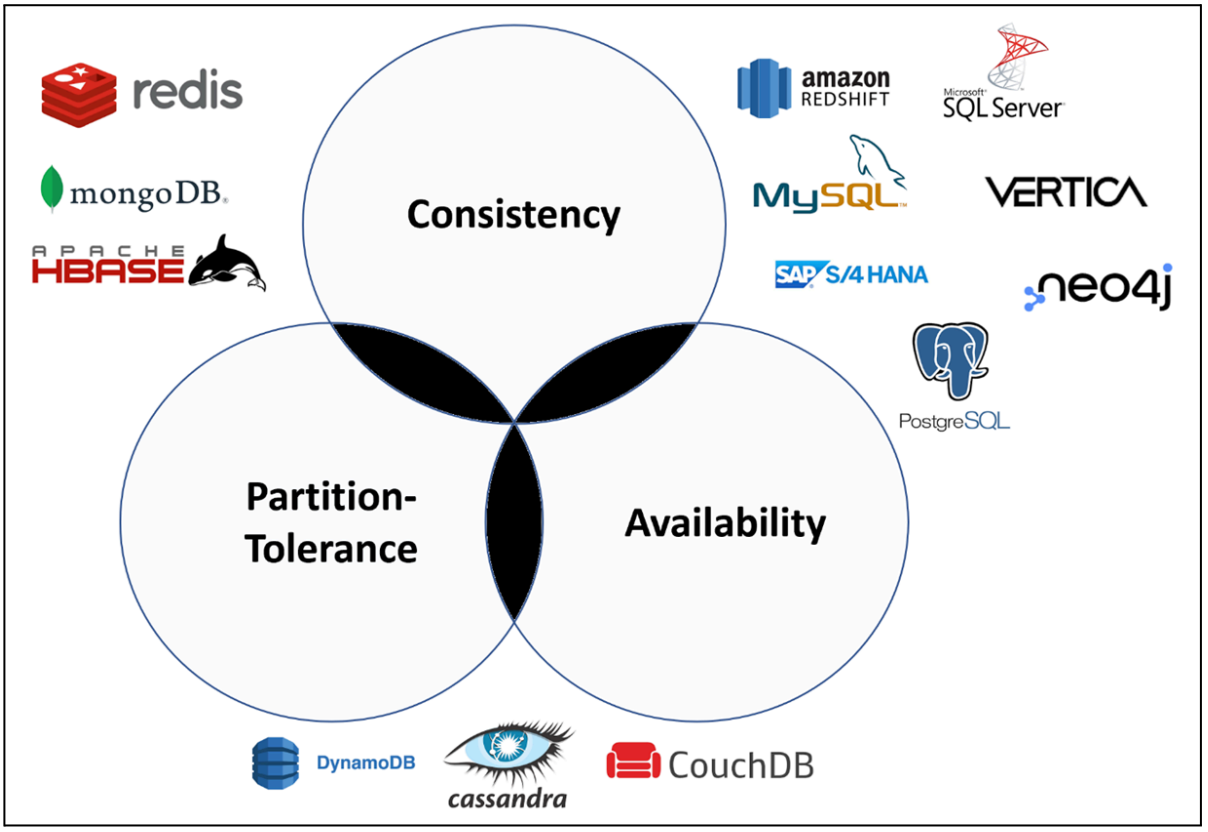


Рис. 2.

Традиційні реляційні бази даних, спроектовані для роботи на одному пристрої (сервері), тому забезпечують узгодженість (consistency) та доступність (availability). Наприклад, MySQL, PostgreSQL.

Існують сторонні рішення від Google і від компанії Percona, які дозволяють перенести базу даних MySQL на межу CP, тобто забезпечити узгодженість даних (consistency) та стійкість до розподілу (partition tolerance). Однак у цьому випадку страждає доступність (availability) – клієнт змушений чекати результату операції оновлення бази даних.

Існує велика кількість найрізноманітніших NoSQL баз даних, які знаходяться на різних гранях трикутника CAP теореми.

Удосконалення IT-проєктів, розподілена система сучасних веб-застосунків призводить до того, що потрібні нові бази даних, які залишаються стійкими до розподілу (partition tolerance). Саме із-за на сьогодні великий попит на NoSQL бази даних, які компенсуються слабку пристосованість реляційних баз даних до розподіленої роботи.

Тому в сучасних IT-проєктах реляційну базу даних найчастіше супроводжує одна або кілька NoSQL баз даних. Реляційна база даних виступає як основне сховище, а NoSQL бази даних вирішують спеціалізовані завдання чи забезпечують мережеву зв’язність за умов нестабільної роботи.

Завдання для самоконтролю

1. Роль бази даних в сучасних IT-проєктах.
2. Що таке СКБД?
3. Хто вперше описав теоретичні основи реляційної бази даних?
4. Історія розвитку СКБД.
5. Історія розвитку реляційних баз даних.
6. Що таке `primary key`?
7. Що таке `foreign key`?
8. Що таке `ACID`?
9. Що таке `Atomicity`?
10. Що таке `Consistency`?
11. Що таке `Durability`?
12. Теорема CAP.
13. Що таке `CA`?
14. Що таке `AP`?
15. Що таке `CP`?

2. РЕЛЯЦІЙНА БАЗА ДАНИХ MYSQL

2.1. Архітектура MySQL

MySQL на сьогоднішній день одна із найпопулярніших баз даних із відкритим кодом. За популярністю її перевершує лише комерційна СКБД Oracle. MySQL використовується в багатьох проєктах. Існує безліч розгалужень цієї бази даних: Percona, MariaDB.

Архітектуру MySQL можна умовно розділити на наступні частини:

1) ядро, тобто сама база даних;

2) так звані “двигуни”, які реалізують той чи інший механізм бази даних. За замовчуванням використовується InnoDB.

Така архітектура дозволяє розробляти базу даних зусиллями кількох команд. Одна команда може зосередитися на базі даних, інша – на якомусь окремому механізмі. Наприклад, InnoDB довго розроблявся окремою компанією.

MySQL побудована за клієнт-серверною технологією: клієнт та сервер є програмами, які можуть бути розташовані на різних комп’ютерах або на одному.

Сервер зберігає та обслуговує базу даних, клієнти надсилають базі даних запити на мові SQL і отримують у відповідь таблиці з даними.

2.2. Клієнти MySQL

За допомогою консольної утиліти (терміналу), спеціалізованих комп’ютерних програм з графічним інтерфейсом, а також за допомогою програмного коду можно звертатися до MySQL сервера.

Наприклад, за допомогою СКБД DBeaver можливо підключатися до різних баз даних, в тому числі MySQL.

Зручність подібних рішень полягає в наявності графічного інтерфейсу, який мінімізує необхідність використання команд за допомогою консольної утиліти.

Але, можуть виникнути завдання, які простіше і швидше виконати за допомогою відповідних команд. Наприклад, необхідно зняти резервну копію з бази даних або розгорнути її на іншому сервері – для найшвидшого виконання цього завдання доцільно використати утиліту `mysql` з дистрибутива MySQL.

У даному навчальному посібнику інсталяцію MySQL буде продемонстровано із використанням `Docker` контейнеру.

Для цього необхідно:

1) завантажити, інсталювати та запустити `Docker`;

2) створити окрему теку та файл в ній під назвою `docker-compose.yml`;

3) у файлі `docker-compose.yml` написати наступні команди:

```
version: '3.9'
services:
  db:
    image: mysql:8.3.0
    container_name: mysql-db
    environment:
      MYSQL_ROOT_PASSWORD: strong_password
      MYSQL_DATABASE: example_db
      MYSQL_USER: user
      MYSQL_PASSWORD: user_password
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql
      - ./mysql-config:/etc/mysql/conf.d
volumes:
  db_data:
```

4) у консольної утиліті (терміналі) виконати команду

```
docker-compose up -d
```

5) відкрити Docker, перейти в розділ “Containers” та відкрити контейнер mysql-db (рис. 3);

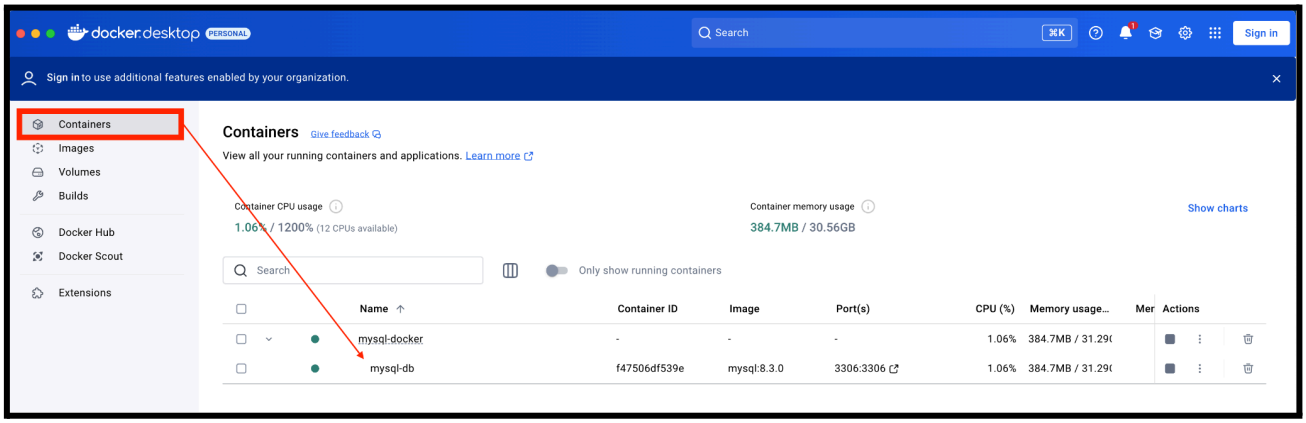


Рис. 3.

6) перейти у розділ “Exec” та для з’єднання з сервером в контейнері Docker виконати команду mysql -u root -p

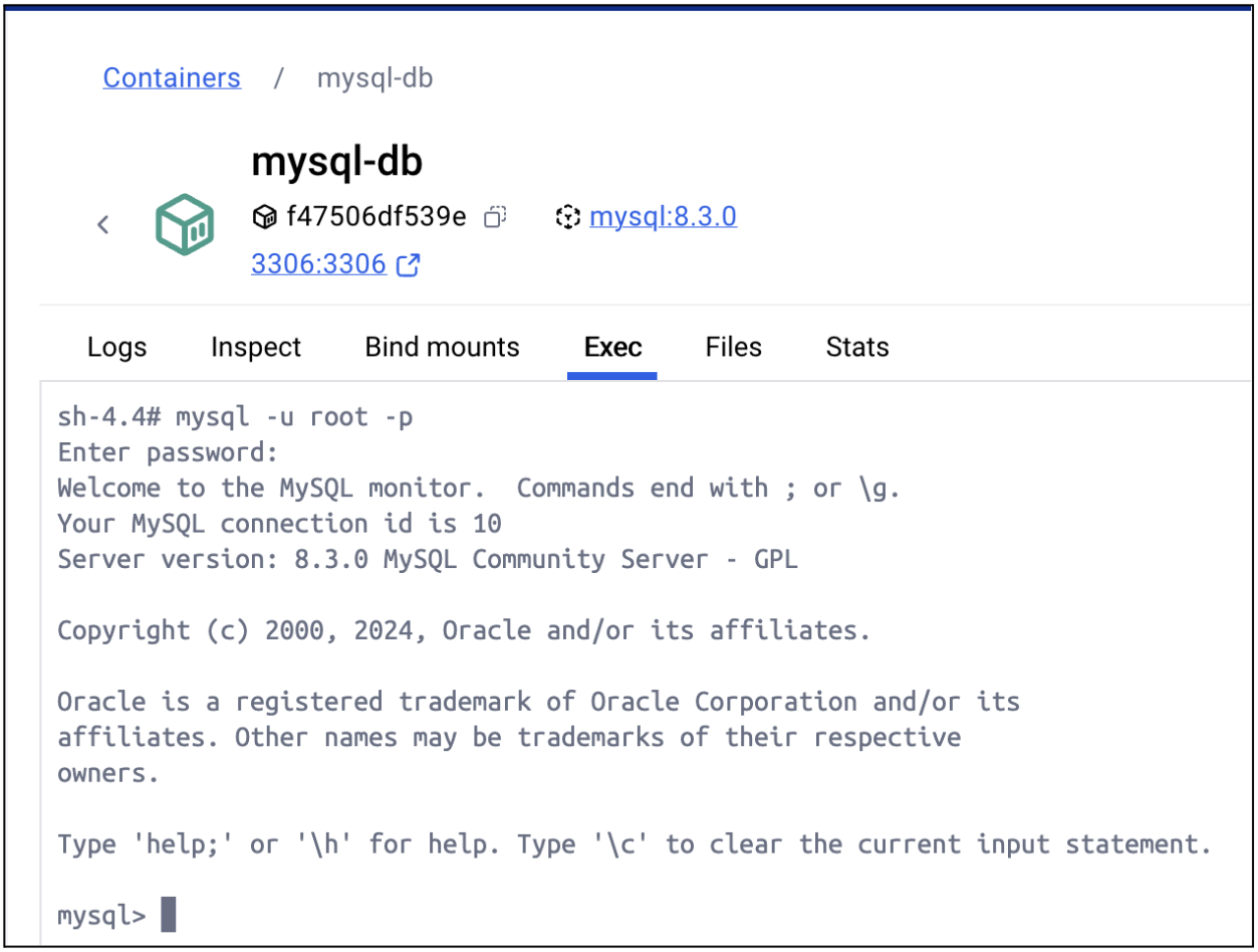


Рис. 4.

Після аргумента `-u` необхідно казати ім’я користувача MySQL, аргумент `-p` необхідно вказати для того, щоб в подальшому ввести пароль. У файлі `docker-compose.yml` було вказано пароль `strong_password`. Після інсталяції MySQL користувач з адміністративними правами має назву `root`.

Після вищезазначеного існує можливість набирати команди. Наприклад, для виведення фрази “Hello world!” необхідно написати наступну команду (рис. 5):

```
SELECT "Hello world!";
```



Рис. 5.

Будь-яку написану команду необхідно завершити крапкою з комою. Якщо команда не поміщається в одному рядку, то можно перейти на наступний – для цього необхідно натиснути клавішу `Enter`. Для відправки команди на сервер MySQL необхідно в кінці написати крапку з комою.



Рис. 6.

У першому рядку таблиці, яка була згенерована у результаті надсилання SQL запиту (рис. 6), містяться заголовки стовпців, а в рядках – відповідь сервера на запит.

Як правило заголовками є імена, які записані в таблицях бази даних. Якщо в SQL запиті не вказується назва конкретного стовпця, а лише фраза (у наведеному вище прикладі це “Hello world!”), то назвою стовпця виступатиме ця фраза. Після цього вказується кількість повернутих рядків (1 row in set – один рядок у результаті) та час виконання запиту.

Для надсилання запитів можна використовувати будь-який регістр символів, наприклад **select** можна набрати маленькими літерами:

```
select "Hello world!";
```

Для повторного використання раніше введених запитів можно використати клавіші <↑> та <↓> (стрілка вгору та стрілка вниз).

Якщо сервер встановлено на віддаленому хості, можна з'єднатися з ним, вказавши IP-адресу або домен віддаленого сервера у параметрі **-h**, а у параметрі **-P** ми можемо вказати порт.

```
mysql -u root -h 192.168.0.10 -P 3306
```

Утиліта **mysql** підтримує різні команди, що представлені в таблиці нижче.

Команда	**Скорочення**	**Опис**
USE	\u	вибір бази даних
SOURCE	\.	виконання SQL-команди з файлу
SYSTEM	\!	виконання команди операційної системи
STATUS	\s	виведення інформації про стан сервера
EXIT	\q	вихід
	\G	виведення результату у вертикальному форматі

Іноді таблиця, яка була згенерована у результаті надсилання SQL запиту, містить занадто велику кількість стовпців, у результаті чого структура таблиці порушується та сприймати інформацію практично неможливо.

Так, у цьому випадку зручно скористатися спеціалізованою комп'ютерною програмою з графічним інтерфейсом, але це не завжди можливо. Вихід із ситуації – використання так званого вертикального режиму виведення:

```
SELECT * FROM exchange_rates \G;
```

```
mysql> SELECT * FROM exchange_rates;
+----+---------------+---------------+---------+
| id | exchange_date | currency_code | rate    |
+----+---------------+---------------+---------+
|  1 | 2022-11-25    | USD           | 36.5686 |
|  2 | 2022-11-25    | EUR           | 38.0716 |
|  3 | 2022-12-27    | USD           | 36.5686 |
|  4 | 2022-12-27    | EUR           | 38.8432 |
|  5 | 2022-12-27    | SEK           |  3.4910 |
|  6 | 2022-04-26    | USD           | 29.2549 |
|  7 | 2022-04-26    | PLN           |  6.7749 |
|  8 | 2022-04-28    | USD           | 29.2549 |
+----+---------------+---------------+---------+
8 rows in set (0.00 sec)
```

Рис. 7.

```
mysql> SELECT * FROM exchange_rates \G;
*************************** 1. row ***************************
          id: 1
exchange_date: 2022-11-25
currency_code: USD
        rate: 36.5686
*************************** 2. row ***************************
          id: 2
exchange_date: 2022-11-25
currency_code: EUR
        rate: 38.0716
*************************** 3. row ***************************
          id: 3
exchange_date: 2022-12-27
currency_code: USD
        rate: 36.5686
*************************** 4. row ***************************
          id: 4
exchange_date: 2022-12-27
currency_code: EUR
        rate: 38.8432
*************************** 5. row ***************************
          id: 5
exchange_date: 2022-12-27
currency_code: SEK
        rate: 3.4910
*************************** 6. row ***************************
```

Рис. 8.

У такому режимі кожен рядок виводиться як окрема таблиця, де спочатку вказується назва стовпця, а після – значення для цього рядка.

Для цього використовується внутрішня команда `\G` утиліти **mysql**.

Для отримання поточного статусу сервера можно використати команду `STATUS` (рис. 9). Ця команда має скорочений варіант `\s`.

```
mysql> STATUS;
--------------
mysql  Ver 8.3.0 for Linux on aarch64 (MySQL Community Server - GPL)

Connection id:          10
Current database:       currency_exchange
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         8.3.0 MySQL Community Server - GPL
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    utf8mb4
Db     characterset:    utf8mb4
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /var/run/mysqld/mysqld.sock
Binary data as:         Hexadecimal
Uptime:                 45 min 39 sec

Threads: 2  Questions: 26  Slow queries: 0  Opens: 142  Flush tables: 3  Open tables: 61  Queries per second avg: 0.009
--------------
```

Рис. 9.

Виконати системну команду операційної системи можна за допомогою **SYSTEM**. Нижченаведена команда відображає список файлів у поточній директорії (рис. 10):

```
SYSTEM ls;
```

```
mysql> SYSTEM ls;
bin  boot  dev  docker-entrypoint-initdb.d  etc  home  lib
```

Рис. 10.

Для команди **SYSTEM** можно використати скорочення **\!** (рис. 11):

```
\! ls;
```

```
mysql> \! ls;
bin  boot  dev  docker-entrypoint-initdb.d  etc  home  lib
```

Рис. 11.

Для зручності роботи із командами та для їх повторного використання можно створювати відповідні файли. Наприклад, за допомогою команд утиліти mysql створимо файл `hello.sql`, у якому написано “`SELECT Hello world!`” (мал. 12):

```
\! echo "SELECT 'Hello world!'" > hello.sql;
```

```
mysql> \! echo "SELECT 'Hello world!'" > hello.sql;
mysql> \! cat hello.sql;
SELECT 'Hello world!'
```

Рис. 12.

Після того як файл `hello.sql` створено, можно виконати команду `SOURCE` (мал. 13):

```
SOURCE hello.sql;
```

```
mysql> SOURCE hello.sql;
+--------------+
| Hello world! |
+--------------+
| Hello world! |
+--------------+
1 row in set (0.00 sec)
```

Рис. 13.

Такий підхід – запис відповідних команд у файл та її виконання за допомогою `SOURCE` – дозволяє виявляти помилки та швидко їх виправляти.

2.3. Керування базою даних

Створення бази даних засобами SQL здійснюється за допомогою команди: `CREATE DATABASE`:

```
CREATE DATABASE currency_exchange;
```

Щоб переглянути список існуючих баз даних можно використати команду `SHOW DATABASES` (рис. 14):

```
SHOW DATABASES;
```

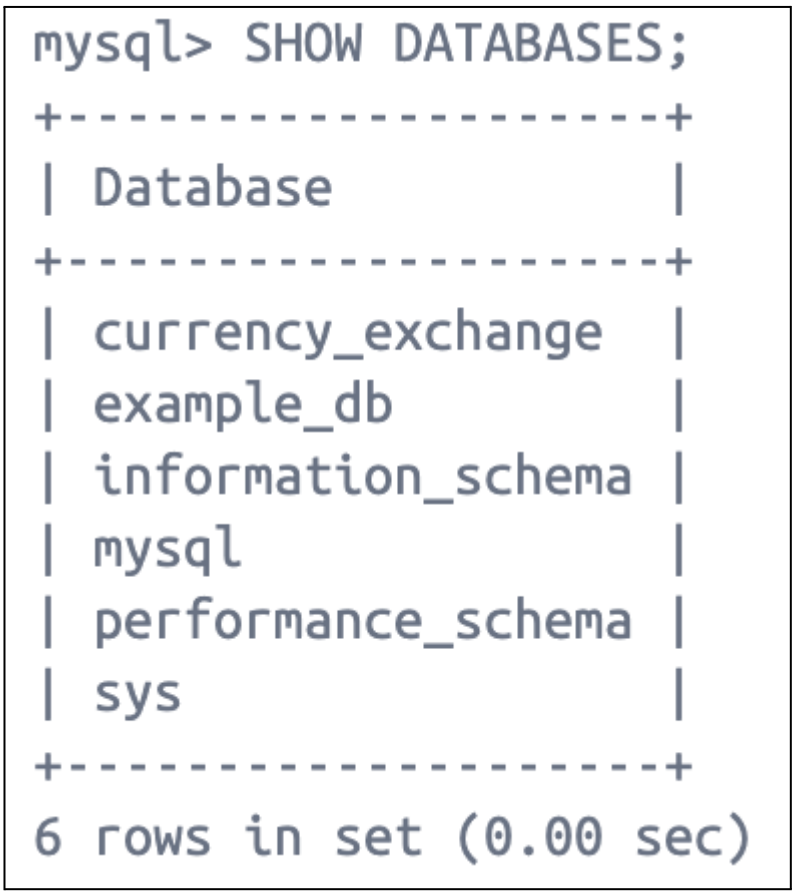


Рис. 14.

За результатами виконання команди SHOW DATABASES відображається список баз даних. Окрім створеної бази даних currency_exchange, також можно побачити системні та навчальні бази даних.

Бази даних performance_schema та sys призначені для дослідження параметрів продуктивності, information_schema – для опису структури бази даних.

Видалити базу даних можно за допомогою виконання команди DROP DATABASE:

```
DROP DATABASE currency_exchange;
SHOW DATABASES;
```

При спробі створити вже існуючу базу даних виникатиме помилка:

```
CREATE DATABASE currency_exchange;
ERROR 1007 (HY000): Can't create database 'currency_exchange'; database exists
```

Для запобігання вищевказаній помилці команду CREATE DATABASE можна доповнити конструкцією IF NOT EXISTS, яка дозволить створити базу даних, якщо вона не існує, а також у разі існування бази даних – ніяких дій не проводити:

```
DROP DATABASE currency_exchange;
Query OK, 0 rows affected (0.00 sec)

CREATE DATABASE IF NOT EXISTS currency_exchange;
Query OK, 1 row affected (0.01 sec)

CREATE DATABASE IF NOT EXISTS currency_exchange;
```

```
Query OK, 1 row affected, 1 warning (0.00 sec)
```

Для початку роботи з таблицями бази даних, необхідно вибрати відповідну базу даних. Наприклад, за допомогою утиліти **mysql** вибрати нову базу даних можна за допомогою `USE`:

```
USE currency_exchange;
```

Якщо поточну базу даних не обрано, то виконання певних команд може викликати помилку – `No database selected`:

```
SHOW TABLES;
ERROR 1046 (3D000): No database selected
```

Можна не використовувати команду `USE` та не обирати поточну базу даних, але у цьому випадку у всіх командах доведеться явно вказувати, яка база даних має використовуватись:

```
SHOW TABLES FROM currency_exchange;
```

2.4. Порядок створення таблиць в MySQL

Створити таблицю всередині бази даних можна за допомогою команди `CREATE TABLE`:

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] users
[(create_definition,...)]
[table_options] [select_statement]
```

Починається команда ключовим словом `CREATE TABLE`, за яким необхідно вказати ім’я таблиці, а в круглих дужках – назви стовпців, тип даних стовпця та інші параметри за необхідності:

```
CREATE TABLE exchange_rates (
    id INT PRIMARY KEY,
    exchange_date DATE,
    currency_code VARCHAR(3),
    rate DECIMAL(10, 4)
);
```

Повторне створення таблиці призводить до помилки `ERROR 1050 (42S01): Table 'exchange_rates' already exists`.

Якщо додати конструкцію `IF NOT EXISTS`, то помилка виникати не буде:

```
CREATE TABLE IF NOT EXISTS exchange_rates (
    id INT PRIMARY KEY,
    exchange_date DATE,
    currency_code VARCHAR(3),
    rate DECIMAL(10, 4)
);
```

Використання конструкції `IF NOT EXISTS` доцільно у разі виконання великої кількості команд, записаних у файл. Виникнення помилки на певному етапі може зупинити виконання інших записаних у файл команд.

Для перегляду структури таблиці можна використати команду `DESCRIBE` (рис. 15):

```
DESCRIBE exchange_rates;
```

```
mysql> DESCRIBE exchange_rates;
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| id            | int           | NO   | PRI | NULL    |       |
| exchange_date | date          | YES  |     | NULL    |       |
| currency_code | varchar(3)    | YES  |     | NULL    |       |
| rate          | decimal(10,4) | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Рис. 15.

Кожен рядок у таблиці (мал. 15) відповідає окремому стовпцю.

Замість імені стовпця можна використовувати LIKE шаблон із застосуванням спецсимволів _ (означає будь-який один символ) та **%** (означає будь-яку кількість символів):

```
DESCRIBE exchange_rates '%a%';
```

```
mysql> DESCRIBE exchange_rates '%a%';
+---------------+---------------+------+-----+---------+-------+
| Field         | Type          | Null | Key | Default | Extra |
+---------------+---------------+------+-----+---------+-------+
| exchange_date | date          | YES  |     | NULL    |       |
| rate          | decimal(10,4) | YES  |     | NULL    |       |
+---------------+---------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Рис. 16.

Завдання для самоконтролю

1. Особливості архітектури MySQL.
2. Загальний порядок інсталяції та запуску MySQL в Docker контейнері.
3. Що таке файл `docker-compose.yml`?
4. Порядок підключення до серверу MySQL.

5. Порядок написання та запуску команд в MySQL.
6. Команда `USE`.
7. Команда `SOURCE`.
8. Команда `SYSTEM`.
9. Команда `STATUS`.
10. Команда `EXIT`.
11. Порядок створення та видалення бази даних.
12. Порядок створення таблиць бази даних.

3. МОВА СТРУКТУРОВАНИХ ЗАПИТІВ (SQL)

3.1. Поняття мови структурованих запитів (SQL)

Ієрархічні та мережеві бази даних часто не мали власної мови запитів за допомогою якої можно було б в інтерактивному режимі спілкуватися із базою даних. Для цього доводилось писати авторські програми.

Одними із перших, де реалізували мову запитів SQL, стали реляційні бази даних.

SQL зародився в компанії IBM і спочатку називався SEQUEL (Structured English Query Language), проте торгова марка була зайнята і назву мови довелося скоротити до SQL (Structured Query Language).

На сьогодні за допомогою SQL можно швидко обробляти та аналізувати великі масиви даних (Big Data).

SQL – це декларативна, а не процедурна мова, тобто за допомогою SQL користувач описує мету та кінцевий результат, який бажає отримати, а не інструкцію щодо виконання алгоритму.

Синтаксис SQL схожий на звичайні слова англійської мови, що спрощує вивчення та розуміння даної мови.

3.2. Коментарі в SQL

SQL за необхідності можно коментувати. Однорядкові коментарі починаються із двох символів дефісу:

```
-- SELECT "Hello world!";
```

Можна використовувати багаторядкові коментарі, які починаються з послідовності `/*`, а завершуються – `*/`

```
/* Це коментар
SELECT "Hello world!"; */
```

3.3. DDL та DML

Інструкції SQL можна розділити на дві великі групи: мову опису даних (DDL) і мову управління даними (DML).

До першої групи (DDL) відносяться інструкції створення, видалення та редагування баз даних та таблиць.

До другої (DML) запити на створення, редагування, видалення та вилучення даних з таблиць.

3.4. Структура SQL

Кожна інструкція SQL починається з певного ключового слова, яке описує подальшу операцію. В інструкції можуть міститися ключові слова `WHERE`, `FROM`, а також імена таблиць або стовпців, інші додаткові ключові слова та скалярні вирази.

Скалярні вирази – це числа та текст. Іншими словами це константи.

```
SELECT "Hello world!";
```

У запиті вище рядок `"Hello world!"` – це скалярний вираз. Число 2 теж є скалярним виразом.

```
SELECT 2 + 2;
```

3.5. Типи даних в SQL

MySQL підтримує кілька типів даних, які можна розбити на п’ять груп:

– числові дані;

– текстові дані;

– спеціальний тип `NULL`, що позначає пусті значення, тобто відсутність будь-якої інформації;

– календарні дані призначені для збереження дати та часу;

– колекційні типи дозволяють зберігати безліч значень або навіть цілі документи в вигляді JSON.

Типи визначають певні характеристики (атрибути) значень, що зберігаються, а також об’єм пам’яті, яку необхідно виділити для їхнього зберігання.

Наприклад, такі атрибути як `NULL` або `NOT NULL` обмежують значення для полів стовпця, надаючи можливість зберігати пусті значення або навпаки заборону на зберігання таких значень.

Атрибут `DEFAULT` дозволяє задати значення поля за замовчуванням.

Атрибут `UNSIGNED` забороняє зберігати від’ємні числові значення.

3.5.1. Числовий тип даних

Числові типи можна розділити на:

1. `TINYINT`;
2. `BOOL` або `BOOLEAN`;
3. `SMALLINT`;
4. `MEDIUMINT`;
5. `INT`;
6. `BIGINT`;
7. `DECIMAL`;

8. `FLOAT`;
9. `DOUBLE`;
10. `SERIAL`.

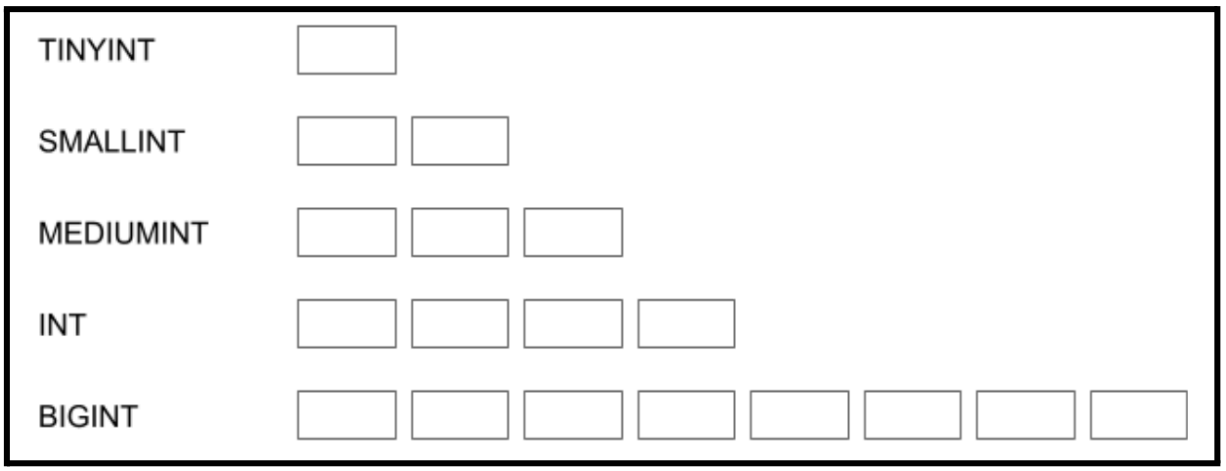


Рис. 17.

Прямокутниками на рис. 17 продемонстровано кількість байт, що відводяться для зберігання числового типу даних. Чим більший об'єм пам'яті займає тип, то тим більший об'єм жорсткого диску та оперативної пам'яті займатиме таблиця з даними.

`TINYINT` займає один байт, тобто 8 біт, та дозволяє записувати цілі числові дані в діапазоні від -128 до 127. Використання атрибуту `UNSIGNED` змінює діапазон від 0 до 255.

`BOOL` або `BOOLEAN` призначений для запису `1` або `0`, тобто констант `True` або `False`.

`SMALLINT` займає два байти та дозволяє записувати цілі числа у діапазоні від -32768 до 32767. Використання атрибуту `UNSIGNED` змінює діапазон від 0 до 65535.

`MEDIUMINT` – запис цілих числових даних в діапазоні від -8388608 до 8388607. Використання атрибуту `UNSIGNED` змінює діапазон від 0 до 16777215.

`INT` займає чотири байти та дозволяє записувати цілі числові дані в діапазоні від -2147483648 до 2147483647. Використання атрибуту `UNSIGNED` змінює діапазон від 0 до 4294967295.

`BIGINT` займає вісім байт та дозволяє записувати цілі числові дані в діапазоні від -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807. Використання атрибуту `UNSIGNED` змінює діапазон від 0 до 18 446 744 073 709 551 615.

3.5.2. Текстовий тип даних

Текстовий тип даних можна умовно розділити на:

1. `CHAR`;
2. `VARCHAR`;
3. `TINYTEXT`;
4. `TEXT`;
5. `MEDIUMTEXT`;
6. `LONGTEXT`.

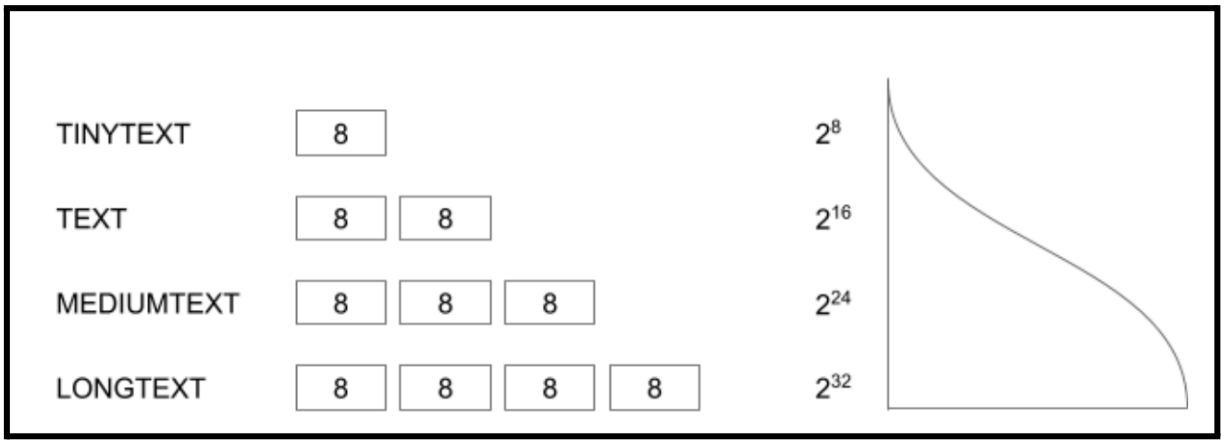


Рис. 18.

`CHAR` займає один байт для `CHAR(1)`, тобто для одного символу. Цей тип даних працює наступним чином – необхідно явно вказати кількість символів для запису і відповідний об’єм пам’яті буде виділятися незважаючи на те чи досягнуто граничну кількість символів заданих для цього типу даних.

`VARCHAR` немає фіксованого розміру, тобто динамічно виділяється пам’ять для зберігання даних залежно від кількості символів. Але, у разі необхідності можно задати максимальну кількість символів.

3.5.3. Тип NULL

Спеціальний тип `NULL` позначає пусті значення, тобто відсутність будь-якої інформації.

```
SELECT NULL;
```

Усі операції з `NULL` повертаються значення `NULL` (рис. 19-20).

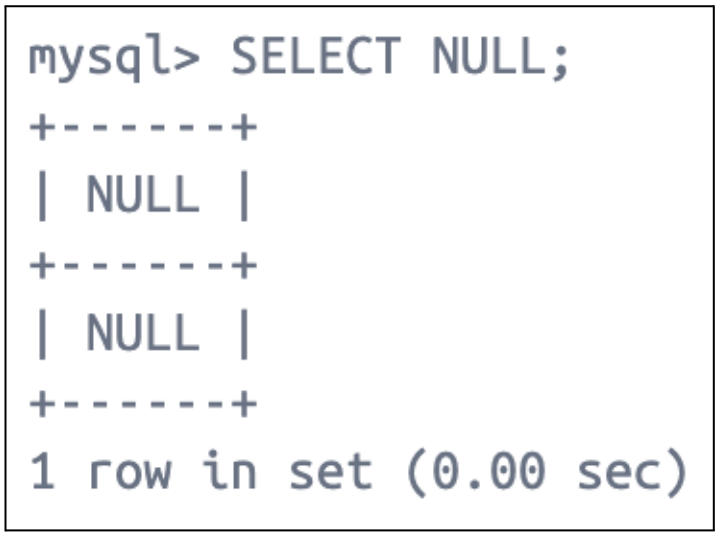


Рис. 19.

```
SELECT NULL + 2;
```

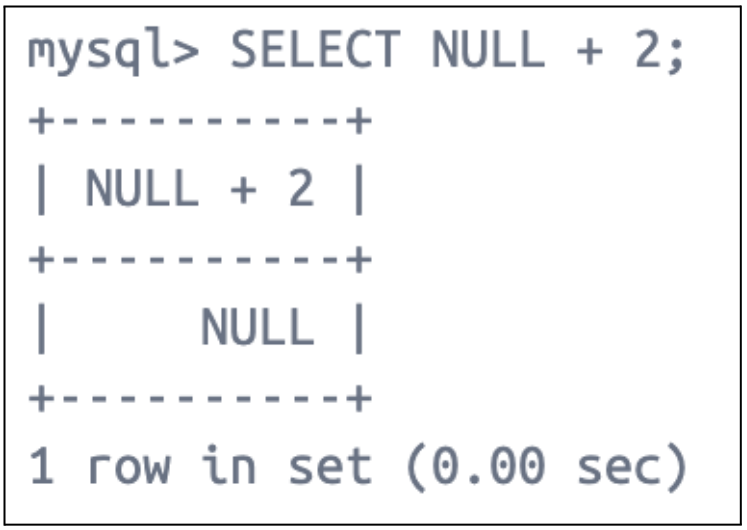


Рис. 20.

На прикладі створення пустого стовпця продемонструємо порядок використання `NULL` (рис. 21).

```
CREATE TABLE tbl (id INT UNSIGNED);
INSERT INTO tbl VALUES();
SELECT * FROM tbl;
```

```
mysql> CREATE TABLE tbl (id INT UNSIGNED);
Query OK, 0 rows affected (0.01 sec)

mysql> INSERT INTO tbl VALUES();
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM tbl;
+------+
| id   |
+------+
| NULL |
+------+
1 row in set (0.00 sec)
```

Рис. 21.

Можно заборонити вставляти `NULL` в поле – для цього необхідно задати атрибут `NOT NULL`.

За допомогою оператора `ALTER TABLE` реалізуємо відповідні зміни в таблицю `tbl` (рис. 22).

```
ALTER TABLE tbl CHANGE id id INT UNSIGNED NOT NULL;
```

Ми можемо заборонити вставляти в поле **NULL** значення – для цього стовпець слід забезпечити атрибутом **NOT NULL**. Давайте для різноманітності не видалятимемо таблицю, а спробуємо її перетворити

```
ALTER TABLE tbl CHANGE id id INT UNSIGNED NOT NULL;
ERROR 1138 (22004): Invalid use of NULL value
```

Ми отримуємо помилку, так як у нас є запис з **NULL**, можна скористатися оператором **TRUNCATE**.

```
TRUNCATE tbl;
```

Повторюємо виклик **ALTER TABLE**:

```
ALTER TABLE tbl CHANGE id id INT UNSIGNED NOT NULL;
```

І дивимося структуру таблиці:

```
mysql> DESCRIBE tbl\G
*************************** 1. row ***************************
Field: id
Type: int(10) unsigned
Null: NO
Key:
Default: NULL
Extra:
1 row in set (0,00 sec)
```

Зверніть увагу: поле Null набуває значення NO.

```
INSERT INTO tbl VALUES();
```

Тепер вставити значення NULL у таблицю не вдасться.

3.5.4. Календарний тип даних

MySQL підтримує п'ять типів календарних типів:

– **TIME** призначений для зберігання часу протягом доби;

– **YEAR** зберігає рік;

– **DATE** зберігає дату з точністю до дня;

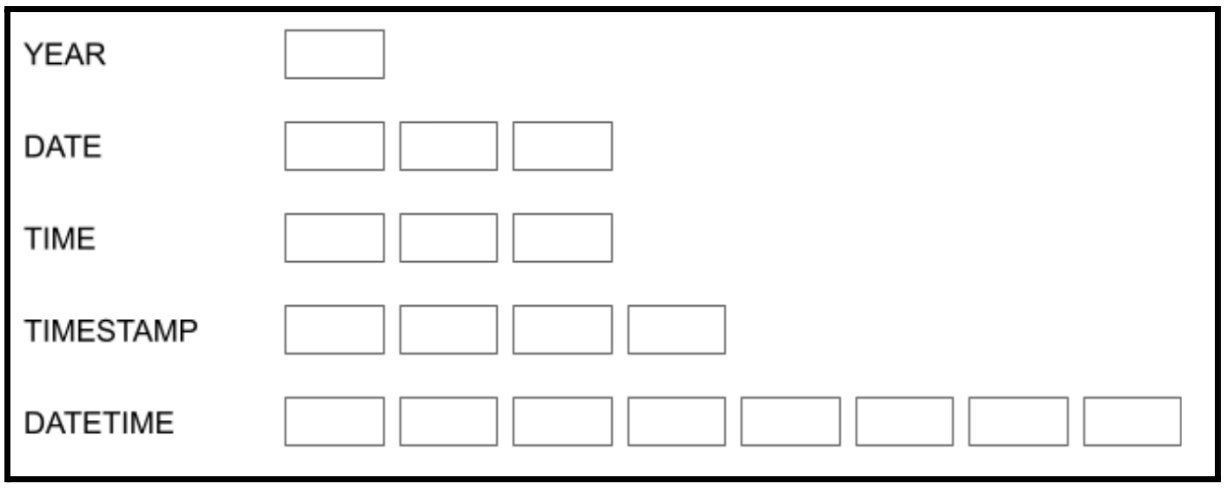
– **DATETIME** зберігає дату та час;

– **TIMESTAMP** також зберігає дату та час, займає вдвічі менше місця, ніж **DATETIME**, але може зберігати лише обмежені дати – в інтервалі від 1970 року до 2038.

Крім того, перший TIMESTAMP-стовпець у таблиці оновлюється автоматично під час операцій створення та оновлення. **TIMESTAMP** зберігає дату в форматі UTC. У таблиці представлені календарні типи, що у таких форматах MySQL повертає календарні значення:

Тип	**Опис**
YEAR	0000
DATE	'0000-00-00'
TIME	'00:00:00'
DATETIME	'0000-00-00 00:00:00'
TIMESTAMP	'0000-00-00 00:00:00'

На малюнку прямокутниками представлена кількість байт, які відводяться під поле кожного з календарні типи:



Якщо ви хочете задати дату, зробити це можна за допомогою наступного виразу:

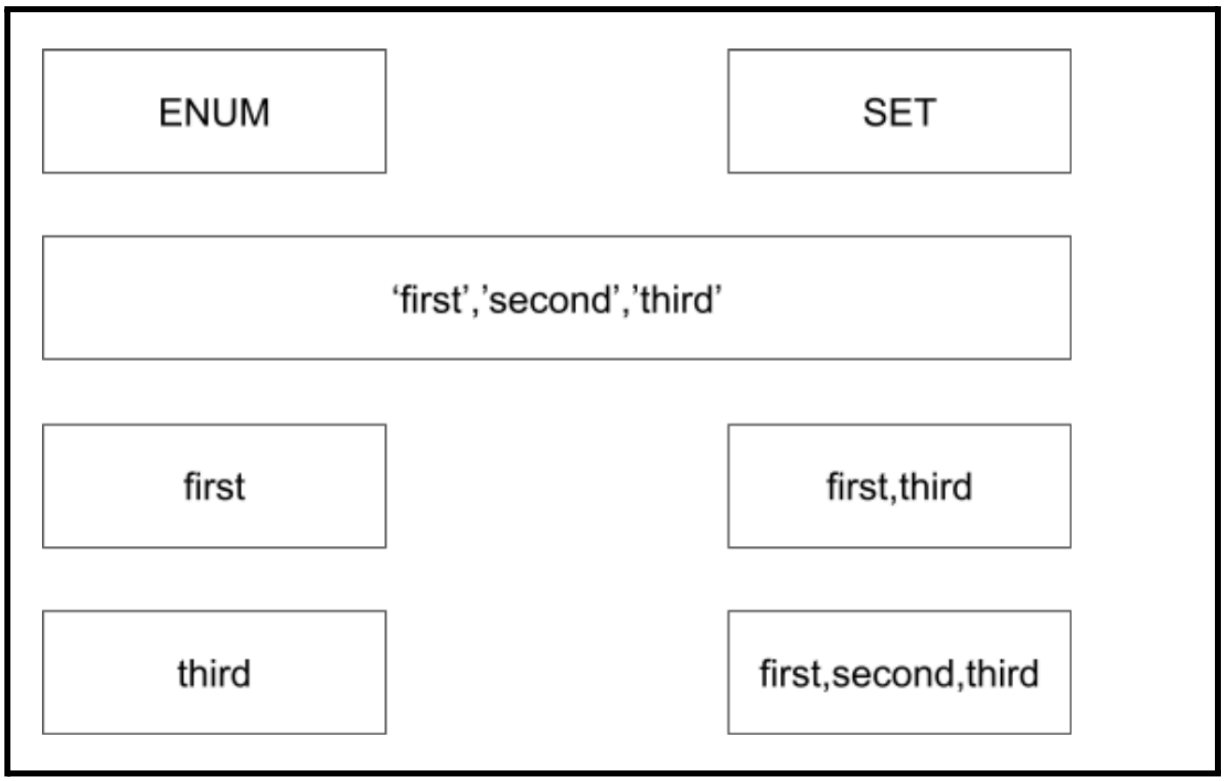
```
SELECT '2018-10-01 0:00:00';
```

З календарними типами можна проводити операції складання та віднімання. Для цього використовується спеціальна конструкція **INTERVAL**:

```
SELECT '2018-10-01 0:00:00' - INTERVAL 1 DAY;
SELECT '2018-10-01 0:00:00' + INTERVAL 1 WEEK;
SELECT '2018-10-01 0:00:00' + INTERVAL 1 YEAR;
SELECT '2018-10-01 0:00:00' + INTERVAL '1-1' YEAR_MONTH;
```

3.5.5. Колекційні типи даних

При оголошенні списку допустимих значень **ENUM** та **SET** задаються списком рядків, але у внутрішньому поданні бази даних елементи множин зберігаються у вигляді чисел. У випадку **ENUM** поле може набувати лише одне значення зі списку. У разі **SET** комбінацію заданих значень.



Останнім часом більшої популярності набув формат **JSON**, готовий об'єкт мови JavaScript. Цей формат інтенсивно використовується для зберігання та передачі колекцій. У MySQL передбачено стовпець JSON-формату. Давайте додамо до таблиці tbl ще один стовпець JSON-типу:

```
DESCRIBE tbl;
ALTER TABLE tbl ADD collect JSON;
DESCRIBE tbl;
```

Давайте вставимо в це поле JSON-об'єкт:

```
INSERT INTO tbl VALUES(1,
'{"first": "Hello"
SELECT * FROM tbl;
SELECT collect->>"$.first" FROM tbl;
SELECT collect->>"$.second" FROM tbl;
```

Завдання для самоконтролю

1. Що таке мова структурованих запитів (SQL)?
2. Правила створення коментарів в SQL.
3. Мова опису даних (DDL).
4. Мова управління даними (DML).
5. Структура SQL.
6. Числовий тип даних в SQL.
7. Текстовий тип даних в SQL.
8. Тип даних NULL в SQL.
9. Календарний тип даних в SQL.
10. Колекційні тип даних в SQL.

4. Індекси та ключи

4.1. Загальне поняття індексів та ключів

Ключі – це стовпці, за допомогою яких ми досягаємо унікальності записів та пов'язуємо записи у різних таблицях.

Ключі дуже часто постачаються індексами, тому в масовій свідомості розробників баз даних часто зливаються в одне поняття. Тим не менш, індексувати можна не тільки стовпці з ключами, але й будь-який стовпець чи комбінацію стовпців таблиці.

Зазвичай записи у таблиці розташовуються у хаотичному порядку. Щоб знайти потрібний запис, необхідно сканувати всю таблицю, потім йде багато часу. Ідея індексів полягає в тому, щоб створити копію стовпця, яка постійно підтримуватиметься у відсортованому стані.

Це дозволяє дуже швидко здійснювати пошук за таким стовпцем, тому що заздалегідь відомо, де необхідно шукати значення.

Зворотний бік медалі – додавання або видалення запису потребує додаткового часу на сортування стовпця. Крім того, створення копії збільшує обсяг пам'яті, необхідний для розміщення таблиці на жорсткому диску та в оперативній пам'яті сервера.

Саме тому СУБД не створює індекси для кожного стовпця та їх комбінацій, а віддає це на відкуп розробникам.

4.2. Види індексів

Існує кілька видів індексів:

● звичайний індекс – таких індексів у таблиці може бути декілька;

● унікальний індекс — унікальних індексів також може бути декілька, але значення в ньому не повинні повторюватися;

● первинний ключ — унікальний індекс, призначений для первинного ключа таблиці. У таблиці може бути лише один первинний ключ;

● повнотекстовий індекс — спеціальний вид індексу для стовпців типу TEXT, що дозволяє проводити повнотекстовий пошук. Розглядати його не будемо, тому що на практиці завдання повнотекстового пошуку здійснюється спеціалізованими базами даних, такими як ElasticSearch.

Первинний ключ таблиці позначається спеціальним індексом **PRIMARY KEY**. Значення первинного ключа має бути унікальним і не повторюватися в межах таблиці. Крім того, стовпці, позначені атрибутом **PRIMARY KEY**, не можуть набувати значення **NULL**. Для позначки поля таблиці як первинний ключ достатньо помістити ключове слово **PRIMARY KEY** в визначення стовпця.

Позначимо поле **id** таблиці **catalogs** атрибутом **PRIMARY KEY**:

```
DROP TABLE IF EXISTS catalogs;
CREATE TABLE catalogs (
id INT UNSIGNED NOT NULL PRIMARY KEY,
name VARCHAR(255) COMMENT 'Назва розділу'
) COMMENT = 'Розділ веб-сайту';
DESCRIBE catalogs;
```

Є альтернативний спосіб оголошення первинного ключа – окремим записом **PRIMARY KEY** з зазначенням назви стовпця у круглих дужках:

```
DROP TABLE IF EXISTS catalogs;
CREATE TABLE catalogs (
id INT UNSIGNED NOT NULL,
name VARCHAR(255) COMMENT 'Назва розділу'
PRIMARY KEY(id)
) COMMENT = 'Розділ веб-сайту';
```

Ключове слово **PRIMARY KEY** може зустрічатися в таблиці лише один раз, оскільки у таблиці дозволено лише один первинний ключ. Індекс необов’язково має бути оголошений по одному стовпцю, цілком допустимо оголошення індексу відразу по двох або більше стовпців.

```
CREATE TABLE catalogs (
  id INT UNSIGNED NOT NULL,
  name VARCHAR(255) COMMENT 'Назва розділу',
  PRIMARY KEY(id, name(10))
) COMMENT = 'Розділ веб-сайту';
DESCRIBE tbl;
```

Тут первинний ключ створюється по стовпцю **id** і першим 10 символам стовпця **name**. Можна індексувати за всіма 255 символами текстового стовпця. Однак розмір індексу буде більшим, як правило, для індексу достатньо перших символів рядка.

Як первинний ключ часто виступає цілісний стовпець. Такий вибір пов'язаний з тим, що цілочисленні типи даних обробляються найшвидше і займають невеликий обсяг.

Інша причина вибору такого типу – тільки цей тип стовпця може бути забезпечений атрибутом **AUTO_INCREMENT**, що забезпечує автоматичне створення унікального індексу.

```
DROP TABLE IF EXISTS catalogs;
CREATE TABLE catalogs (
  id INT UNSIGNED NOT NULL PRIMARY KEY AUTO_INCREMENT,
  name VARCHAR(255) COMMENT 'Назва розділу'
) COMMENT = 'Розділ веб-сайту';
SELECT * FROM catalogs;
```

Передача стовпцю, з цим атрибутом, значення **NULL** або 0, призводить до автоматичного надання йому максимального значення стовпця, плюс 1.

```
INSERT INTO catalogs (name) VALUES ('Процесори');
INSERT INTO catalogs VALUES (0, 'Мат. плати');
INSERT INTO catalogs VALUES (NULL, 'Видеокарти');
SELECT * FROM catalogs;
```

Це зручний механізм, який дозволяє не дбати про генерацію унікального значення засобами прикладної програми, що працює із СУБД MySQL.

MySQL пропонує псевдотип **SERIAL**, який є позначенням типу **BIGINT**, забезпеченого додатковими атрибутами **UNSIGNED**, **NOT NULL**, **AUTO_INCREMENT** та унікальним індексом.

```
SERIAL == BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
```

```
DROP TABLE IF EXISTS catalogs;
CREATE TABLE catalogs (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) COMMENT 'Назва розділу'
) COMMENT = 'Розділ веб-сайту';
```

Так набагато компактніше, а головне – сумісне з іншими базами даних, які часто також підтримують псевдотип **SERIAL**.

На відміну від первинного ключа, таблиця може містити кілька звичайних та унікальних індексів. Щоб було зручно розрізняти їх, індекси можуть мати власні імена. Часто імена індекси збігаються з іменами стовпців, які вони індексують, але для індексу можна призначити і зовсім інше ім'я. Оголошення індексу здійснюється за допомогою ключового слова **INDEX** чи **KEY**.

Для унікальних індексів запроваджується додаткове ключове слово **UNIQUE**. Давайте у таблиці **products** забезпечимо індексом поле **catalog_id**.

```
DROP TABLE IF EXISTS products;
CREATE TABLE products (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) COMMENT 'Назва',
  description TEXT COMMENT 'Опис',
  price DECIMAL (11,2) COMMENT 'Ціна',
  catalog_id INT UNSIGNED,
  created_at DATETIME DEFAULT CURRENT TIMESTAMP,
   updated_at   DATETIME   DEFAULT   CURRENT_TIMESTAMP   ON   UPDATE
CURRENT_TIMESTAMP,
  KEY index_of_catalog_id(catalog_id)
) COMMENT = 'Позиції';

DESCRIBE products;
```

Створити індекс у вже існуючій таблиці можна за допомогою оператора **CREATE INDEX**

```
DROP TABLE IF EXISTS products;
CREATE TABLE products (
  id SERIAL PRIMARY KEY,
  name VARCHAR(255) COMMENT 'Назва',
  description TEXT COMMENT 'Опис',
  price DECIMAL (11,2) COMMENT 'Ціна',
  catalog_id INT UNSIGNED,
  created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
```

```
   updated_at   DATETIME   DEFAULT   CURRENT_TIMESTAMP   ON   UPDATE
CURRENT_TIMESTAMP
) COMMENT = 'Позиції';
CREATE INDEX index_of_catalog ON products (catalog_id);
```

Видалити індекс із таблиці можна за допомогою оператора **DROP INDEX**:

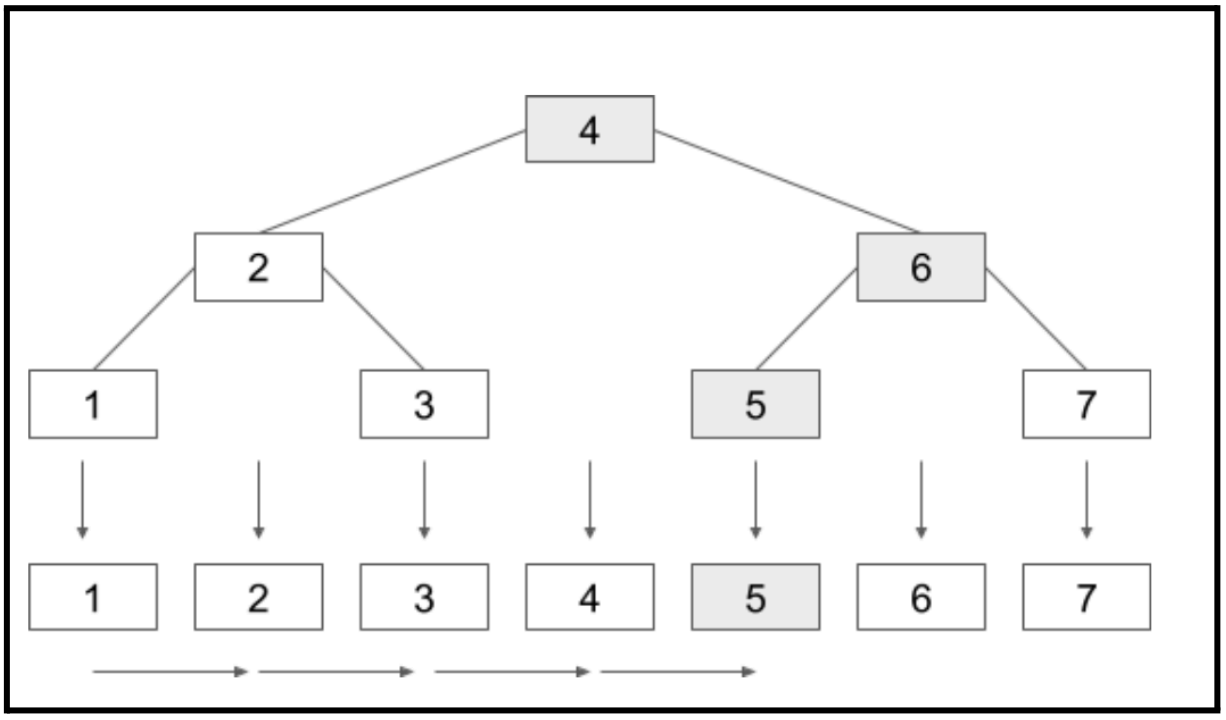
```
DROP INDEX index_of_catalog_id ON products;
```

MySQL підтримує два типи індексу:

1. BTREE-бінарне дерево;

2. HASH-хеш-таблиця.

Загальна ідея бінарного дерева полягає в тому, що значення зберігаються по порядку, і всі листові сторінки знаходяться на однаковій відстані від кореня. BTREE-індекс прискорює доступ до даних, оскільки підсистема зберігання не потрібно сканувати всю таблицю для пошуку потрібної інформації. У BTREE-індекси індексовані стовпці зберігаються в упорядкованому вигляді, і вони корисні для пошуку за діапазоном даних.



Оскільки вузли дерева відсортовані, їх можна використовувати як для пошуку значень, так і запити з сортуванням за допомогою **ORDER BY**.

```
CREATE INDEX index_of_catalog_id USING BTREE ON products (catalog_id);
CREATE INDEX index_of_catalog_id USING HASH ON products (catalog_id);
DESCRIBE products;
```

Хеш-індекс будується на основі хеш-таблиці та корисний тільки для точного пошуку із зазначенням усіх стовпців індексу. Для кожного рядка підсистема зберігання обчислює хеш-код індексованих стовпців. В індексі зберігаються хеш-коди та покажчики на відповідні рядки.

Селективність індексу – це відношення кількості проіндексованих значень до загального кількості рядків у таблиці. Індекс з високою селективністю хороший тим, що дозволяє MySQL при пошуку відповідностей фільтрувати більше рядків. Унікальний індекс має селективність, рівну одиниці.

Якщо селективність індексу низька, після локалізації ділянки або набору елементів, де знаходиться потрібне значення, його потрібно додатково просканувати для точного пошуку значення. У разі унікального індексу ми можемо одразу отримати шуканий результат, без сканування навіть невелика ділянка таблиці.

Завдання для самоконтролю

1. Які типи даних ви знаєте? Опишіть їх.
2. Чи можна перетворити дробове число в ціле? Ціле в дробове? У яких випадках можна рядок перетворити в число?
3. Наведіть приклади операцій. Для чого призначена операція присвоєння?
4. Які існують правила і рекомендації для іменування змінних?

5. Наведіть приклади використання функцій оброблення символів.

6. Які функції існують для введення і виведення символів?

7. Як отримати довжину рядка?

8. Що таке кортеж?

9. Що таке словник?

10. Які методи списків ви знаєте?

5. CRUD операції

5.1. Загальне поняття

Чотири базові операції: створення (create), читання (read), оновлення (update) та видалення (delete) записів називаються за першими буквами англійських слів CRUD-операціями. У мові запитів SQL їм відповідає чотири оператори: **INSERT**, **SELECT**, **UPDATE** та **DELETE**.

5.2. INSERT

За операцію вставки даних відповідає оператор **INSERT**. Він підтримує кілька типів вставки, перший з яких - однорядкова вставка:

```
INSERT INTO catalogs VALUES (NULL, 'Процесори');
INSERT INTO catalogs VALUES (NULL, 'Мат.плати');
INSERT INTO catalogs VALUES (NULL, 'Відеокарти');
```

Ключове слово **NULL** можна замінити **DEFAULT**:

```
INSERT INTO catalogs VALUES (DEFAULT, 'Процесори');
INSERT INTO catalogs VALUES (DEFAULT, 'Мат.плати');
INSERT INTO catalogs VALUES (DEFAULT, 'Відеокарти');
```

Безліч INSERT-запитів можна замінити на багаторядковий оператор **INSERT**:

```
INSERT INTO catalogs VALUES
  (DEFAULT, 'Процесори');
```

```
(DEFAULT, 'Мат.плати');
(DEFAULT, 'Відеокарти');
```

Такий варіант вставки працює набагато швидше.

5.3. SELECT

Для отримання даних використовується оператор **SELECT**:

```
SELECT id, name FROM catalogs;
```

Після ключового слова **SELECT** ми вказуємо список вилучених стовпців. У нас їх всього два: **id** та **name**. Після ключового слова **FROM** ми вказуємо ім'я бази даних, у разі – **catalogs**. Порядок шпальт після ключового слова **SELECT** можна змінювати.

```
SELECT name, id FROM catalogs;
```

Можемо виводити лише частину стовпців, наприклад лише **name**:

```
SELECT name FROM catalogs;
```

За допомогою зірочки ми можемо запросити всі стовпці бази даних у порядку, в якому вони визначено у таблиці:

```
SELECT * FROM catalogs;
```

Завдання для самоконтролю

1. Які типи даних ви знаєте? Опишіть їх.

2. Чи можна перетворити дробове число в ціле? Ціле в дробове? У яких випадках можна рядок перетворити в число?

3. Наведіть приклади операцій. Для чого призначена операція присвоєння?

4. Які існують правила і рекомендації для іменування змінних?

5. Наведіть приклади використання функцій оброблення символів.

6. Які функції існують для введення і виведення символів?

7. Як отримати довжину рядка?

8. Що таке кортеж?

9. Що таке словник?

10. Які методи списків ви знаєте?

6. АГРЕГАЦІЯ ДАНИХ

6.1. Групування даних

У мові SQL для таких груп призначене спеціальне ключове слово GROUP BY.

Наприклад, завдання отримання унікальних значень можна вирішити так:

```
SELECT catalog_id FROM products GROUP BY catalog_id;
```

Як значення для створення груп можуть виступати не тільки стовпці таблиці, а й значення, що обчислюються. Наприклад, давайте розділимо користувачів у таблиці на три групи: народилися у 80-х, 90-х та 2000-х роках.

```
SELECT id, name, SUBSTRING(birthday_at, 1, 3) FROM users;
```

Тут ми перетворимо календарний тип **DATETIME** поля birthday_at до строкового значення та за допомогою функції **SUBSTRING** отримуємо перші три цифри року народження значенням псевдонім, що обчислюється, за допомогою ключового слова **AS** і відсортуємо значення при допомоги **ORDER BY**.

```
SELECT id, name, SUBSTRING(birthday_at, 1, 3) AS decade FROM users
ORDER BY decade;
```

Зверніть увагу, що ми можемо використати псевдонім **decade** у конструкції **ORDER BY**. За допомогою конструкції **GROUP BY** ми можемо згрупувати поля за декадами:

```
SELECT SUBSTRING(birthday_at, 1, 3) AS decade FROM users GROUP BY
decade;
```

Якщо ми спробуємо вивести імена користувачів, ми зазнаємо невдачі:

```
SELECT id, name, SUBSTRING(birthday_at, 1, 3) AS decade FROM users
GROUP BY decade;
```

Кожна з груп містить кілька користувачів і незрозуміло, якого з них слідує виводити. Раніше MySQL виводила випадкового користувача, проте зараз така поведінка скасовано. Такий режим, як і раніше, можна включити, СУБД навіть підказує в повідомленні про помилки, як це можна зробити. Однак краще цього не робити, щоб ваш SQL-код залишався сумісним з іншими СУБД.

Яку користь можна отримати із згрупованих значень? MySQL надає кілька функцій, що називаються агрегатними. Вони дозволяють працювати з вмістом груп, отриманих GROUP BY. Наприклад, ми можемо підрахувати кількість записів усередині кожного з груп:

```
SELECT COUNT(*), SUBSTRING(birthday_at, 1, 3) AS decade FROM users
GROUP BY decade;
```

Таким чином, у нас 3 користувача народилися у 80-х, два у 90-х та один у 2000-х. Отримані значення ми, як і раніше, можемо сортувати за допомогою конструкції **ORDER BY**:

```
SELECT
  COUNT(*),
  SUBSTRING(birthday_at, 1, 3) AS decade
FROM
  users
```

```
GROUP BY
  decade
ORDER BY
  decade DESC;
```

Причому сортувати можна не групованому значенню, а й за будь-яким іншим полем. Наприклад, давайте призначимо функції **COUNT()** псевдонім **total** та відсортуємо результати за цим значенням:

```
SELECT
  COUNT(*) AS total,
  SUBSTRING(birthday_at, 1, 3) AS decade
FROM
  users
GROUP BY
  decade
ORDER BY
  total DESC;
```

Порядок проходження ключових слів важливий: ми не можемо розміщувати ключове слово **ORDER BY** раніше **GROUP BY**, інакше ми отримуємо повідомлення про помилку. Це саме стосується ключового слова **LIMIT**, яке завжди має розташовуватися після всіх інших ключових слів.

Коли у запиті ми не використовуємо конструкцію **GROUP BY**, вся таблиця сприймається як одна велика група. Тому якщо ми застосуємо функцію **COUNT(*)** до таблиці users, ми можемо отримати кількість всіх користувачів, без урахування року їх народження:

```
SELECT COUNT(*) FROM users;
```

Завдання для самоконтролю

1. Які типи даних ви знаєте? Опишіть їх.

2. Чи можна перетворити дробове число в ціле? Ціле в дробове? У яких випадках можна рядок перетворити в число?

3. Наведіть приклади операцій. Для чого призначена операція присвоєння?

4. Які існують правила і рекомендації для іменування змінних?

5. Наведіть приклади використання функцій оброблення символів.

6. Які функції існують для введення і виведення символів?

7. Як отримати довжину рядка?

8. Що таке кортеж?

9. Що таке словник?

10. Які методи списків ви знаєте?

КВЕСТОВІ ЗАВДАННЯ

1. У районі Севастопольської площі м. Києва близько 15:30 25.05.2021 року був здійснений розбійний напад на відділення «Швидко Гроші». Під час досудового розслідування було встановлено двох осіб (контактний номер першої особи – 050 *** 79 03, другої особи – 063 *** 00 02), ймовірно причетних до вчинення даного злочину. Але, встановити факт наявності зв'язку між двома даними особами слідчому не вдалося.

Розкажіть про аналітичні програмні продукти, які використовуються під час аналізу телефонного трафіку? Яким способом Ви можете встановити факт з'єднання двох осіб (за допомогою SQL та NoSQL баз даних)?

Отримавши дозвіл судді на тимчасовий доступ до інформації, що знаходиться в операторів, провайдерів телекомунікацій необхідно здійснити аналіз отриманої інформації за допомогою SQL та NoSQL баз даних *(файли для обробки та аналізу надаються викладачем).*

2. До Солом'янського управління поліції Головного управління Національної поліції в місті Києві близько 07:00 ранку 18.05.2021 року надійшло повідомлення про зникнення гр. Х.

Батьки зниклої пояснили, що гр. Х. проживає самостійно за адресою: м. Київ, вул. Солом'янська, 41, кв. 33. Опитані сусіди повідомили поліції, що за дві неділі до зникнення за гр. Х. постійно у період з 21:00 до 22:00 приїжджав автомобіль. Останній раз сусіди бачили це 17.05.2021 року. Будь-які ідентифікаційні ознаки водія та автомобіля, який приїжджав за гр. Х., сусіді не можуть повідомити із-за темної пори доби. Контактний номер мобільного телефону гр. Х. +380635**2331.

Розкажіть про аналітичні програмні продукти, які використовуються під час аналізу телефонного трафіку? Яким способом Ви зможете встановити абонента мобільного телефону, який ймовірно причетний до зникнення гр. Х.?

Визначте абонента мобільного телефону, який ймовірно причетний до зникнення гр. Х, використовуючи SQL та NoSQL бази даних *(файли для обробки та аналізу надаються викладачем).*

3. 12.05.2021 року по вул. Мілютенко, 21 в м. Києві у період з 04:50 по 05:00 незаконно заволоділи транспортним засобом BMW X5 (2016 року випуску), державний номерний знак – АА 0001 ІО.

Розкажіть про аналітичні програмні продукти, які використовуються під час аналізу телефонного трафіку? Яким способом Ви можете встановити конкретних абонентів мобільного зв'язку, які могли бути свідками або особами, що вчинили злочин, за допомогою SQL та NoSQL баз даних?

Отримавши дозвіл судді на тимчасовий доступ до інформації, що знаходиться в операторів, провайдерів телекомунікацій необхідно здійснити аналіз отриманої інформації за допомогою SQL та NoSQL баз даних *(файли для обробки та аналізу надаються викладачем).*

4. У ході досудового розслідування у слідчого виникла необхідність встановити місцезнаходження гр. А, стосовно якого будуть проводитися негласні слідчі (розшукові) дії. За інформацією сім'я гр. А володіє трьома транспортними засобами (Mercedes Benz G500 AA 0001 JK, BMW 750i AA 0002 JK, Jaguar S-Type AA 0003 JK) та чотирма квартирами в місті Києві за адресами:

– м. Київ, був. Кольцова, 143;

– м. Київ, вул. Верхній Вал, 4А;

– м. Київ, вул. Мілютенка, 38;

– м. Київ, вул. Михайла Драгоманова, 20А.

Розкажіть про можливості інформаційно-аналітичного комплексу «Безпечне місто», а також картографічних сервісів в мережі Інтернет? Яким способом Ви можете встановити ймовірне місцепроживання гр. А на основі інформації, отриманої з інформаційно-аналітичного комплексу «Безпечне місто»?

Отримавши інформацію з інформаційно-аналітичного комплексу «Безпечне місто» здійсніть її аналіз за допомогою SQL та NoSQL баз даних.

Проаналізувавши інформацію надайте зображення місцевості, де проживає гр. А *(файли для обробки та аналізу надаються викладачем).*

5. Під час досудового розслідування злочину, передбаченого ч. 5 ст. 191 Кримінального кодексу України «Привласнення, розтрата майна або заволодіння ним шляхом зловживання службовим становищем в особливо великих розмірах», слідчим була отримання інформація із Реєстру речових прав на нерухоме майно Міністерства юстиції України стосовно володільців квартир у будинку, що розташований за адресою: м. Київ, вул. Солом'янська, 98.

Розкажіть про Реєстр речових прав на нерухоме майно Міністерства юстиції України. Яким способом Ви можете встановити злочинну схему розтрати бюджетних коштів на основі інформації, отриманої із Реєстру речових прав на нерухоме майно Міністерства юстиції України?

Отримавши оброблену інформацію з Реєстру речових прав на нерухоме майно Міністерства юстиції України здійсніть її аналіз за допомогою SQL та NoSQL баз даних.

6. Під час досудового розслідування злочину, передбаченого ч. 5 ст. 191 Кримінального кодексу України «Привласнення, розтрата майна або заволодіння ним шляхом зловживання службовим становищем в особливо великих розмірах», слідчим була отримана інформація з інформаційно-телекомунікаційної системи «АРКАН» Державної прикордонної служби України про історію перетинів кордону підозрюваним.

Розкажіть про інформаційно-телекомунікаційну систему «АРКАН» Державної прикордонної служби України. Яким способом Ви зможете встановити зв'язки підозрюваного з іншими особами на основі інформації, отриманої із інформаційно-телекомунікаційної системи «АРКАН» Державної прикордонної служби України?

Отримавши інформацію із інформаційно-телекомунікаційної системи «АРКАН» Державної прикордонної служби України здійсніть її обробку та аналіз за допомогою SQL та NoSQL баз даних.

Визначивши осіб, які спільно перетинали кордон із підозрюваним, встановіть персональні дані цих осіб за допомогою технік OSINT *(файли для обробки та аналізу надаються викладачем).*

7. У рамках кримінального провадження за фактом вчинення злочину, передбаченого ч. 2 ст. 364 Кримінального кодексу України «Зловживання владою або службовим становищем», слідчим була отримана інформація з інформаційно-телекомунікаційної системи «АРКАН» Державної прикордонної служби України про історію перетинів кордону особи, що переховується від органів досудового розслідування.

Розкажіть про інформаційно-телекомунікаційну систему «АРКАН» Державної прикордонної служби України. Яким способом Ви можете встановити місцеперебування та зв’язки особи, що переховується від органів досудового розслідування, на основі інформації, отриманої із інформаційно-телекомунікаційної системи «АРКАН» Державної прикордонної служби України?

Отримавши інформацію із інформаційно-телекомунікаційної системи «АРКАН» Державної прикордонної служби України здійсніть її обробку та аналіз за допомогою SQL та NoSQL баз даних *(файли для обробки та аналізу надаються викладачем).*

8. Під час досудового розслідування злочину, передбаченого ч. 3 ст. 209 Кримінального кодексу України «Легалізація (відмивання) доходів, одержаних злочинним шляхом в особливо великому розмірі», слідчим була отримана інформація про рух коштів на банківському рахунку (банківські транзакції).

Розкажіть про аналітичні програмні продукти, які використовуються під час аналізу банківських транзакцій? Яким способом Ви можете встановити

конкретних фізичних (юридичних) осіб, які могли бути причетні до легалізації (відмиванні) доходів, одержаних злочинним шляхом, за допомогою SQL та NoSQL баз даних?

Отримавши дозвіл судді на тимчасовий доступ до банківських транзакцій необхідно здійснити аналіз отриманої інформації за допомогою SQL та NoSQL баз даних *(файли для обробки та аналізу надаються викладачем).*

9. Під час досудового розслідування злочину, передбаченого ч. 5 ст. 191 Кримінального кодексу України «Привласнення, розтрата майна або заволодіння ним шляхом зловживання службовим становищем в особливо великих розмірах», слідчим була отримана інформація про рух коштів на банківському рахунку (банківські транзакції).

Розкажіть про аналітичні програмні продукти, які використовуються під час аналізу банківських транзакцій? Яким способом Ви можете встановити конкретних фізичних (юридичних) осіб, які могли бути причетні до розкрадання бюджетних коштів за допомогою SQL та NoSQL баз даних?

Отримавши дозвіл судді на тимчасовий доступ до банківських транзакцій необхідно здійснити аналіз отриманої інформації за допомогою SQL та NoSQL баз даних *(файли для обробки та аналізу надаються викладачем).*

ВИСНОВКИ

Технології бази даних SQL та NoSQL є універсальним, достатньо доступним та функціональним рішенням, що застосовується для обробки, аналізу та візуалізації оперативної інформації, яка застосовується оперативними та аналітичними підрозділами правоохоронних органів для розкриття та прогнозування кримінальних правопорушень під час здійснення кримінального аналізу (КА).

Як було вже вказано авторами раніше, під час виконання завдань КА за допомогою можливостей технологій баз даних можливо виконувати різноманітні задачі, в тому числі, наприклад, обробляти та аналізувати:

1) інформацію з електронних комунікаційних мереж:

– з'єднання абонентів мобільного зв'язку;

– історії використання мобільних терміналів та телефонних сім-карток;

2) рух товарів (надання послуг), грошових коштів та віртуальних активів:

– рух товарів (надання послуг) з Єдиного реєстру податкових накладних (ЄРПН) та автоматизованої системи митного оформлення (АСМО) «Інспектор»;

– рух грошових коштів за банківськими рахунками;

– рух грошових коштів за допомогою електронних платіжних систем;

– рух віртуальних активів;

3) інформацію з державних інформаційних ресурсів:

– з системи «Аркан»;

– з Єдиного реєстру довіреностей;

– з Державного реєстру речових прав на нерухоме майно;

– з інформаційно-аналітичного комплексу «Безпечне місто» та інформаційної підсистеми «Гарпун» «Інформаційного порталу Національної поліції України» про маршрути пересування транспортних засобів;

4) інформацію з комп'ютерів та мобільних терміналів зв'язку;

5) відкриті дані:

– з Єдиного державного веб-порталу відкритих даних;

– з Єдиного веб-порталу використання публічних коштів;

– з електронної системи публічних закупівель Prozorro;

– з системи онлайн аукціонів з продажу та здачі в оренду майна ProZorro.Продажі тощо.

У цьому посібнику авторами докладно висвітлено загальний порядок використання можливостей технологій баз даних при роботі з інформацією в електронній (цифровій) формі. Наведені у посібнику методи застосування інструментів обробки та аналізу даних технологій баз даних доповнені прикладами їх використання в практичних завданнях КА.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Amsterdam, Jonathan. Atomic File Transactions, Part 1. URL: http://archive.oreilly.com/pub/a/onjava/2001/11/07/atomic.html

2. Обробка та аналіз за допомогою MS Excel та IBM i2 Analyst's Notebook інформації щодо одночасного перетину кордону декількома особами: практ. посібн. / [В. Школьніков, О. Корнейко, С. Тіхонов та ін.; за заг. ред. В. Школьнікова та О. Корнейка]. К. : Вид-во Нац. акад. внутр. справ, 2020. 144 с.

3. Встановлення місцеперебування осіб з використанням інформаційно-аналітичного комплексу «Безпечне місто»: практ. посібн. / [В. Школьніков, О. Корнейко, С. Тіхонов та ін.]. К. : Вид-во Нац. акад. внутр. справ, 2020. 68 с.

4. Кластерний аналіз телефонного трафіку: практ. посібн. / [В. Школьніков, О. Корнейко, С. Тіхонов та ін.; за заг. ред. В. Школьнікова та О. Корнейка]. К. : Вид-во Нац. акад. внутр. справ, 2020. 40 с.

5. Виявлення за допомогою технологій кримінального аналізу фактів розтрати, привласнення чи заволодіння коштами державного бюджету на основі даних з Державного реєстру речових прав на нерухоме майно : практ. посібн. / [В. Школьніков, О. Корнейко, С. Тіхонов та ін.; за заг. ред. В. Школьнікова та О. Корнейка]. К. : Вид-во Нац. акад. внутр. справ, 2020. 180 с.

6. Обробка та аналіз інформації про протиправні транзакції криптоактивів : практ. посібн. / В. Школьніков, О. Корнейко, Ю. Орлов; за заг. ред. О. Корнейка. К. : Вид-во Нац. акад. внутр. справ, 2021. 134 с.

7. Комп'ютерна програма "Встановлення зв'язків особи, яка перетинала державний кордон". Свідоцтво про реєстрацію авторського права на твір № 121550. URL: https://iprop-ua.com/cr/gmk8ccu6

8. Комп'ютерна програма "Встановлення фактів кримінальних правопорушень, вчинених шляхом кредитно-фінансових операцій". Свідоцтво

про реєстрацію авторського права на твір № 121557. URL: https://iprop-ua.com/cr/ztu7vls7

9. Комп'ютерна програма "Встановлення фактів розтрати, привласнення чи заволодіння коштами державного бюджету". Свідоцтво про реєстрацію авторського права на твір № 121554. URL: https://iprop-ua.com/cr/q24hghbe

10. Здобуття доказової інформації в мережі Інтернет із використанням можливостей мови програмування Python : метод. реком. / [В.І. Школьніков, Ю.Ю. Орлов, О.В. Корнейко, А.А. Вознюк]. К. : Вид-во Нац. акад. внутр. справ, 2019. 58 с.

Навчальне видання

Школьніков Владислав Ігорович
Корнейко Олександр Васильович

ВИКОРИСТАННЯ ТЕХНОЛОГІЙ SQL ТА NoSQL БАЗ ДАНИХ ДЛЯ ВИРІШЕННЯ ЗАВДАНЬ КРИМІНАЛЬНОГО АНАЛІЗУ

Навчальний посібник

Відповідальний за випуск та комп’ютерна верстка: В. Школьніков

Відділ підготовки навчально-наукових видань
Національної академії внутрішніх справ.
03035, м. Київ, пл. Солом’янська, 1.
Свідоцтво про внесення суб’єкта видавничої справи до державного реєстру видавців, виготовників і розповсюджувачів видавничої продукції ДК № 4155 від 13.09.2011.
Підписано до друку 30.12.2024. Формат 60x84/16. Папір офсетний.
Ум.-друк. арк. 6,02. Тираж 50 прим.

Друк: ФОП Поліщук О. В.
Свідоцтво суб'єкта видавничої справи ДК № 2142 від 31.03.2015.
07400, м. Бровари, вул. Незалежності, 2, кв. 148.
тел. (044) 592-13-49