Лабораторна робота №5

Дослідження вимірювальної інформації у базі SQLite

Мета: ознайомитись з принципами зберігання вимірювальних даних у базах SQLite, освоїти аграгацію, фільтрацію індексацію та часові запити

5.1 Теоретичні відомості

База даних — це структурований набір даних, що зберігаються на комп'ютері. Це потужний інструмент для ефективного зберігання та впорядкування даних. Сьогодні ви можете знайти бази даних скрізь, від вашого смартфона до комп'ютера та онлайн-сервісів.

Ось деякі основні характеристики баз даних:

- Ефективне зберігання даних : Бази даних пропонують централізоване місце для зберігання структурованих даних, таких як продажі, запаси та фінанси.
- Швидкий пошук даних : Бази даних дозволяють швидко отримувати доступ до інформації, набагато швидше, ніж читання даних, що зберігаються в електронних таблицях та інших плоских файлах.
- Ефективне керування даними : Бази даних дозволяють швидко вставляти, оновлювати та видаляти дані.
- Спільний доступ до даних : Бази даних дозволяють кільком користувачам і програмам одночасно отримувати доступ до центральної інформації.
- Аналіз даних : Бази даних забезпечують структуру для аналізу даних та отримання цінної інформації.

Ось два поширені типи баз даних:

- Реляційні системи керування базами даних (РСБД): це найпоширеніший тип баз даних, що організовує дані в таблиці з рядками та стовпцями. Популярними РСБД є PostgreSQL, MySQL, MariaDB, Oracle Database, SQL Server та IBM Db2.
- Бази даних документів (або бази даних NoSQL): ці типи баз даних зберігають дані у вигляді документів. Популярними базами даних документів є MongoDB, Databricks та Amazon DynamoDB.

SQL — це стандартна мова для взаємодії з реляційними СУБД. Вона дозволяє:

- Створювати та підтримувати структури бази даних, такі як таблиці.
- Вставляти, оновлювати та видаляти дані.
- Отримати дані з таблиць.

SQL складається з трьох основних частин:

- Мова визначення даних (DDL) займається створенням та модифікацією структури бази даних. Наприклад, оператори CREATE TABLE, ALTER TABLE та DROP TABLE.
- Мова маніпулювання даними (DML) надає оператори для запитів даних, такі як оператор SELECT , та модифікації даних, як-от оператори INSERT , UPDATE та DELETE .
- Мова керування даними (DCL) включає оператори, що працюють з авторизацією та безпекою користувачів, такі як оператори GRANT та REVOKE.

Основою реляційної СУБД є таблиці. Таблиці дозволяють упорядковувати дані в рядках і стовпцях:

- Стовпці: представляють певні поля
- Рядки: зберігає окремі записи.

SQL був однією з перших комерційних мов баз даних з 1970 року. Відтоді різні постачальники баз даних впроваджували SQL у свої продукти з деякими варіаціями. Американський інститут стандартів (ANSI) опублікував перші стандарти SQL у 1986 році, щоб забезпечити більшу відповідність постачальників. ANSI оновив стандарт SQL у 1992 році, SQL92 та SQL2, а також знову у 1999 році як SQL99 та SQL3. Щоразу ANSI додавав нові функції та команди до мови SQL. Сьогодні ANSI та Міжнародна організація зі стандартизації підтримують стандарти SQL як стандарт **ISO/IEC 9075.** Найновішим випуском стандарту є **SQL:2023**. Стандарти SQL формалізують структури синтаксису та поведінку SQL у різних продуктах баз даних. Це стає ще більш важливим для баз даних з відкритим кодом, таких як MySQL та PostgreSQL , де СУБД розробляються переважно спільнотами, а не великими корпораціями.

SQL — це *декларативна* мова. Це означає, що ви повідомляєте базі даних *, чого* хочете, а не *як* це отримати. Ця особливість робить SQL дуже легкою для вивчення порівняно з іншими мовами програмування. Базові структури операторів SQL - це дієслова, підмети та умови. Більшість SQL-інструкцій дотримуються певного шаблону:

- **Дієслово** (дія): це дія, яку має виконати база даних, наприклад SELECT, INSERT, UPDATE, та DELETE.
- **Тема** (ціль): це об'єкт бази даних, з яким ви працюєте, наприклад, таблиця.
- **Умова** (фільтр): виберіть, які дані вас цікавлять.

```
SELECT first_name
FROM employees
WHERE YEAR (hire_date) = 1999;
```

SELECT first_name: отримує значення у first_name стовпці, FROM employees: отримує дані з employees таблиці, WHERE YEAR(hire_date) = 1999: враховувати лише тих співробітників, які приєдналися до 1999.

Літерали — це константи або фактичні значення, які ви розміщуєте в SQL-інструкції. **Рядки** беруться в одинарні лапки, наприклад, 'Anthony' та 'Blue'. Зверніть увагу, що рядковий літерал чутливий до регістру. Наприклад, 'Anthony' відрізняється від 'anthony'.

Ключові слова – це зарезервовані слова в SQL. Вони мають спеціальні значення. Поширені ключові слова: SELECT, FROM, , WHERE, CREATE, INSERT, тощо. Ключові слова не враховують регістр. За домовленістю ми використовуватимемо великі літери для ключових слів, щоб виділити їх. Ідентифікатори посилаються на об'єкти в базі даних, такі як таблиці, стовпці, індекси тощо.

Цей SELECT оператор дозволяє отримувати дані з однієї або кількох таблиць. Ось основний синтаксис оператора SELECT, який отримує дані з однієї таблиці:

```
SELECT
  select_list
FROM
```

```
  table_name;
```

Під час оцінки SELECT оператора система бази даних FROM спочатку оцінює речення, а потім SELECT речення. Крапка з комою (;) не є частиною запиту. Сервер бази даних використовує крапку з комою для розділення двох SQL-інструкцій.

Якщо потрібно отримати дані з усіх стовпців таблиці, потрібно перерахувати всі стовпці в реченні SELECT таким чином:

```
SELECT
  column1,
  column2,
  column3
FROM
  table_name;
```

або

```
select * from table_name;
```

ORDER BY є необов'язковим пунктом оператора SELECT. Цей ORDER BY пункт дозволяє сортувати результуючий набір за одним або кількома виразами сортування у порядку зростання та/або спадання.

Ось синтаксис цього ORDER BY речення:

```
SELECT
  select_list
FROM
  table_name
ORDER BY
  sort_expression [ASC | DESC];
```

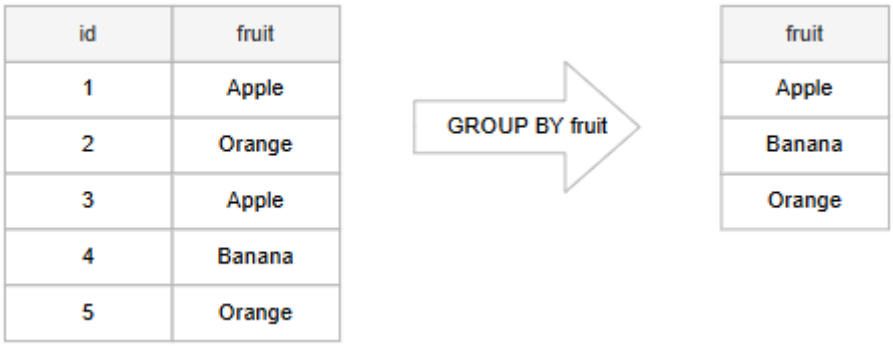
sort_expression Спочатку вкажіть у реченні вираз сортування ( ), ORDER BY на основі якого потрібно відсортувати результуючий набір. Це sort_expression може бути стовпець таблиці або вираз, що містить стовпець таблиці. По-друге, використовуйте ASC для сортування набору результатів у порядку зростання та DESC для сортування набору результатів у порядку спадання. sort_expression Спочатку вкажіть у реченні вираз сортування ( ), ORDER BY на основі якого потрібно відсортувати результуючий набір. Це sort_expression може бути стовпець таблиці або вираз, що містить стовпець таблиці.

По-друге, використовуйте ASC для сортування набору результатів у порядку зростання та DESC для сортування набору результатів у порядку спадання.

GROUP BY необов'язковим пунктом оператора SELECT. Цей GROUP BY пункт дозволяє групувати рядки на основі значень одного або кількох стовпців. Він повертає один рядок для кожної групи.

```
SELECT
  column1,
  column2,
  aggregate_function (column3)
FROM
  table_name
GROUP BY
  column1,
```

```
  column2;
```



Речення GROUP BY групує рядки результуючого набору в групи. Щоб указати умову для фільтрації груп, використовується речення HAVING. Якщо ви використовуєте HAVING речення без GROUP BY речення, HAVING речення поводиться як речення WHERE .

```
SELECT
  column1,
  column2,
  aggregate_function (column3)
FROM
  table1
GROUP BY
  column1,
  column2
HAVING
  group_condition;
```

Щоб вибрати різні значення зі стовпця таблиці, використовується DISTINCT оператор у SELECT реченні наступним чином:

```
SELECT DISTINCT
  column1
FROM
  table_name;
```

У цьому синтаксисі SELECT оператор повертає результуючий набір, який містить унікальні значення з column1 таблиці.

Щоб обмежити кількість рядків, що повертаються оператором SELECT , використовуються речення LIMIT and OFFSET .

```
SELECT
  column_list
FROM
  table1
ORDER BY
  column_list
LIMIT
  row_count
OFFSET
  row_to_skip;
```

Визначає LIMIT row_count кількість рядків (row_count), що повертаються запитом. Речення OFFSET row_to_skip пропускає row_to_skip рядки, перш ніж почати їх повертати.

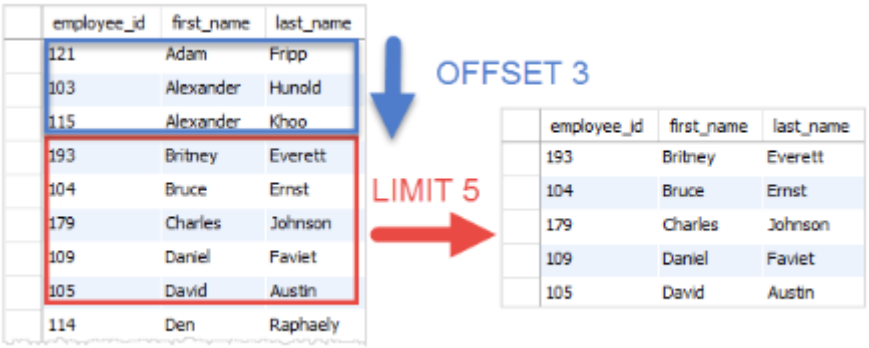


Рис.5.1 – Приклад роботи OFFSET та LIMIT

Щоб вибрати певні рядки з таблиці на основі однієї або кількох умов, икористовується WHERE речення в SELECT операторі. Речення WHERE з'являється одразу після FROM речення. Воно містить один або декілька логічних виразів, які обчислюють кожен рядок у таблиці. Зверніть увагу, що SQL має тризначну логіку, а саме true, fals ета NULL. Це означає, що якщо рядок призводить до того, що значення condition обчислюється як false або null, запит не включатиме цей рядок до результуючого набору.

Оператор	Значення
=	Дорівнює
<> (!=)	Не дорівнює
<	Менше ніж
>	Більше ніж
<=	Менше або дорівнює
>=	Більше або дорівнює

Рис. 5.2 – Оператори порівняння

```
SELECT
  employee_id,
  first_name,
  last_name,
  salary
FROM
  employees
WHERE
  salary > 14000
ORDER BY
  salary DESC;
```

Цей запит виведе співробітників, чиї зарплати перевищують 14,000 за допомогою WHERE речення

Оператор AND (I) об’єднує два або більше логічних виразів — обидва мають бути істинними (TRUE), щоб умова виконалась.

```
SELECT * FROM readings
WHERE temperature_c > 20 AND humidity_pct < 50;
```

Повертає рядки, де температура більше 20°C **і** вологість менше 50%.

Оператор OR (АБО) виконує логічне “або”: умова істинна, якщо хоча б один вираз є істинним.

```
SELECT * FROM readings
WHERE temperature_c > 25 OR light_lux > 500;
```

Оператор BETWEEN перевіряє, чи належить значення до заданого діапазону (включно з межами).

```
SELECT * FROM readings
WHERE temperature_c BETWEEN 18 AND 25;
```

Повертає записи, де температура у межах 18–25°C.

Оператор LIKE використовується для пошуку за шаблоном у текстових полях. Символи підстановки:

% — будь-яка кількість символів;

_ — один символ.

```
SELECT * FROM locations
WHERE name LIKE '%Lab%';    -- містить слово 'Lab'
```

Оператор `IS NULL` перевіряє, чи поле має значення **NULL** (невизначене).

```
SELECT * FROM readings
WHERE co_ppm IS NULL;
```

Повертає записи, де значення `co_ppm` відсутнє.

INNER JOIN необов'язковим пунктом оператора SELECT. Цей INNER JOIN пункт дозволяє об'єднувати рядки з двох пов'язаних таблиць.

```
SELECT
  column1,
  column2
FROM
  table1
  INNER JOIN table2 ON condition;
```

Спочатку вкажіть першу таблицю в FROM реченні (table1), вкажіть другу таблицю, рядки якої потрібно об'єднати з першою таблицею в INNER JOIN реченні (table2), визначте умову condition для зіставлення рядків між двома таблицями після ON ключового слова. Ця умова називається умовою об'єднання.

Для кожного рядка в table1, INNER JOINречення перевіряє кожен рядок в table2та перевіряє умову. Якщо condition дорівнює true, INNER JOIN об'єднує рядки з обох таблиць в один рядок і включає його до кінцевого результуючого набору. Зазвичай умова порівнює значення між двома стовпцями двох таблиць на предмет рівності:

```
SELECT
  column1,
  column2
FROM
  table1
  INNER JOIN table2 ON column1 = column2;
```

Припустимо, є дві таблиці:

Таблиця Хмає два стовпці id (ключ) та x.

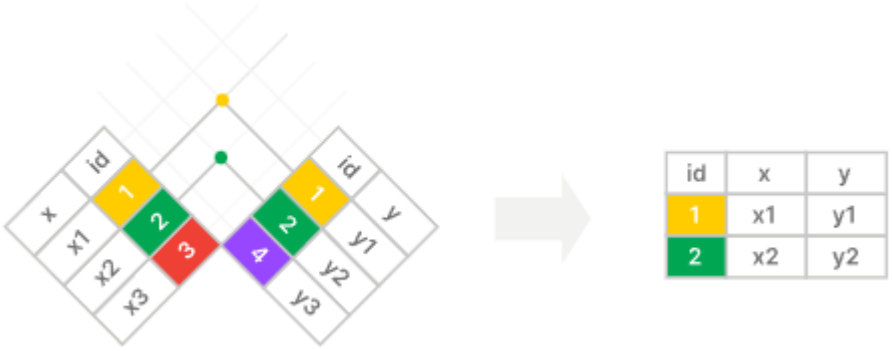
Таблиця Yтакож має два стовпці id(ключ) та y.

id	x
1	x1
2	x2
3	x3

id	y
1	y1
2	y2
4	y3

Внутрішнє об'єднання зіставляє рядки між таблицями X та Y, використовуючи значення у id стовпцях.

Внутрішнє об'єднання включає лише рядки зі збігаючимися значеннями у id стовпцях і не включає незбігаючі рядки в результуючий набір:



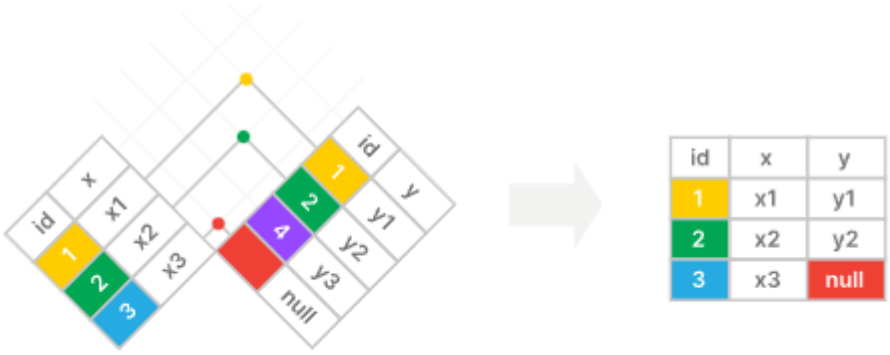
Речення LEFT JOIN є необов'язковим реченням оператора SELECT. LEFT JOIN речення дозволяє об'єднувати рядки з двох таблиць.

```
SELECT
  column1,
  column2
FROM
  left_table
LEFT JOIN right_table ON condition;
```

id	x
1	x1
2	x2
3	x3

id	y
1	y1
2	y2
4	y3

Ліве об'єднання зіставляє рядки між таблицями X, Y використовуючи значення у id стовпцях обох таблиць. Ліве об'єднання включає всі рядки з лівої таблиці (X) та відповідні рядки з правої таблиці (Y); якщо відповідних рядків немає, використовується null для стовпців правої таблиці (Y):



FULL OUTER JOIN є необов'язковим пунктом оператора SELECT. Цей FULL OUTER JOIN пункт дозволяє об'єднувати рядки між двома таблицями.

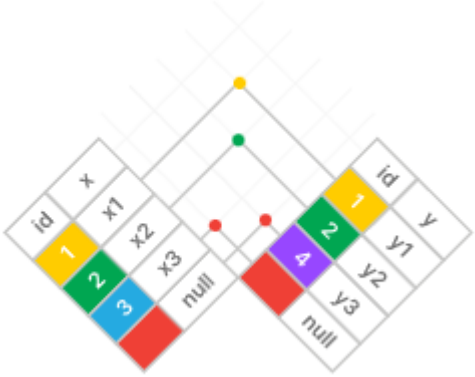
```
SELECT
  column1,
  column2
FROM
  table1
FULL JOIN table2 ON table2.column2 = table1.column1;
```

Спочатку надайте першу таблицю (table1) у FROM реченні, після ключових слів вкажіть другу таблицю (table2), яку потрібно об'єднати з першою таблицею (table1) FULL OUTER JOIN, визначте a condition для зіставлення рядків між двома таблицями. Ви можете поєднувати кілька умов за допомогою AND операторів OR.

id	x
1	x1
2	x2
3	x3

id	y
1	y1
2	y2
4	y3

Повне зовнішнє об'єднання зіставляє рядки між таблицями X, Y використовуючи значення у id стовпцях обох таблиць. Повне зовнішнє об'єднання включає рядки з обох таблиць, незалежно від того, чи мають ці рядки відповідні рядки з іншої таблиці. Воно використовується null для стовпців рядків у таблиці, які не мають відповідних рядків в іншій таблиці:



id	x	y
1	x1	y1
2	x2	y2
3	x3	null
4	null	y3

Речення CROSS JOIN є необов'язковим реченням оператора SELECT. CROSS JOIN речення дозволяє створювати комбінації всіх рядків з двох таблиць.

```
SELECT
  select_list
FROM
  table1
CROSS JOIN table2;
```

id	x
1	x1
2	x2
3	x3

id	x
1	x1
2	x2
3	x3

Перехресне з'єднання об'єднує кожен рядок з лівої таблиці (X) та правої таблиці (Y), щоб створити кінцевий набір результатів:

id	x	y
1	x1	y1
2	x1	y2
3	x1	y3
1	x2	y1
2	x2	y2
3	x2	y3
1	x3	y1
2	x3	y2
3	x3	y3

У реляційних базах даних таблиця — це структурований набір даних, організований у рядки та стовпці:

- Рядки представляють записи.
- Стовпці представляють атрибути даних.

Щоб створити нову таблицю, використовується CREATE TABLE оператор. Ось основний синтаксис CREATE TABLE оператора:

```
CREATE TABLE courses (
```

```
  name VARCHAR(255) NOT NULL,
  description TEXT,
  duration DEC(4, 2) NOT NULL
   );
```

name у стовпці зберігається назва курсу. Тип даних стовпця name — VARCHAR максимальна кількість 255 символів. Якщо зберегти рядок, більший за це значення, система бази даних видасть помилку. — це NOT NULL обмеження, яке гарантує, що name стовпець завжди міститиме дані. Іншими словами, він не повинен містити NULL. descriptiony стовпці зберігається опис курсу. Тип даних стовпця description — TEXT який може зберігати дуже великий текст. descriptionне має обмежень, таких як NOT NULL. Це означає, що ви можете зберігати NULL у стовпці. duration стовпець зберігає тривалість курсу в годинах, наприклад 4.5. Тип даних стовпця duration— це DEC(4,2)той, що зберігає точні десяткові числа

Первинний ключ — це стовпець або набір стовпців, які однозначно ідентифікують кожен рядок у таблиці. Первинний ключ гарантує, що кожен рядок унікальний. Таблиця містить один і лише один первинний ключ. Якщо первинний ключ містить один стовпець, цей стовпець називається стовпцем первинного ключа. Якщо стовпець первинного ключа містить два або більше стовпців, це стовпці первинного ключа. Первинний ключ також відомий як складений ключ .

Ось основні характеристики первинного ключа:

- Унікальність : кожне значення у стовпці(ях) первинного ключа має бути унікальним.
- Не NULL : Стовпець первинного ключа не може містити NULL.
- Незмінний : значення у стовпці первинного ключа не повинні змінюватися.
- Щоб створити первинний ключ, використовується ключове слово PRIMARY KEY

```
CREATE TABLE projects (
  project_id INT PRIMARY KEY,
  project_name VARCHAR(255) NOT NULL,
  start_date DATE NOT NULL,
  end_date DATE NOT NULL
);
```

У цьому прикладі ми використовуємо PRIMARY KEY ключові слова у project_id визначенні стовпця. Зазвичай PRIMARY KEY обмеження також NOT NULL неявно застосовує обмеження, за винятком SQLite .У цьому прикладі ми визначаємо PRIMARY KEY як обмеження стовпця, оскільки оголошуємо його у визначенні стовпця первинного ключа.

Функція SQL COUNT— це агрегатна функція , яка повертає кількість рядків, повернутих запитом. Наприклад, ви можете використовувати COUNT функцію в операторі SELECT , щоб отримати кількість співробітників, кількість співробітників у кожному відділі, кількість співробітників, які обіймають певну посаду тощо.

Нижче показано синтаксис функції SQL COUNT:

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

5.2 Порядок виконання лабораторної роботи

1 Завантажити файл measurement.sqlite;

2 Відкрити онлайн SQLite Viewer за адресою: https://inloop.github.io/sqlite-viewer/ та завантажити файл measurement.sqlite;

3 Ознайомитись з базою та на сайті: https://dbdiagram.io/d побудувати діаграму бази даних за допомогою файлу dbdiagram.txt.

4 Вивести скільки всього було виміряно значень з усіх датчиків

5 Вивести список сенсорів та їх місце положення

6 Вивести останні 10 вимірювань певного сенсора;

7 Вивести кількість вимірювань на день;

8 Обчислити середню температуру та вологість за день для кожного сенсора;

9 Визначити мінімальну напругу живлення, вивести список батарей в яких наруга менша за 3.22V

5.3 Зміст звіту

1. Найменування і мета роботи;
2. Код по кожному пункту порядку виконання роботи;
3. Результати роботи по кожному пункту виконання роботи;
4. Висновки.

5.4 Контрольні запитання

1. Що таке SQL і для чого він використовується?

2. Як вибрати лише унікальні значення з поля?

3. Як працює BETWEEN і чи включає він межі діапазону?

4. Як знайти максимальне/мінімальне значення?

5. Що таке JOIN і для чого він використовується?

Додаток 1

```
Table locations {
  id integer [pk]
  name text
  description text
}

Table sensors {
  id integer [pk]
  code text [unique]
  type text [note: 'env_multi або iaq_mox']
  location_id integer
  fw_version text
  serial text [unique]
}

Table readings {
  ts text [note: 'ISO timestamp']
  sensor_id integer
  temperature_c real
  humidity_pct real
  light_lux integer
  voltage_v real
  co_ppm real
  no2_ppb real
  indexes {
    (sensor_id, ts) [name: 'ix_readings_sensor_ts']
    (ts) [name: 'ix_readings_ts']
  }
  primary key (ts, sensor_id)
}

Ref: sensors.location_id > locations.id
Ref: readings.sensor_id > sensors.id

Table v_hourly_avg {
  sensor_id integer [ref: > sensors.id]
  hour_ts text
  avg_temp_c real
  avg_humidity_pct real
  avg_light_lux real
  avg_voltage_v real
  avg_co_ppm real
  avg_no2_ppb real

}
```