# Основні команди SQL-запитів

# (на прикладі БД Postgres)

Олександр Лавріненко
грудень 2024

# SELECT - вибір даних

## Основна команда

SELECT використовується для вибору даних з таблиць.

Команда SELECT може включати різні умови, обмеження та функції для вибору потрібних даних.

FROM - звідки, з якої таблиці будуть вибиратись дані

LIMIT - обмеження за кількістю рядків у result set.

## Приклади

Вибрати всі дані з таблиці "**users**":

```
SELECT *
FROM users;
```

Вибрати лише ім'я та прізвище з таблиці "**users**" у перших ста записів:

```
SELECT first_name, last_name
FROM users
LIMIT 100;
```

# `WHERE` - фільтрування даних

### Фільтрування

`WHERE` - це команда, яка дозволяє фільтрувати дані за певними критеріями.

Використовується для обмеження результатів `SELECT` до тих рядків, які відповідають умові.

### Умови

Умови можуть включати оператори порівняння (`<, >, =, !=`), логічні оператори (`AND, OR, NOT`) та інше (`BETWEEN, IN`).

### Приклад

Вибрати всі дані з таблиці "**users**":

```
SELECT *
FROM users
WHERE age BETWEEN 34 AND 40;
```

# `ORDER BY` - сортування даних

### Сортування за стовпцем

`ORDER BY` дозволяє впорядкувати результати запиту за одним або кількома стовпцями. Наприклад, щоб відсортувати таблицю "users" за прізвищем у порядку зростання (від А до Я):

```
SELECT *
FROM users
ORDER BY last_name ASC;
```

### Сортування у спадаючому порядку

Щоб відсортувати дані у спадаючому порядку (від Я до А), використовуйте DESC:

```
SELECT * FROM users
ORDER BY last_name DESC;
```

Для сортування за кількома стовпцями, вкажіть їх через кому (`ASC` можна не вказувати):

```
SELECT *
FROM users
ORDER BY last_name DESC, first_name;
```

(спочатку за прізвищем у спадаючому порядку, потім за ім'ям у зростаючому)

# `GROUP BY` - групування даних

### Групування

Групує рядки за значенням одного або декількох стовпців.

### Агрегація

Часто використовується разом з функціями агрегації (`COUNT, SUM, AVG` тощо) для обчислення підсумків.

### Зведення

Дозволяє отримати зведену інформацію про дані.

### Приклад групування

```
SELECT a.user_id
FROM accounts AS a
WHERE a.currency != 'UAH'
GROUP BY a.user_id
ORDER BY a.user_id;
```

### Приклад агрегації та зведення

```
SELECT a.user_id,
COUNT(a.id) AS "cnt_acc",
SUM(a.amount/100.0) AS "sum_bigamount"
FROM accounts AS a
GROUP BY a.user_id;
```

# JOIN - об’єднання таблиць

### З'єднання

JOIN використовується для об'єднання даних з різних таблиць.

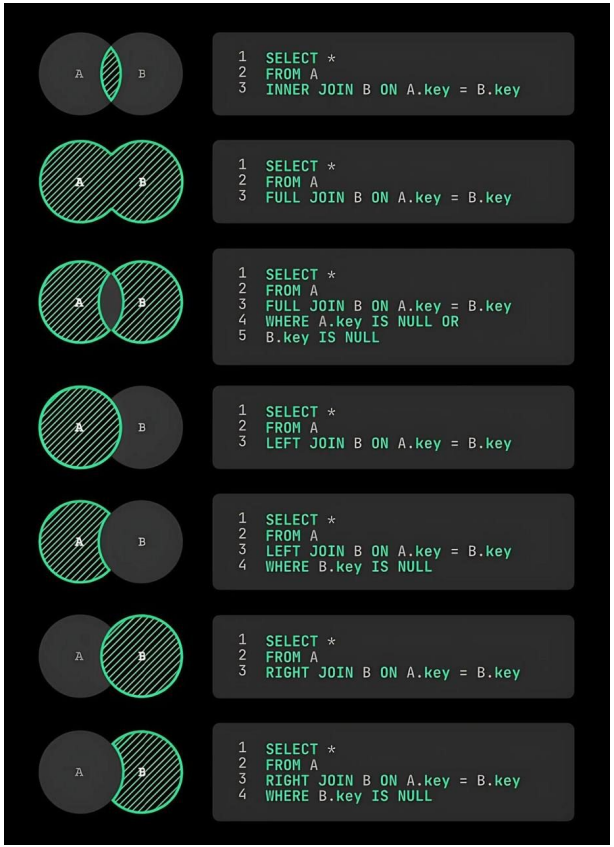
### Типи з'єднань

Існують різні типи з'єднань, такі як:

INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL JOIN, CROSS JOIN.

### Вибір

Ви можете вибрати дані з обох таблиць, які відповідають умові з'єднання.




CODEUS

# INSERT - додавання даних

### Додавання

INSERT використовується для додавання нових рядків до таблиці.

### Значення

Ви повинні вказати значення для кожного стовпця в таблиці.

### Унікальність

Якщо є обмеження унікальності (***constraints***), вставлені значення повинні бути унікальними.

### Приклад додавання

Додаємо кілька полів (за назвою):

```
INSERT INTO students (name, age)
VALUES ('Potuzhniy', 46);
```

Додаємо усі поля, які є в таблиці:

```
INSERT INTO students
VALUES (1,'Potuzhniy', 46, 'Kyiv');
```

# UPDATE - оновлення даних

### Оновлення

UPDATE використовується для оновлення наявних даних у таблиці.

### Умова

Ви повинні вказати умову, щоб визначити, які рядки слід оновити.

### Нові значення

Ви також повинні вказати нові значення для стовпців, які потрібно оновити.

### Приклади використання

**UPDATE без умов - не робіть так!**

```
UPDATE persons
SET age = 35, name = 'John Doe';
```

Ми затремо УСІ наявні дані у стовпчиках `age` та `name` таблиці даними `35` та `John Doe`.

**UPDATE з умовами:**

```
UPDATE persons
SET age = 35, name = 'John Doe'
WHERE person_id = 18;
```

Тут оновиться тільки рядок, який відповідає умові.

# DELETE - видалення даних

**Видалення**

DELETE використовується для видалення рядків з таблиці.

**Умова**

Ви можете вказати умову, щоб визначити, які рядки слід видалити.

**Обережність**

Видалення даних є **незворотнім**, тому будьте обережні при використанні цієї команди.

**Приклад видалення УСІХ даних з таблиці:**

```
DELETE
FROM persons;
```

**Приклад видалення тільки тих даних з таблиці, які відповідають зазначеним умовам:**

```
DELETE
FROM persons
WHERE age < 18;
```

CODEUS

# CREATE TABLE - створення таблиці

**Constraints:**

PRIMARY KEY

NOT NULL

FOREIGN KEY (REFERENCES)

UNIQUE

CHECK

**Приклад створення таблиці**

```
CREATE TABLE
IF NOT EXISTS accounts (
    id          INT PRIMARY KEY,
    user_id     INT REFERENCES users(id),
    currency    CHAR(3) REFERENCES nbu_rates(rate),
    amount      numeric(6,2) CHECK (amount > 0),
    start_date  DATE NOT NULL,
    end_date    DATE NOT NULL
);
```

# Корисні команди

`CURRENT_DATE` - отримуємо поточну дату (без годин)

`ROUND(duration, 1)` - округлення до десятих (1 знак після роздільника)

**Приклади застосування**

```
SELECT CURRENT_DATE as date;

SELECT
ROUND(duration, 1) AS rounded_duration
FROM some_table;

SELECT
EXTRACT(EPOCH FROM (end_time - start_time)) /
60 AS "minutes"
FROM some_table;
```