3. БАЗИ ТА СХОВИЩА ДАНИХ

**Посилання на плейлист де є вся інформація**:

<https://youtube.com/playlist?list=PLeFClW9cFhQSPCZORfkuFbG7pT8yg7BfJ&si=qYx7HBg1ugScQGGY>

База: <https://www.youtube.com/watch?v=QbrbSmy69dg&list=PLeFClW9cFhQSPCZORfkuFbG7pT8yg7BfJ&index=15>

3.1. Ключі та нормалізація даних: Основні нормальні форми (1NF, 2NF, 3NF, BCNF).

Основні нормальні форми - це набір правил, які дозволяють структурувати дані таким чином, щоб уникнути аномалій при зберіганні та оновленні даних.

<https://www.youtube.com/watch?v=zqQxWdTpSIA>

Нульова форма (UNF):

* Перед приведенням БД до нормальної форми, потрібно привести її до табличного виду так, щоб він відповідав базовим принципам реляційної теорії.
* Рядки в таблицях не повинні бути пронумеровані (порядок рядків та стовпців не має значення).

Перша нормальна форма (1NF):

Призначення:

* Рядок - зберігати дані;
* Стовпців – зберігати структурну інформацію;
* Осередок - зберігати атомарне значення.

Принципи:

* У таблиці не повинно бути дублюючих рядків;
* У кожному осередку таблиці зберігається атомарне значення (одне складове значення);• У стовпці зберігаються дані одного типу;
* Немає масивів та списків у будь-якому вигляді.

Якщо таблиця створена з дотриманням всіх реляційних принципів, отже, вона вже знаходиться в першій нормальній формі, таким чином, по суті, абсолютно всі реляційні БД знаходяться в першій нормальній формі.

Друга нормальна форма (2NF):

* Таблиця повинна бути в 1NF;
* Таблиця повинна мати ключ;
* Усі неключові стовпці таблиці повинні залежати від повного ключа (якщо ключ складової).

Ключ - це стовпець чи набір стовпців, якими гарантовано можна відрізнити рядки друг від друга, т. е. ключ ідентифікує кожен рядок таблиці. За ключом ми можемо звернутися до конкретного рядка даних у таблиці.

* Якщо ключ складається з кількох стовпців, то всі інші неключові стовпці повинні залежати від усього ключа, тобто від усіх стовпців у цьому ключі.
* У таблиці повинно бути даних, які можна отримати, знаючи лише половину ключа, т. е. лише один стовпець зі складеного ключа.
* Таблиця повинна мати правильний ключ, яким можна ідентифікувати кожен рядок.

Третя нормальна форма (3NF):

* Таблиця повинна бути в 2NF;
* У таблицях має бути відсутня транзитивна залежність.

Транзитивна залежність - Це коли неключові стовпці залежить від значень інших неключових стовпців.

Неключові стовпці не повинні намагатися грати роль ключа в таблиці.

Нормальна форма Бойса-Кодда (BCNF):

* Таблиця повинна бути в 3NF;
* Ключові атрибути складеного ключа не повинні залежати від неключових атрибутів.

Вимоги BCNF є актуальними лише для таблиць зі складовими ключами.

Частина складеного первинного ключа повинна залежати від неключевого стовпця.

3.2. Основні концепції систем баз даних:

Основні концепції систем баз даних охоплюють різні аспекти проектування, управління та захисту даних. Давайте розглянемо кожен з них:

* Модель даних - Це абстрактна структура, яка визначає, як дані будуть організовані, зберігатися та використовуватися в системі баз даних. Приклади моделей даних: реляційна, ієрархічна, мережева, об'єктно-орієнтована тощо.
* Мова запитів - Це набір синтаксичних правил для формулювання запитів до бази даних з метою вибірки, модифікації, видалення або вставки даних. Приклади: SQL (Structured Query Language), LINQ (Language Integrated Query), GraphQL тощо.
* Транзакція - Це логічна одиниця обробки, що виконується в базі даних, яка забезпечує атомарність, консистентність, ізоляцію та довіру (ACID). Транзакція виконується або повністю, або не виконується взагалі.
* ACID - властивості транзакції:
* Атомарність: Транзакція виконується як атомарна одиниця, що означає, що вона або виконується повністю, або не виконується зовсім.
* Консистентність: Транзакція переводить базу даних з одного стану в інший стан, що також є допустимим для системи.
* Ізоляція: Рівень ізоляції визначає, які зміни, внесені однією транзакцією, будуть видимі іншим транзакціям.
* Довіра: Гарантується, що якщо транзакція успішно завершилася, то її зміни будуть незмінними.
* Резервне копіювання та відновлення – Процес створення резервних копій бази даних для запобігання втраті даних в разі непередбачуваних подій, таких як віруси, помилки або катастрофи. Відновлення – процес відновлення даних з резервної копії.
* Розподіленість і реплікація даних:
* Розподіленість даних – це процес розміщення даних на різних серверах для полегшення доступу та зменшення навантаження.
* Реплікація даних – це процес створення та підтримки кількох копій даних для забезпечення доступності та витривалості.
* Безпека даних - Забезпечення конфіденційності, цілісності та доступності даних. Включає в себе аутентифікацію, авторизацію, шифрування, контроль доступу тощо.

3.3. Моделювання даних:

1. Створення моделі даних для інформаційної системи:

* Цей процес полягає в аналізі вимог до системи та розробці моделі, яка відображає структуру та взаємозв'язки даних.
* Модель даних визначає сутності, їх атрибути та взаємозв'язки між ними.

1. Концептуальна, логічна, фізична моделі даних:

* Концептуальна модель описує загальну структуру даних без урахування деталей реалізації. Це високорівневе представлення даних та їх відносин.
* Логічна модель визначає, як дані будуть організовані та взаємозв'язані в рамках конкретної СУБД, без залежності від конкретної фізичної реалізації.
* Фізична модель визначає, як дані будуть зберігатися та організовані на рівні файлової системи чи бази даних.

1. ER-модель:

* ER (Entity-Relationship)-модель - це спосіб візуалізації та опису структури даних, що базується на сутностях, атрибутах та взаємозв'язках між ними.
* У цій моделі сутності представлені у вигляді прямокутників, а взаємозв'язки - у вигляді стрілок або ліній, що з'єднують сутності.

1. Нотації ER-моделей:

* Нотації ER-моделей визначають стандарти для позначення сутностей, атрибутів та взаємозв'язків.
* Найпоширеніші нотації включають в себе нотацію Чена (символи прямокутників, овалів, стрілок) та нотацію Кроу (застосовання стрілок та ліній для з'єднання сутностей).

3.4. Реляційні бази даних.

Реляційні бази даних (РБД) - це тип баз даних, де дані представлені у вигляді таблиць з рядками та стовпцями.

1. Особливості організації та зберігання даних у реляційних базах даних:

* Таблиці: Дані організовані у вигляді таблиць, де кожен рядок представляє запис, а кожен стовпець - атрибут цього запису.
* Ключі: В реляційних базах даних ключі використовуються для унікальної ідентифікації записів (первинний ключ) та для встановлення взаємозв'язків між таблицями (зовнішні ключі).
* Нормалізація: Реляційні бази даних часто використовують нормалізацію для уникнення дублювання даних та забезпечення цілісності даних.
* Мова запитів SQL: Для доступу до даних в реляційних базах даних використовується мова запитів SQL (Structured Query Language).

1. Основні характеристики реляційних баз даних:

* Цілісність даних: Реляційні бази даних дотримуються цілісності даних, що означає, що вони гарантують коректність та консистентність даних.
* Нормалізація: Реляційні бази даних сприяють нормалізації даних для забезпечення оптимальної структури даних.
* Множинність: Реляційні бази даних дозволяють створювати складні взаємозв'язки між таблицями за допомогою зовнішніх ключів.

1. DBMS (Database Management System):

* DBMS - це програмне забезпечення, що дозволяє створювати, управляти та використовувати бази даних.
* DBMS надає інтерфейс для виконання операцій з даними, таких як додавання, видалення, зміна та пошук, а також забезпечує безпеку, цілісність та ефективність роботи з даними.
* Приклади популярних реляційних DBMS включають MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server тощо.

3.5. Побудова запиту:

1. Мова SQL (Structured Query language).

* SQL - це стандартна мова запитів для взаємодії з реляційними базами даних.
* Вона включає в себе різноманітні команди для створення, модифікації та видалення даних, а також для виконання операцій вибірки та сортування.Приклади команд
* SQL: SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER тощо.

1. Мова DDL (Data Definition Language).

* DDL - це підмова SQL, яка використовується для визначення та зміни структури бази даних.
* Вона включає команди для створення, зміни та видалення об'єктів бази даних, таких як таблиці, індекси, представлення тощо.
* Приклади команд DDL: CREATE TABLE, ALTER TABLE, DROP TABLE, CREATE INDEX тощо.

1. Мова DML (Data Manipulation Language).

* DML - це підмова SQL, яка використовується для маніпулювання даними в межах таблиць бази даних.
* Вона включає команди для вибірки, вставки, оновлення та видалення даних.
* Приклади команд DML: SELECT, INSERT, UPDATE, DELETE тощо.

1. Мова DCL (Data Control Language).

* DCL - це підмова SQL, яка використовується для керування правами доступу до даних.
* Вона включає команди для надання або відмови від доступу до бази даних та її об'єктів.
* Приклади команд DCL: GRANT, REVOKE тощо.

1. Мова TCL (Transaction Control Language).

* TCL - це підмова SQL, яка використовується для керування транзакціями.
* Вона включає команди для управління транзакціями, такими як COMMIT для збереження змін та ROLLBACK для скасування їх.
* Приклади команд TCL: COMMIT, ROLLBACK тощо.

3.6. Обробка запитів: Основні операції реляційної алгебри:

1. Відбір (selection)

* Вибирає рядки, які задовольняють певну умову.
* Синтаксис: σ <умова> (R), де R - вихідна таблиця, а умова - логічний вираз, який повинен бути задоволений.
* Синтаксис мовою SQL:

SELECT \* FROM table\_name WHERE condition;

1. Проєкція (projection)

* Вибирає стовпці з вихідної таблиці.
* Синтаксис: π <список\_атрибутів> (R), де R - вихідна таблиця, а список\_атрибутів - перелік атрибутів, які потрібно вибрати.
* Синтаксис мовою SQL:

SELECT column1, column2, ... FROM table\_name;

1. Об'єднання (union)

* Об'єднує рядки з двох таблиць без дублювання.
* Синтаксис: R ∪ S, де R і S - вихідні таблиці.
* Синтаксис мовою SQL:

SELECT \* FROM table1

UNION

SELECT \* FROM table2;

1. Перетин (intersection)

* Вибирає рядки, які зустрічаються у двох таблицях.
* Синтаксис: R ∩ S, де R і S - вихідні таблиці.
* Синтаксис мовою SQL:

SELECT \* FROM table1

INTERSECT

SELECT \* FROM table2;

1. Різниця (difference)

* Вибирає рядки з першої таблиці, які не зустрічаються у другій таблиці.
* Синтаксис: R - S, де R і S - вихідні таблиці.
* Синтаксис мовою SQL:

SELECT \* FROM table1

EXCEPT

SELECT \* FROM table2;

1. Декартовий добуток (cartesian product)

* Створює комбінації всіх можливих пар рядків з двох таблиць.
* Синтаксис: R × S, де R і S - вихідні таблиці.
* Синтаксис мовою SQL:

SELECT \* FROM table1, table2;

1. Об'єднання за атрибутом (Join)

* Об'єднує рядки з двох таблиць на основі спільного атрибуту.
* Синтаксис: R ⨝<умова\_об'єднання> S, де R і S - вихідні таблиці, а умова\_об'єднання - умова, за якою об'єднуються рядки.
* Синтаксис мовою SQL:

SELECT \* FROM table1 INNER JOIN table2 ON

table1.column = table2.column;

1. Ділення (Division)

* Вибирає всі унікальні значення з однієї таблиці, які мають відповідність до всіх значень з іншої таблиці.
* Синтаксис: R ÷ S, де R і S - вихідні таблиці.