Лекція 1

Вступ до баз даних та інформаційних систем

1. Мета та завдання курсу

Метою курсу є формування у студентів базових теоретичних знань та практичних навичок у сфері проєктування, використання та супроводу баз даних як основи сучасних інформаційних систем.

У контексті академічного курсу **«Бази даних та інформаційні системи»**, роль даних розглядається не просто як «сховище цифр», а як динамічний ресурс, навколо якого будується архітектура всієї системи.

Ось ключові аспекти ролі даних, які є фундаментом цього курсу:

1. Перетворення: Дані → Інформація → Знання

В теорії ІС фундаментальним є розуміння ієрархії **DIKW** (Data, Information, Knowledge, Wisdom).

- **Дані (Data):** Сирі факти, сигнали, які самі по собі не мають контексту (наприклад, число «40»).
- **Інформація (Information):** Дані, що пройшли обробку та отримали контекст («Температура пацієнта — 40°C»).
- **Знання (Knowledge):** Розуміння того, як використати інформацію для прийняття рішень («Це висока температура, потрібне лікування»).

2. Моделювання реального світу

Роль даних у базах даних (БД) полягає у створенні **інформаційної моделі** предметної області.

- **Абстракція:** Ми не зберігаємо весь об'єкт, а лише ті атрибути, що важливі для системи (наприклад, для студента — це ПІБ та номер заліковки, а не колір очей).
- **Структурування:** Використання ER-моделей (Entity-Relationship) дозволяє відобразити зв'язки між об'єктами реального світу в цифрову форму.

3. Забезпечення цілісності та несуперечності

В рамках курсу вивчається роль СКБД (систем керування базами даних) як «охоронця» даних. Дані в сучасних ІС мають відповідати принципам **ACID**:

- **Atomicity (Атомарність):** Транзакція виконується повністю або не виконується зовсім.
- **Consistency (Узгодженість):** Дані завжди відповідають встановленим правилам (наприклад, вік не може бути від'ємним).
- **Isolation (Ізольованість):** Користувачі не заважають один одному при одночасній роботі.
- **Durability (Довговічність):** Якщо дані записані, вони не зникнуть при збої.

4. Роль даних у багаторівневій архітектурі

Сучасні ІС зазвичай будуються за принципом розділення обов'язків. Дані займають нижній, але найважливіший рівень:

1. **Рівень представлення (UI):** Те, що бачить користувач.
2. **Рівень бізнес-логіки:** Алгоритми обробки.
3. **Рівень даних:** Безпосередньо БД та засоби доступу до них. *Така структура дозволяє змінювати інтерфейс програми, не зачіпаючи самі дані.*

5. Спільне використання та безпека

Дані в ІС — це спільний ресурс. Їхня роль полягає в тому, щоб бути доступними для багатьох користувачів одночасно, забезпечуючи при цьому:

- **Розмежування прав доступу:** Кожен бачить лише те, що йому дозволено.
- **Централізацію:** Усунення дублювання інформації (надлишковості), що критично для великих підприємств.

Порівняння підходів до роботи з даними

Характеристика	Файлові системи (старий підхід)	Бази даних (сучасний підхід)
Залежність	Програма жорстко прив'язана до формату файлу	Дані незалежні від програм (через СКБД)

Характеристика	Файлові системи (старий підхід)	Бази даних (сучасний підхід)
Надлишковість	Висока (дані часто дублюються)	Мінімальна (завдяки нормалізації)
Пошук	Повільний (послідовний перебір)	Швидкий (завдяки індексам)
Безпека	Базова (на рівні файлів ОС)	Гнучка (на рівні таблиць, рядків, стовпців)

1. Основні визначення

- **База даних (БД)** — це впорядкована сукупність взаємопов’язаних даних, що зберігаються в пам'яті комп’ютера та відображають стан об’єктів певної предметної області (наприклад: банк, бібліотека, інтернет-магазин).
- **СУБД (Система управління базами даних)** — це комплекс програмних засобів, за допомогою яких користувачі можуть створювати структуру БД, наповнювати її, редагувати, здійснювати пошук та захищати дані.

  Проста аналогія: База даних — це «книга» (інформація), а СУБД — це «читач з ручкою та індексами» (інструмент), який знає, як швидко знайти потрібну сторінку або додати новий розділ.

Ключові компоненти реляційної моделі

Більшість сучасних систем є **реляційними** (від англ. *relation* — відношення/таблиця). Вся інформація в них представлена у вигляді двовимірних таблиць.

- **Таблиця (Відношення):** Основний об'єкт для зберігання даних.
- **Поле (Атрибут / Стовпець):** Визначає властивість об'єкта (наприклад, «Прізвище» або «Ціна»). Кожне поле має свій **тип даних** (текст, число, дата).
- **Запис (Кортеж / Рядок):** Повний набір даних про один конкретний примірник об’єкта (наприклад, дані про одного конкретного студента).
- **Первинний ключ (Primary Key):** Унікальний ідентифікатор запису (наприклад, номер паспорта або ID), який не дозволяє рядкам повторюватися.
- **Зовнішній ключ (Foreign Key):** Поле, яке посилається на первинний ключ іншої таблиці, створюючи зв'язок між ними.

Функції СУБД

Сучасна СУБД (наприклад, MySQL, PostgreSQL, Oracle) бере на себе складну технічну роботу:

1. **Управління пам'яттю:** Визначає, як дані фізично лежать на диску.
2. **Забезпечення цілісності:** Не дозволяє видалити клієнта, якщо у нього є активні замовлення.
3. **Управління доступом:** Хто може тільки читати, а хто — видаляти дані.
4. **Резервне копіювання:** Відновлення системи після збоїв.
5. **Підтримка мови SQL:** Обробка запитів на вибірку та зміну даних.

Рівні архітектури (Модель ANSI/SPARC)

Для забезпечення незалежності даних від програм, архітектуру БД поділяють на три рівні:

- **Зовнішній рівень (View):** Те, що бачить кінцевий користувач (інтерфейс програми).
- **Концептуальний рівень (Logical):** Логічна структура всієї БД (таблиці та зв’язки), над якою працює адміністратор.
- **Внутрішній рівень (Physical):** Опис того, як дані записані у файлах на жорсткому диску.

Поняття транзакції та ACID

В рамках курсу ви обов'язково зустрінете термін **Транзакція** — це логічна одиниця роботи (наприклад, переказ грошей з картки на картку). Вона повинна відповідати принципам **ACID**:

- **A (Atomicity) — Атомарність:** Виконується або все, або нічого.
- **C (Consistency) — Узгодженість:** Після завершення БД залишається в коректному стані.
- **I (Isolation) — Ізольованість:** Паралельні транзакції не заважають одна одній.
- **D (Durability) — Довговічність:** Якщо транзакція підтверджена, дані не зникнуть навіть при вимкненні світла.

в’язки між таблицями — це основа реляційних баз даних. Саме вони дозволяють уникати дублювання даних і забезпечують цілісність інформації.

У реляційній моделі зв'язки реалізуються за допомогою **зовнішніх ключів**

Зв'язок «один-до-багатьох» (1:M або 1:N)

Це найпоширеніший тип зв'язку. Він означає, що один запис у таблиці А може відповідати багатьом записам у таблиці Б, але запис у таблиці Б має лише одного «власника» в таблиці А.

Як будується: Зовнішній ключ (FK) завжди додається до таблиці на стороні **«багато»**.

- **Приклад:** Один **Автор** може написати багато **Книг**, але одна конкретна книга (зазвичай) має одного основного автора.
- **Реалізація:** У таблицю Books ми додаємо стовпець author_id, який посилається на id у таблиці Authors.

Зв'язок «багато-до-багатьох» (M:M або M:N)

Цей зв'язок виникає, коли один запис у таблиці А може відповідати багатьом записам у таблиці Б, і навпаки.

Як будується: Реляційні бази даних **не підтримують прямий зв'язок** між двома таблицями типу «багато-до-багатьох». Щоб його реалізувати, створюється третя, проміжна таблиця (її називають **таблицею зв’язку** або **junction table**).

- **Приклад:** Один **Студент** відвідує багато **Курсів**, а на один **Курс** записано багато **Студентів**.
- **Реалізація:**
    1. Маємо таблицю Students (id, name).
    2. Маємо таблицю Courses (id, title).
    3. Створюємо таблицю Enrollments (student_id, course_id). Вона містить два зовнішні ключі, які разом можуть утворювати складений первинний ключ.

Зв'язок «один-до-одного» (1:1)

Зустрічається рідше. Один запис у таблиці А відповідає лише одному запису в таблиці Б.

Як будується: Зовнішній ключ можна додати в будь-яку з таблиць (зазвичай у ту, що є «дочірньою»), або зробити первинний ключ однієї таблиці одночасно і зовнішнім ключем до іншої.

- **Приклад: Користувач** та його **Профіль** (з додатковими налаштуваннями, які не хочеться тримати в головній таблиці для швидкодії).
- **Реалізація:** В таблиці Profiles поле user_id є унікальним (UNIQUE) зовнішнім ключем до Users.

Порівняльна таблиця

Тип зв'язку	Де розміщується зовнішній ключ (FK)?	Приклад
1:1	У будь-якій з двох таблиць (зазвичай у допоміжній)	Людина — Паспорт
1:M	У таблиці на стороні **«багато»**	Відділ — Співробітники
M:M	У спеціальній **проміжній таблиці**	Актори — Фільми

Нормалізація — це покроковий процес проектування структури бази даних, який дозволяє уникнути надлишковості (дублювання) даних та забезпечити їхню цілісність.

Простими словами: ми беремо одну велику, «брудну» таблицю і розбиваємо її на кілька менших, логічно пов'язаних таблиць.

Навіщо потрібна нормалізація?

Без неї в системі виникають **аномалії**:

- **Аномалія вставки:** ви не можете додати дані про курс, поки на нього не записався хоча б один студент.
- **Аномалія видалення:** якщо ви видалите єдиного студента з курсу, ви можете випадково видалити й опис самого курсу.
- **Аномалія оновлення:** якщо назва курсу змінилася, її доведеться виправляти у сотнях рядків, де згадується кожен студент.

Три основні етапи (Нормальні форми)

1. Перша нормальна форма (1NF): Атомарність

Правило: Кожна клітинка таблиці повинна містити лише одне неподільне (атомарне) значення. Не повинно бути списків або повторюваних груп.

- *До:* В одному полі «Телефони» записано три номери через кому.
- *Після:* Кожен номер — в окремому рядку (або краще — в окремій таблиці).

Друга нормальна форма (2NF): Повна функціональна залежність

Правило: Таблиця вже в 1NF, і всі неключові поля залежать від **всього** первинного ключа (якщо він складений), а не від його частини.

- *Приклад:* Таблиця «Замовлення товарів». Ключ складається з ID_Замовлення + ID_Товару. Поле Дата_Замовлення залежить лише від ID_Замовлення, а не від товару.
- *Дія:* Виносимо дані про саме замовлення (дата, клієнт) в одну таблицю, а перелік товарів у цьому замовленні — в іншу.

Третя нормальна форма (3NF): Відсутність транзитивних залежностей

Правило: Таблиця в 2NF, і жодне неключове поле не залежить від іншого неключового поля. Всі мають залежати тільки від «ключа, всього ключа і нічого, крім ключа».

- *Приклад:* Таблиця «Студенти» зі стовпцями: ID, Прізвище, ID_Групи, Назва_Кафедри.
- *Проблема:* Назва_Кафедри залежить від ID_Групи, а не безпосередньо від студента.
- *Дія:* Створюємо окрему таблицю «Групи», де вказуємо назву кафедри, а в таблиці «Студенти» залишаємо лише ID_Групи.

Як нормалізація визначає зв'язки?

Процес нормалізації автоматично «підказує» нам архітектуру:

1. Коли ми виносимо повторювані дані в окрему таблицю (наприклад, Дані про кафедру з таблиці Студентів) — утворюється зв'язок **один-до-багатьох (1:M)**.
2. Якщо після нормалізації ми бачимо, що об'єкти існують незалежно, але взаємодіють (Студенти та Курси) — це сигнал до створення проміжної таблиці та зв'язку **багато-до-багатьох (M:M)**.

Золоте правило: У навчальних курсах та більшості бізнес-систем вважається достатнім доведення бази даних до **Третьої нормальної форми (3NF)**.