

# БД

## 1. Означення бази даних, відмінності БД від файлових систем.

База даних (БД) — це структурована сукупність даних, які відображують стан об'єктів певної предметної області та зв'язки між ними.

- база даних – сукупність взаємопов'язаних даних, які можна спільно використовувати та керування якими здійснюється централізовано

Відмінності БД від файлових систем:

- взаємопов'язаність даних (файли більш незалежні)
- мінімальна надлишковість (“кожен факт - один раз”)
- незалежність даних від програм
- цілісність даних (відповідність схемі бд, правилам, умовам і тд)
- захист від неавторизованого доступу
- спільне використання даних

## 2. Архітектура ANSI SPARC.



### 3 рівні: концептуальна, внутрішня та зовнішня схеми

- Концептуальна: модель предметної області
- Внутрішня: реалізація концептуальної моделі за допомогою конкретики якоїсь бд
- Зовнішня: зовнішні моделі що надаються користувачам БД

## 3. Особливості та недоліки мережних та ієрархічних БД.

### Ієрархічна:

- особливості:
  - на вищому рівні ієрархії - єдиний кореневий сегмент
  - кожен сегм, крім кореневого, зв'язаний одним і тільки одним батьківським сегментом вищого рівня
  - сегмент зв'язаний з 1 або кількома дочірніми сегментами
- недоліки:
  - асиметрія пошуку за симетричними запитам
  - дублювання даних у разі моделювання зв'язку (багато-до-багато)
  - низький рівень мови запитів
  - аномалії додавання видалення оновлення даних

### Мережева:

- особливості:
  - тип запису- пойменована впорядкована сукупність імен полів
  - тип набору - сукупність зв'язків між двома або кількома типами записів
  - сукупність типів наборів утворює багаторівневу ієрарх або мереж структуру
- недоліки:
  - дублювання даних у разі моделювання зв'язку (багато-до-багато)
  - завелика кількість зв'язків (зв'язків може бути  $n^2$ )

#### **4. Історія розвитку баз даних.**

+1963: ієрархічні БД

+1965: мережеві БД

1970: запропонована реляційна модель даних (запропонував Кодд)


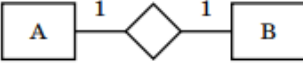

1980: реляційні БД

+1997-1998: стандарт об'єктно орієнтованої моделі і реалізація об'єктно орієнтованої БД

#### **5. Семантичне моделювання: класифікація зв'язків за множинністю та обов'язковістю.**

**Семантичне моделювання** — це спосіб формального опису предметної області, у якому визначаються сутності, їх атрибути та зв'язки між ними, з урахуванням значення (семантики)

- Множинність

Тип зв'язку	Графічне позначення	Зі скількома об'єктами А може бути зв'язано об'єкт В?	Зі скількома об'єктами В може бути зв'язано об'єкт А?
«один-до-багатьох»		з одним	з багатьма
«один-до-одного»		з одним	з одним
«багато-до-багатьох»		з багатьма	з багатьма

- Обов'язковістю (обов'язковість/необов'язковість участі)

Зв'язок між сутностями А і В називають обов'язковим з боку сутності А, якщо кожен об'єкт цієї сутності повинен брати участь у зв'язку. На моделі «сутність-зв'язок» обов'язкові зв'язки позначають подвійними лініями.

## 6. Семантичне моделювання: слабкі сутності, зв'язок «є», тернарний зв'язок.

### Слабкі сутності:

Сутність називається слабкою, якщо їй не вистачає власних атрибутів для утворення ключа. Її ключ утворюють з деяких її власних атрибутів, а також ключів інших сутностей. Зв'язки слабкої сутності з тими сутностями, що доповнюють її ключ, називаються підтримувальними та позначаються подвійними ромбами, а сама слабка сутність — подвійним прямокутником.

### Зв'язок є:

Сутності А і В з'єднані зв'язком «є», що називається також зв'язком «загальний тип – різновид», якщо сутність В — це різновид сутності А. Зв'язок «є» має множинність «один-до-одного» і є обов'язковим з боку сутності-різновиду. Він позначається рівнобедреним трикутником, вершина якого спрямована до загального типу.

### Тернарний зв'язок:

Якщо відносини між трьома сутностями неможливо адекватно зобразити за допомогою кількох бінарних зв'язків, усі три сутності з'єднують одним зв'язком, що називається тернарним і має три множинності.

## 7. Основний принцип семантичного моделювання.

Головний принцип семантичного моделювання. Модель «сутність-зв'язок» повинна дозволити зберігання будь-якого факту лише в одному місці.

## 8. Реляційна модель даних: основні означення, поняття домену, відношення.

**Реляційна модель даних** — це модель представлення даних, у якій вся інформація зберігається у вигляді відношень (таблиць)

**Реляційне відношення** - сукупність заголовка та тіла

**Заголовок**- множина атрибутів

**Тіло** відношення - множина кортежів

**Кортеж** - множина пар вигляду {<ім'я;значення>...<ім'я; значення>} де імена збігаються з іменами атрибутів а значення мають відповідні типи

**Домен**-список допустимих значень для атрибуту

**Відношення**-(підмножина декартового добутку) зв'язок між елементами двох множин

## 9. Синтаксис та семантика операцій реляційної алгебри.

- Об'єднання:  $A \cup B$  (усі кортежі  $A$  та  $B$ . !!!  $A$  і  $B$  мають бути сумісні)
- Перетин:  $A \cap B$  (Кортежі, що належать одночасно  $A$  та  $B$ )
- Різниця:  $A \setminus B$  (Кортежі, що є в  $A$ , але не входять у  $B$ )
- Вибірка:  $A[<\text{умова}>]$  (Вибирає ті кортежі (рядки) , які задовольняють умову)

- Проекція:  $A[\langle \text{список атрибутів} \rangle]$  (Вибирає тільки вказані атрибути з відношення  $A$ . Видаляє повтори)
- Декартів добуток:  $A \times B$  (Поєднує кожен кортеж з  $A$  з кожним кортежем з  $B$ .)
- З'єднання:  $A[\langle \text{умова} \rangle]B$  (Поєднання за умовою)
- Природне з'єднання:  $A * B$  (автоматичне поєднання по спільних атрибутах з однаковими значеннями)
- Ділення:  $A \% B \text{ per } C$  (знаходить ті значення з  $A$ , що повністю охоплюють усі значення з  $B$ )

## 10. Синтаксис та семантика виразів реляційного числення.

Вираз реляційного числення означає:

"Знайти всі кортежі, що задовольняють логічну умову у WHERE"

Реляційний вираз має вигляд:

`<прототип кортежу> WHERE <логічний вираз>`

- Прототип кортежу — перелік атрибутів, які мають бути у результаті (наприклад: `Sx.Sname` )
- Логічний вираз — умова, яку повинні задовольняти дані. Складається з:
  - простих виразів (наприклад: `Sx.STATUS > 10` )
  - або кванторних (наприклад: `∃ Sx(Sx.STATUS > 10)` )
- Змінні можуть бути:
  - вільні (оголошуються в прототипі — результат запиту),
  - зв'язані (оголошуються під квантором  $\exists$  або  $\forall$  — проміжні).

Результат- множина кортежів з атрибутами, вказаними в прототипі

## 11. Порівняння потужності реляційної алгебри та реляційного числення. Поняття про реляційну повноту. Алгоритм редукції Кодда.

## ✚ Реляційна алгебра

- Процедурна мова запитів — описує як отримати результат (через послідовність операцій).
- Використовує операції: проєкція, вибірка, об'єднання, добуток, з'єднання, різниця, ділення тощо.

## ✚ Реляційне числення

- Декларативна мова запитів — описує *що* потрібно отримати (через логічну умову).

## ✚ Потужність (виражальна здатність):

Реляційна алгебра  $\equiv$  реляційне числення

Обидві мови еквівалентні за потужністю: будь-який запит, який можна записати в одній, можна виразити і в іншій.

### Реляційна повнота:

Мова називається реляційно повною, якщо вона не менш потужна, ніж реляційне числення. Редукція Кодда показує, що реляційна алгебра є реляційно повною.

### Алгоритм редукції Кодда:

- 1) Для кожного безкванторного логічного виразу беремо декартів добуток визначених для нього таблиць.
- 2) Сам безкванторний логічний вираз реалізуємо через вибірку.
- 3) Квантори існування перетворюємо на проєкції на всі атрибути, що не належать підкванторним змінним.
- 4) NOT, AND, OR між кванторами виражаємо через теоретико-множинні операції.
- 5) Квантор загальності – через ділення.
- 6) Беремо проєкцію на атрибути, вказані в прототипі кортежу.

## 12. Синтаксис та семантика вибіркового запиту SQL.

Вибірковий запис SQL:

SELECT [DISTINCT] <список атрибутів> - вибір конкретних атрибутів

FROM <таблиця/таблиці> - звідки відбуватиметься вибір

[WHERE <умова відбору>] - умова за якою буде вибір

### 13. Синтаксис та семантика запитів з групуванням у SQL.

SELECT [DISTINCT] <список атрибутів> - вибір конкретних атрибутів  
FROM <таблиця/таблиці> - звідки відбуватиметься вибір  
[WHERE <умова відбору>] - умова за якою буде вибір  
[GROUP BY <список атрибутів групування>] - групування рядків таблиці за значеннями одного або кількох атрибутів  
[HAVING <умова відбору груп записів>] - “фільтрація” груп сформованих GROUP BY

### 14. Агрегатні функції у мові SQL та в реляційному численні.

**Синтаксис:** <агрегатна\_функція>(<атрибут>)

**Агрегатні функції:** COUNT(\*) — кількість рядків, SUM(атрибут) — сума значень, AVG(атрибут) — середнє значення, MAX(атрибут) — максимум ...

### 15. Запити на оновлення, додавання та видалення даних у SQL.

Додавання:

INSERT INTO <таблиця> (<список\_атрибутів>)  
VALUES (<список\_значень>);

Оновлення:

UPDATE <таблиця>  
SET <атрибут1> = <значення1>, <атрибут2> = <значення2>, ...  
[WHERE <умова>];

Видалення:

DELETE FROM <таблиця>  
[WHERE <умова>];



## 16. Визначення реляційної структури в SQL. Оператори створення та змінення таблиць.

**Створення таблиці:** CREATE TABLE

```
CREATE TABLE <ім'я_таблиці> (  
    <ім'я_атрибута> <тип_даних> [обмеження],  
    ...  
);
```

**Зміна таблиці:** ALTER TABLE

ALTER TABLE <таблиця>

ADD <атрибут> <тип\_даних>; - додавання стовпця

DROP COLUMN <атрибут>; - видалення стовпця

**Видалення таблиці:** DROP TABLE

```
DROP TABLE <ім'я_таблиці>;
```

## 17. Реалізація обмежень цілісності мовою SQL. Створення первинного ключа та ключа UNIQUE. Зовнішні ключі. Обмеження CHECK.

PRIMARY KEY- Унікальний і обов'язковий ідентифікатор

FOREIGN KEY- Зовнішній ключ для зв'язку з іншою таблицею

```
CREATE TABLE <ім'я_таблиці> (  
    <ім'я_атрибута> <тип_даних> [обмеження],  
    PRIMARY KEY(<ім'я_атрибута>),  
    FOREIGN KEY(<ім'я_атрибута>) references <Таблиця>  
);
```

NOT NULL Значення обов'язкове (як обмеження)

UNIQUE Значення має бути унікальним (як обмеження)

CHECK Перевірка умов (наприклад, вік > 0)

## 18. Реалізація обмежень цілісності за допомогою тригерів.

Тригери використовуються для реалізації складних обмежень цілісності, що не покриваються стандартними засобами. Вони автоматично перевіряють або змінюють дані при спробах модифікації таблиці.

```
CREATE TRIGGER <ім'я_тригера>  
ON <таблиця>  
AFTER(INSTEAD OF) INSERT | UPDATE | DELETE  
AS  
BEGIN  
    сама логіка  
END;
```

## 19. Поняття та властивості функціональних залежностей. Аксіоми Армстронга.

Функціональна залежність - кожному значенню  $X$  відповідає одне значення  $Y$ . Аналогічно для атрибутів: набір атрибутів  $B$  фз від  $A$ , якщо кожному значенню  $A$  може відповідати тільки одне значення  $B$ :  $A \rightarrow B$

### Аксіоми Армстронга:

- 1) Рефлексивність:  $AB \rightarrow A$ .
- 2) Транзитивність:  $A \rightarrow B, B \rightarrow C \Rightarrow A \rightarrow C$ .
- 3) Доповнення:  $A \rightarrow B \Rightarrow AC \rightarrow BC$ .

### Правила виводу

Декомпозиція:  $A \rightarrow BC \Rightarrow A \rightarrow B \text{ і } A \rightarrow C$ .

$BC \rightarrow B, BC \rightarrow C$  за рефлексивністю (\*)

$A \rightarrow BC \text{ і } (*) \Rightarrow A \rightarrow B \text{ і } A \rightarrow C$  за транзитивністю.

## 20. Незвідне покриття множини функціональних залежностей: означення та алгоритм пошуку.

Множина ФЗ  $F^+$  є замиканням множини ФЗ  $F$ , якщо вона містить всі ФЗ, які випливають з  $F$ .

Множини ФЗ  $F$  і  $G$  є еквівалентними ( $F \sim G$ ), якщо  $F^+ = G^+$ .

Множина ФЗ  $G$  є **незвідним покриттям** множини ФЗ  $F$ , якщо

- 1)  $G \sim F$ .
- 2) Якщо з  $G$  викреслити хоча б одну ФЗ або хоча б 1 атрибут із якоїсь залежності, то властивість 1 втратиться.

Алгоритм:

- 1) За правилом декомпозиції розкладаємо ФЗ так, щоб справа було по 1 атрибуту.
- 2) Намагаємося за аксіомами Армстронга вивести якомога більше одних ФЗ з інших і викреслити їх.
- 3) Переглядаємо всі ФЗ. Нехай розглядається ФЗ  $X \rightarrow Y$ . Якщо  $Y \in X^+(F)$ , де  $F$  - всі інші ФЗ, то  $X \rightarrow Y$  викреслюємо.
- 4) Для кожної ФЗ зліва з кількома атрибутами перевіряємо, чи можна видалити кожен окремий атрибут без втрати залежності. Якщо так — атрибут видаляється.
- 5) Якщо на кроці 4 щось викреслилось, то повторюємо кроки 3 і 4.

## 21. Означення ключа відношення та нормальних форм: 2-ї, 3-ї, нормальної форми Бойса-Кодда.

**Ключем** відношення називається така множина атрибутів  $K$ , що:

- 1) Від  $K$  функціонально залежать всі атрибути відношення.
- 2) Якщо з набору  $K$  вилучити хоча б 1 атрибут, то властивість 1 втратиться.

- **1 НФ** - усі відношення з атомарними значеннями атрибутів.
- Відношення у **2НФ**, якщо жоден з неключових атрибутів не залежить від частини якогось ключа.
- Відношення у **3НФ**, якщо:
  - 1) Воно перебуває у 2НФ.
  - 2) Не містить залежностей між неключовими атрибутами.
- Відношення перебуває у НФ Бойса-Кодда (НФБК), якщо лівою частиною кожної ФЗ є деякий ключ.

## 22. Декомпозиція відношень. Теорема Хіза. Метод таблиці перевірки правильності декомпозиції.

**Декомпозиція відношень** — це розбиття одного відношення на два або більше з метою усунення надлишковості, аномалій і досягнення нормальних форм без втрат

### Теорема Хіза:

Нехай є відношення  $R(A,B,C)$ , де множини атрибутів  $A$ ,  $B$  і  $C$  не перетинаються, і є ФЗ  $A \rightarrow B$ . Тоді  $R$  можна поділити на  $R(A,B)$  і  $R(A,C)$  без втрат даних.

### ▲ Метод таблиці перевірки правильності декомпозиції

Це окремий метод, який не згаданий у твоїй відповіді, а мав би бути.

#### ◆ Ідея:

Метод перевірки, чи є декомпозиція без втрат:

- Створюється таблиця з рядками — по одному на кожне нове підвідношення;
- У таблиці кожен стовпчик — атрибут із початкового відношення;
- Заповнюються символічні значення для перевірки, чи можна відновити оригінальне відношення шляхом з'єднання підвідношень.