Лекція

[**2.1. Концептуальне проектування БД**](https://rdb.dp.ua/uk/chapter_02#2.1)

[**2.2. Історія реляційної моделі**](https://rdb.dp.ua/uk/chapter_02#2.2)

[**2.3. Структура даних у реляційній моделі**](https://rdb.dp.ua/uk/chapter_02#2.3)

[**2.4. Логічне проектування БД**](https://rdb.dp.ua/uk/chapter_02#2.4)

[**2.5. Приклад логічної моделі**](https://rdb.dp.ua/uk/chapter_02#2.5)

[**2.6. КОНТРОЛЬНІ ПИТАННЯ**](https://rdb.dp.ua/uk/chapter_02#2.6)

**Мета**

*Мета розділу – ознайомлення з концептуальним проектуванням БД, обґрунтування вибору реляційної моделі даних та ознайомлення з історією її створення, термінологією та структурою даних, описом відносин на етапі логічного проектування реляційної БД.*

**2.1. Концептуальне проектування БД**

Безпосередньо проектування БД (рис. 1.3) починається з концептуального моделювання. Це найбільш загальний опис, який називається ***схемою БД***. Схема створюється за допомогою мови визначення даних СКБД, вибраної для реалізації проекту. Вона не дозволяє подати дані так, щоб створену схему розуміли користувачі всіх категорій

Для виходу із цієї ситуації дані подають у вигляді ***моделі даних****– інтегрованого набору понять для їхнього опису, зв'язків між ними й обмежень, що накладаються на них* [1, 2, 4, 6, 9 - 11]. Це необхідно для того, щоб усі учасники проекту змогли ознайомитися з набором правил та однозначно розуміти їх, оскільки за ними будуватимуть БД з визначеними типами доступних операцій та сукупністю обмежень, що підтримують цілісність та гарантують коректність використовуваних даних.

***Концептуальне проектування БД*** полягає у виборі або створенні моделі подання даних технологічного процесу, яка не залежить від будь-яких аспектів її фізичного розміщення, організації й обробки даних [4].

Вибір або створення моделі подання даних базується на аналізі вимог користувачів кінцевої системи. Модель не залежить від типу вибраної СКБД, набору прикладних програм, які створені або будуть створені, мов програмування, що використаються, самої обчислювальної платформи та ін. Модель подання даних, яка отримана в результаті концептуального проектування, є джерелом інформації для фази логічного проектування БД. Якщо жодна з існуючих моделей (реляційна, мережна, ієрархічна, об'єктно-орієнтована і т.д.) Вам не підійшла, то у цьому випадку треба створювати власну модель подання даних.

Далі будемо вважати, що за результатами концептуального проектування вибрана реляційна модель виходячи з того, що 80% сучасних СКБД підтримують саме її. Причому багато нереляційних систем тепер забезпечуються реляційним інтерфейсом користувача, незалежно від базової моделі даних, яка використовується.

**2.2. Історія реляційної моделі**

Реляційна модель уперше була описана Є.Ф. Коддом (E.F. Codd) у 1970 році в статті “Реляційна модель даних для великих банків даних, що використовуються спільно”. Цю статтю прийнято вважати важливим етапом в історії розвитку систем БД, хоча слід зазначити, що раніше була запропонована модель, основана на множині (Childs, 1968) [1, 4, 6, 7, 9, 10, 12].

Мета створення реляційної моделі формулювалася так [4]: Цели создания реляционной модели формулировались так [4]:

1. Забезпечення максимально можливого ступеня незалежності прикладних програм від даних.
2. Створення міцного фундаменту для вирішення семантичних питань, а також проблем несуперечності й надмірності даних.
3. Розширення мов керування даними за рахунок включення операцій над множинами.

Важливі дослідження стосовно можливостей реляційної моделі були проведені у рамках виконання трьох проектів.

Перший з них був розроблений наприкінці 70-х років корпорацією IBM у місті Сан-Хосе, штат Каліфорнія, під керівництвом Астрахана (Astrahan). Його результатом стало створення системи за назвою “System R”, що була прототипом дійсно реляційної СКБД [12]. Цей проект був задуманий з метою одержання реальних доказів практичності реляційної моделі. Він став найважливішим джерелом інформації таких проблем реалізації, як керування паралельністю, трансакціями, оптимізація запитів, безпеки і цілісності даних, технологія відновлення, урахування людського фактору й розробка інтерфейсу користувача.

Виконання проекта сприяло появі наукових статей і створенню інших прототипів реляційних СКБД, і все це дозволило [4]:

1. розробити мову керування даними SQL або по літерах “S-Q-L” за допомогою мнемонічного імені “See-Quell”, яка і до тепер має статус формального стандарту ISO (International Organization for Standardization) і є фактичним галузевим стандартом мови реляційних СКБД;
2. створити різні комерційні реляційні СКБД, які вперше з'явилися на ринку на початку 80-х років, наприклад, DB2 і SQL/DS корпорації IBM, а також ORACLE корпорації ORACLE Corporation.

Другим проектом, який відіграв помітну роль у розробці реляційної моделі даних, був INGRESS (Interactive Graphics Retrieval System), причому робота над ним проводилася у Каліфорнійському університеті (місто Берклі) майже у той самий час, що й над проектом System R [12]. Проект INGRESS включав розробку прототипу реляційної СКБД і мав ту саму мету, що й проект System R. Ці дослідження сприяли появі академічної версії INGRESS, яка зробила істотний внесок у загальне визнання реляційної моделі даних. Пізніше від даного проекту відійшли комерційні продукти INGRESS фірми Relational Technology Inc. (тепер Ca-Open lngres фірми Computer Associates) і Intelligent Database Machine фірми Britton Lee Inc.

Третім проектом була система Peterlee Relational Test Vehicle наукового центру корпорації IBM, розташованого у місті Петерлі, Великобританія (Todd, 1976) [12]. Цей проект був більш теоретичним, ніж проекти System R і INGRES. Його результати мали дуже велике й навіть принципове значення, особливо в таких галузях, як обробка запитів і оптимізація, а також функціональні розширення системи [4].

Крім того, пізніше для найбільш повного й точного відображення змісту даних (Codd, 1979), підтримки об'єктно-орієнтованих понять (Stonebraker and Rowe, 1986) та дедуктивних можливостей (Gardarin and Valduriez, 1989), межі реляційної моделі даних були розширені.

Таким чином, комерційні системи на базі реляційної моделі даних стали з'являтися наприкінці 70-х – початку 80-х років. Нині налічується сотень типів різних реляційних СКБД як для мейнфреймів, так і для персональних комп'ютерів, хоча багато з них не зовсім відповідають точному визначенню реляційної моделі даних. Прикладами реляційних СКБД для персональних комп'ютерів є СКБД Oracle та MySQL від Oracle, Access, Foxpro, SQL-Server від Microsoft, InterBase фірми Embarcadero, Rbase фірми Microrim і т.д.

**2.3. Структура даних у реляційній моделі**

**2.3.1. Відношення**

Реляційна модель заснована на математичному понятті відношення [4, 6, 12, 13]. Припустимо, що у нас є дві множини, *D*1 і *D*2, де *D*1={2,4} й *D*2={1,3,5}. ***Декартовим добутком****цих двох множин* (*позначається як* *D*1 × *D*2) *називається набір з усіх можливих пар, у яких першим йде елемент множини* *D*1*, а другим – елемент множини* *D*2. Альтернативний спосіб визначення цього добутку полягає у пошуку усіх комбінацій елементів, де першим йде елемент множини *D*2, а другим – елемент множини *D*1. У результаті одержимо:

*D*1 × *D*2={(2, 1), (2, 3), (2, 5),(4, 1), (4,3), ( 4, 5)}.

Будь-яка підмножина цього декартового добутку є відношенням. Наприклад, у ньому можна виділити відношення *R*:

*R* = {(2, 1), (4, 1)}.

Для визначення тих можливих пар, які будуть входити у відношення, можна задати деякі умови їх вибірки. Наприклад, якщо звернути увагу на те, що відношення *R* містить усі можливі пари, у яких другий елемент рівний 1, то визначення відношення *R* можна сформулювати у такий спосіб:

*R* = {(*x*, *y*),| *x* є *D*1, *y* є *D*2, *y* = 1}.

На основі тих же множин можна сформувати інше відношення – *S*, у якім перший елемент завжди повинен бути у два рази більше другого. Тоді визначення відношення *S* можна сформулювати так:

*S* = {(*x*, *y*),| *x* є *D*1, *y* є *D*2, *x* = 2 × *y*}.

У даному прикладі тільки одна можлива пара даного декартового добутку відповідає цій умові:

*S* = {(2, 1)}.

Поняття відношення можна легко поширити й на три множини. Нехай є три множини: – *D*1, *D*2 і *D*3. Декартів добуток *D*1 × *D*2 × *D*3 цих трьох множин є набір, який складається з усіх можливих трійок елементів, у яких першим йде елемент множини *D*1, другим – елемент множини *D*2, а третім – елемент множини *D*3. Будь-яка підмножина цього декартового добутку є відношенням. Обчислимо декартів добуток трьох множин *D*1={1,3}, *D*2={2,4} і *D*3={5,6}:

*D*1 × *D*2 × *D*3 = {(1, 2, 5), (1, 2, 6), (1, 4, 5), (1, 4, 6), (3, 2, 5), (3, 2, 6), (3, 4, 5), (3, 4, 6)}.

Будь-яка підмножина з наведених вище трійок елементів є відношенням. Збільшуючи кількість множин, можна дати узагальнене визначення відношення на *n* доменах.

Нехай є *n* множин *D*1, *D*2, …, *D*n. Декартів добуток для цих *n* множин можна визначити в такий спосіб:

*D*1 × *D*2 × … × *D*n = {() | *d<*1 є *D*1, *d<*2 є *D*2, … *d<*n є *D*n}.

Звичайно цей вираз записують у такому вигляді:

Будь-яка множина *n*-арних кортежів цього декартового добутку є відношенням *n* множин. Звернемо увагу на те, що для визначення цих відношень необхідно знати множини або домени, з яких будуть відбиратися значення.

Фізичним відображенням відношення є таблиця.

***Відношення****– це плоска таблиця, що складається зі стовпців і рядків.*

У реляційній моделі відношення використовуються для зберігання інформації про суб'єкти (сутності), які розміщені у БД.

**2.3.2. Опис структури відношення**

***Атрибут****– це пойменований стовпець відношення.*

Відношення звичайно має вигляд двовимірної таблиці, у якій рядки відповідають окремим записам, а стовпці – атрибутам. При цьому атрибути можуть розташовуватися у будь-який послідовності – незалежно від їх перепорядкування відношення буде залишатися тим самим, а тому буде мати той самий зміст. Кожний атрибут реляційної БД визначається на деякому домені.

***Домен****– це набір припустимих значень для одного або декількох атрибутів.*

Домен – це найважливіший елемент реляційної моделі. Він може відрізнятися для кожного з атрибутів, але два й більше атрибутів можуть бути визначені за допомогою одного домену. Користувач за його допомогою може централізовано розкривати зміст і джерело значень, які одержують атрибути. Отже, під час виконання реляційної операції системі доступно більше інформації. Це дозволяє їй уникнути семантично некоректних операцій. Наприклад, безглуздо порівнювати прізвище автора з назвою книги, навіть якщо для обох цих атрибутів визначені на доменах, які базуються на рядках символів. З іншого боку, помісячна орендна плата за об'єкт нерухомості й кількість місяців, протягом яких він здавався в оренду, належать різним доменам (перший атрибут має грошовий тип, а другий – цілочисловий). Однак процес множення величин із цих доменів є припустимою операцією. Як випливає з цих двох прикладів, забезпечити повну реалізацію поняття домену зовсім непросто, а тому в багатьох реляційних СКБД цей елемент підтримується не повністю, а лише частково.

Елементами відношення є кортежі або рядки таблиці. Кортежі можуть розташовуватися у будь-якому порядку, при цьому відношення буде залишатися незмінним і відповідно, мати той самий зміст.

***Кортеж****– це рядок відношення.*

Опис структури відношення разом зі специфікацією доменів і будь-якими іншими обмеженнями можливих значень атрибутів іноді називають його заголовком (або змістом (intension)). Звичайно воно є фіксованим, доти, доки його зміст не почне змінюватися за рахунок додавання в нього додаткових атрибутів. Кортежі називаються розширенням (extension), станом (state) або тілом відношення, яке постійно змінюється.

***Ступінь відношення****визначається кількістю атрибутів, яку воно містить.*

Відношення тільки з одним атрибутом має ступінь 1 і називається унарним (*unary*) відношенням (або 1-арним кортежем). Відношення з двома атрибутами називається бінарним (*binary*), відношення з трьома атрибутами – тернарним (*ternary*), а для відношення з більшою кількістю атрибутів використовується термін *n*-арний (*n-агу*). Визначення степеня відношення є частиною заголовка відношення.

***Кардинальність****– це кількість кортежів, яку містить відношення.*

Ця характеристика змінюється при кожному додаванні або вилученні кортежів. Кардинальність є властивістю тіла відношення й визначається поточним станом відношення в довільно взятий момент.

**2.3.3. Реляційна схема**

***Реляційна схема****– це ім'я відношення, за яким йде множина пар імен атрибутів і доменів.*

Наприклад, для атрибутів *A*1, *A*2, …, *A*n з доменами *D*1, *D*2, …, *D*n реляційною схемою буде множина {*A*1:*D*1, *A*2:*D*2, … *A*n:*D*n}. Відношення *R*, яке задане реляційною схемою *S*, є множиною відображень імен атрибутів на відповідні їм домени. Таким чином, відношення *R* є безліччю таких *n*-арних кортежів {*A*1:*d*1, *A*2:*d*2, … *A*n:*d*n}, де {*d*1:*D*1, *d*2:*D*2, … *d*n:*D*n}.

Кожний елемент *n*-арного кортежу складається з атрибута та його значення. Звичайно при записі відношення у вигляді таблиці імена атрибутів наводяться у заголовках стовпців, а кортежі утворюють рядки формату *d*1, *d*2, …, *d*n, де кожне значення береться з відповідного домену. Таким чином, у реляційній моделі відношення можна розглядати як довільну підмножину декартового добутку доменів атрибутів, тоді як таблиця – це всього лише фізичне відображення такого відношення.

**2.3.4. Властивості відношень**

1. Відношення має унікальне ім'я в БД.
2. Кожна чарунка відношення містить тільки неподільне значення.
3. Кожний атрибут має унікальне ім'я.
4. Значення атрибута беруться з одного домену.
5. Послідовність розташування атрибутів не має ніякого значення.
6. Кожний кортеж відношення є унікальним.
7. Теоретично послідовність розташування кортежів у відношенні не має значення. (Однак на практиці вона може суттєво вплинути на ефективність доступу до даних.)

Більша частина властивостей відношень БД походить від властивостей математичних відношень.

1. Оскільки відношення є множиною, то послідовність розташування в ньому елементів не відіграє ніякої ролі. Отже, порядок кортежів у відношенні несуттєвий.
2. У множині не має бути повторюваних елементів. Аналогічно, відношення не може містити кортежів-дублікатів.
3. При обчисленні декартового добутку множин з простими однозначними елементами (наприклад, цілочисловими значеннями), кожен елемент у кожному кортежі має єдине значення. Аналогічно, кожна чарунка відношення містить тільки одне значення. Однак математичне відношення не потребує нормалізації. У своїй роботі Кодд пропонує заборонити використання груп, що повторюються, з метою спрощення реляційної моделі даних.
4. Набір можливих значень для даної позиції відношення визначається множиною або доменом. У відношенны БД усі значення у кожному атрибуті мають походити від того домену, на якому визначено атрибут.

Однак у математичному відношенні послідовність розташування елементів у кортежі має велике значення. Наприклад, припустима пара значень (1, 2) зовсім відмінна від припустимої пари (2, 1). Це твердження невірне для відношень у реляційній моделі, де спеціально обумовлюється, що послідовність розміщення атрибутів не є суттєвою. Однак, якщо структура відношення вже визначена, то послідовність розміщення елементів у кортежах його тіла має відповідати послідовності імен атрибутів.

**2.3.5. Реляційні ключі**

Реляційні ключі потрібні для унікальної ідентифікації кожного окремого кортежу відношення за значеннями його атрибутів.

***Суперключ (super key)****– це атрибут або множина атрибутів, яка єдиним чином ідентифікує кортеж даного відношення.*

Оскільки суперключ може містити додаткові атрибути, які необов'язкові для унікальної ідентифікації кортежу, нас будуть цікавити суперключі, що складаються тільки з тих атрибутів, які дійсно необхідні для унікальної ідентифікації кортежів. Таким чином, суперключ може складатися з декількох потенційних ключів.

***Потенційний ключ****– це суперключ, який не містить підмножини, що також є суперключем даного відношення. Відношення може мати декілька потенційних ключів.*

Якщо ключ складається з декількох атрибутів, то він називається складеним ключем.

Потенційний ключ для даного відношення володіє двома властивостями.

***Унікальність****. У кожному кортежі відношення значення ключа єдиним чином ідентифікують цей кортеж.*

***Мінімальність****. Жоден з атрибутів не може бути виведений з ключа без порушення унікальності.*

Зверніть увагу на те, що будь-який конкретний набір кортежів відношення не можна використовувати для доказу того, що якийсь атрибут або комбінація з атрибутів є потенційним ключем. Той факт, що в деякий момент часу не існує значень-дублікатів, зовсім не означає, що їх не може бути взагалі. Однак наявність значень-дублікатів у конкретному існуючому наборі кортежів цілком може бути використана для демонстрації того, що деяка комбінація з атрибутів не може бути потенційним ключем. Для ідентифікації потенційного ключа потрібно знати значення атрибутів в "реальному світі". Тільки це дозволить обґрунтовано прийняти рішення щодо можливого існування значень-дублікатів. Виходячи тільки з подібної семантичної інформації, можна гарантувати, що деяка комбінація з атрибутів є потенційним ключем відношення.

***Первинний ключ****– це потенційний ключ, який вибраний для унікальної ідентифікації кортежів усередині відношення.*

Оскільки відношення не містить кортежів-дублікатів, то завжди можна унікальним чином ідентифікувати кожний його рядок. Це значить, що відношення завжди має первинний ключ. У найгіршому випадку вся множина атрибутів може використовуватися як первинний ключ, але звичайно, щоб розрізнити кортежі, достатньо використати трохи меншу підмножину атрибутів. Потенційні ключі, які не вибрані як первинний ключ, називаються альтернативними ключами.

***Зовнішній ключ****– це атрибут або множина атрибутів усередині відношення, який відповідає потенційному ключу якогось (може бути, того самого) відношення.*

Відношення, де перебуває потенційний ключ, називають базовим, а іноді цільовим або батьківським відношенням. Відношення із зовнішнім ключем називають дочірнім відношенням.

**2.3.6. Реляційна БД**

***Реляційна БД****– набір нормалізованих відношень.*

Реляційна БД складається з відношень, структура яких визначається за допомогою особливих методів, які названі нормалізацією (normalization).

**2.3.7. Опис структури реляційних даних на різних етапах проектування**

На концептуальному рівні описується модель представлення даних у вигляді строгих математичних термінів реляційної алгебри, які визначають їхню структуру, методи керування й методи забезпечення цілісності даних. У будь-якій реляційній СКБД передбачається, що користувач сприймає БД як набір зв'язаних між собою таблиць. Таке сприйняття не стосується фізичної структури БД, де мова йде про її реалізацію на вторинних носіях із зазначенням структур зберігання та методів доступу, які використовуються для їх ефективної обробки. На етапі фізичного проектування відношення називається ***файлом*** (*file*), кортежі – ***записами*** (*records*), а атрибути – ***полями*** (*fields*) і т.д. (табл. 2.1).

Таблиця 2.1

|  |
| --- |
| Терміни, що описують структуру даних у реляційній моделі |
| **Концептуальне проектування** | **Логічне проектування** | **Фізичне проектування** |
| Відношення | Таблиця | Файл |
| Кортеж | Рядок | Запис |
| Атрибут | Стовпець | Поле |
| Степінь відношення | Кількість стовпців | Кількість полів |
| Кардинальність | Кількість рядків | Кількість записів |
| Ключ | Ключ | Індекс |

**2.4. Логічне проектування БД**

***Логічне проектування БД****це процес створення схеми БД (логічної моделі БД) з урахуванням обраної моделі представлення даних, але незалежної від типу цільової СУБД та інших фізичних аспектів реалізації* [4, 6].

Мета логічного проектування полягає у створенні логічної моделі БД для частини підприємства, яка досліджується. Модель подання даних, яка отримана на етапі концептуального проектування, є основою логічної моделі даних, що враховує особливості бізнес-процесів організації та їх реалізації у вибраній СКБД. Однак, на цьому етапі ігноруються всі інші аспекти вибраної СКБД – наприклад, будь-які особливості фізичної організації її структур зберігання даних та побудови індексів.

Логічна модель даних є джерелом інформації для етапу фізичного проектування та забезпечує розроблювача фізичної БД засобами знаходження компромісів, які необхідні для досягнення поставлених цілей. Вона також відіграє важливу роль на етапі експлуатації та супроводження вже готової системи. При правильно організованому супроводі, логічна модель даних, яка підтримується в актуальному стані, дозволяє точно й наочно уявити собі будь-які внесені в БД зміни, а також оцінити їх вплив на прикладні програми і дані, які вже наявні у БД.

Логічна модель, що відображує особливості функціонування підприємства одночасно багатьох типів користувачів, називається ***глобальною логічною моделлю даних***. Існує два основних підходи створення глобальної логічної моделі даних – це централізований та на основі інтеграції уявлень.

***Централізований підхід*** полягає у злитті вимог окремих користувачів, що виражені у вигляді їхніх різних уявлень, у єдиний набір вимог для усіх користувачів, який використовується для створення глобальної логічної моделі даних.

Характерною рисою централізованого підходу є те, що списки вимог складаються до створення глобальної логічної моделі даних. Цей підхід застосовується тільки за умови, що БД не занадто велика або складна.

***Метод інтеграції уявлень*** полягає у злитті окремих локальних логічних моделей даних, що відображають уявлення різних груп користувачів, у єдину глобальну логічну модель даних.

Цей підхід більш керований, оскільки уся робота попередньо розподіляється на дрібні та легко контрольовані частини. Труднощі, звичайно, виникають лише при спробах злиття локальних моделей даних, що створені різними розробниками, які можуть використовувати різні терміни для однакових понять або, навпаки, використовувати один термін для різних понять.

Логічне проектування БД – найважливіший етап, що забезпечує загальний успіх розробки системи в цілому. Якщо створений проект не є точним відображенням методів роботи та структури підприємства, то буде дуже складно, якщо взагалі можливо, визначити усі уявлення (зовнішні схеми), які необхідні користувачам, або організувати підтримку цілісності БД. Крім того, можуть виникнути труднощі з фізичною реалізацією БД або забезпеченням прийнятної продуктивності системи. У той самий час здатність адаптації до змін є ознакою вдало спроектованої БД. Отже, завжди має сенс витратити необхідний час та енергію на створення найкращої з можливих логічної моделі.

**2.5. Приклад логічної моделі**

Припустимо, що Вам треба створити БД, яка оперативно може надавати інформацію про відвідування студентами лекцій. Для цього необхідно знати скільки годин кожен із студентів пропустив по предметах, які їм викладають.

Поділимо інформацію, яка міститься у БД, на два типи: ***довідкову*** та ***оперативну***. Виходячи з цього у нашій БД обліку відвідування лекцій буде дві довідкові таблиці: «Студенти» і «Предмети», та одна таблиця з оперативною інформацією «Відвідуваність», де будемо вести облік (рис. 2.1). Залишається додати, що спочатку необхідно вносити дані в таблиці, які відносяться до довідкової інформації, а потім – до оперативної.



Рис. 2.1. БД обліку відвідуваності занять

Зверніть увагу на те, що у стовпці «№ залік. кн.» таблиці «Відвідуваність» знаходяться тільки значення, що містяться в стовпці «№ залік. кн.» таблиці «Студенти», а в стовпці «Шифр пр.» таблиці «Відвідуваність» – тільки значення, що містяться в стовпці «Шифр» таблиці «Предмети». Таким чином, можна за № залікової книжки з таблиці «Відвідуваність» установити прізвище студента, а за шифром предмета визначити його назву. З іншого боку, можна сказати, хто зі студентів не пропустив жодної лекції, і який предмет відвідують усі студенти. Спробуйте назвати прізвище сумлінного студента самостійно, та лекції з якого предмету ніхто не пропустив.

При проектуванні завжди дивіться на БД, що створюєте, з двох позицій: з боку розробника та з боку користувача. Для розробника важливо без проблем вносити в БД інформацію та подавати її в тому вигляді, який влаштовує користувача. Для користувача важливо одержувати інформацію в зручній для нього формі. Дотримати баланс цих інтересів непросто. Особливо, коли є ліміт часу, за який користувач прагне одержати необхідні дані.

На рис. 2.1. наведений вигляд БД з боку розробника. Спробуйте самостійно зобразити таблицю, за допомогою якої Вам зручно було б уявити скільки лекцій та з яких предметів пропущено кожним із студентів. Це питання ми розглянемо далі, коли будемо вивчати «Представлення даних».

Така простота не повинна Вас уводити в оману. Тільки досвід створення БД допоможе Вам виробити свої власні прийоми й підходи. У запропонованому курсі розглянуті концепції й прийоми проектування та створення БД, з якими корисно ознайомитися та взяти їх за основу.

**2.6. КОНТРОЛЬНІ ПИТАННЯ**

1. Мета та сутність концептуального проектування БД?
2. На яких складових базується реляційна структура даних?
3. Як пов’язані між собою декартів добуток та двовимірна таблиця?
4. Що складає структуру відношення та які характеристики воно має?
5. Які властивості відношень Вам відомі?
6. Дайте визначення відомим Вам реляційним ключам.
7. Які властивості потенційного ключа Вам відомі?
8. Мета та сутність логічного проектування БД?
9. Порівняйте між собою відомі Вам підходи створення логічної моделі БД.

**Висновок**

*Концепція реляційної організації даних на практиці довела свою життєздатність, залишивши за собою близько 80% світового ринку СКБД. Гнучкі підходи до логічного моделювання реляційних БД надають широкі можливості наочно реалізувати як прості, так і складні бізнес-процеси для широкого кола користувачів різної кваліфікації.*

https://rdb.dp.ua/uk/chapter\_02