**Лекція №4. Реляційна модель даних.**

**Модель даних** - це сукупність структури даних, операцій над ними (операції маніпулювання даними) та обмежень цілісності:

* в який спосіб відбувається об’єднання даних у структури,
* які існують обмеження на значення даних
* як здійснюється оперування цими даними

**Реляційна модель даних** — це модель даних, де текстова чи числова інформація, що подається за допомогою таблиць. Кожна таблиця, яка називається відношенням, складається з рядків, які називаються *кортежами*, та стовпчиків, які називаються *атрибутами*. Реляційна модель визначає представлення даних (структура), захищеність від некоректних змін (цілісність) та операції, що можуть бути виконані з даними (операції з даними).

Назва «реляційна» походить від лат. relativus, – *відносний*, оскільки реляційна модель даних базується на відношеннях між записами в таблиці.

Реляційна модель даних була винайдена британським ученим Едгаром Франком Коддом (23 серпня 1923 -18 квітня 2003 року), який вперше використав для моделювання даних математичні принципи теорії множин і математичної логіки. Він довго працював над розробкою нової теорії організації даних. Зрештою, у 1970 р. Кодд написав наукову статтю «Реляційна модель даних для великих спільних обсягів даних». IBM, де на той час працював Кодд, спочатку відмовилася від реалізації реляційної моделі, однак науковець зумів довести ефективність даної моделі через клієнтів IBM. Освоювання відбувалося дуже повільно і IBM задіяла нових розробників, які створили власну нереляційну мову SEQUEL, яка згодом була перейменована в SQL.

Реляційна модель даних - створена Едгаром Коддом логічна модель даних, що описує:

* структури даних у вигляді (що змінюються в часі) наборів відносин;
* теоретико-множинні операції над даними: об'єднання, перетин різницю і декартове множення;
* спеціальні реляційні операції: селекція, проекція, з'єднання і поділ;
* спеціальні правила, що забезпечують цілісність даних.

Реляційна модель даних - це спосіб розгляду даних, тобто припис для способу представлення даних (за допомогою таблиць) і для способу роботи з таким поданням (за допомогою операторів). Вона пов'язана з трьома аспектами даних: структурою (об'єкти), цілісністю і обробкою даних (оператори).

Сьогодні реляційну модель використовують майже всі комерційні бази даних та їх СКБД. У 2002 р. журнал Forbes помістив реляційну модель даних в список найважливіших інновацій останніх 85 років.

Цілі створення реляційної моделі даних:

* забезпечення більш високого ступеня незалежності від даних;
* створення міцного фундаменту для вирішення семантичних питань і проблем несуперечності і надмірності даних;
* розширення мов управління даними за рахунок включення операцій над множинами.

***Вимоги до таблиць реляційної моделі***

Реляційна модель зображає об’єкти та зв’язки між ними у вигляді таблиць. Кожна таблиця повинна задовольняти певні вимоги:

* будь-який елемент таблиці є одним елементом даних, тому значення у таблиці повинні бути одиночними;
* взaємозв'язки подаються як oб'єкти;
* таблиця відображає один об’єкт і складається з кортежів та атрибутів;
* кожен стовпчик таблиці має унікальне ім’я;
* усі записи у стовпці мають бути одного типу;
* кожна таблиця повинна містити первинний ключ – поле чи декілька полів, що ідентифікують кожен рядок таблиці;
* у таблиці не може бути двох однакових кортежів;
* кардинальне число — кількість кортежів;
* ступінь відносини — кількість атрибутів.

***Об’єкти реляційної моделі***

Реляційна модель даних передбачає структуру даних, обов'язковими об'єктами якої є:

• відношення;

• атрибут;

• домен;

• кортеж;

• ступінь;

• кардинальність;

• первинний ключ.

Відношення - це плоска (двовимірна) таблиця, що складається з стовпців і рядків:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Прізвище | Ім'я | Посада | р.н. |
| 1 | Петров | Ігор | директор | 1968 |
| 2 | Іванов | Олег | юрист | 1973 |
| 3 | Кім | Олена | Бухгалтер | 1980 |
| 4 | Сенін | Ілля | Менеджер | 1981 |
| 5 | Васін | Сергій | Менеджер | 1978 |

Атрибут - це пойменований стовпець відносини.

Домен - це набір допустимих значень для одного або декількох атрибутів.

Кортеж - це рядок відносини.

Ступінь визначається кількістю атрибутів, яке воно містить .

Кардинальність - це кількість кортежів, яке містить відношення.

Первинний ключ - це унікальний ідентифікатор для таблиці.

***Відносини і їх реалізація в реляційної моделі даних*** *.*

Ставлення *R* на безлічі доменів *D* 1 , *D* 2 , ..., *D* n - це підмножина декартова множення цих доменів:

*R* ⊆ *D* 1 × *D* 2 × ... × *D* n

*Приклад.*Визначено домени: *D*1 - безліч прізвищ викладачів, *D*2 - безліч аудиторій, *D*3 - безліч навчальних груп, *D*4 - безліч навчальних дисциплін. Записати відносини: 1) закріплення викладачів за навчальними курсами; 2) розклад занять в групах.

*Рішення*.

1) закріплення викладачів за навчальними курсами:

*R* ⊆ *D* 1 × *D* 4.

Це відношення визначає безліч викладачів, які ведуть безліч навчальних дисциплін.

2) розклад занять в групах:

*R* ⊆ *D* 2 × *D* 3 × *D* 4.

Це відношення визначає безліч аудиторій, в яких проводяться заняття з безлічі навчальних дисциплін для безлічі навчальних груп.

*Властивості відносин:*

* унікальне ім'я відносини;
* унікальне ім'я атрибута;
* немає однакових кортежів;
* кортежі не впорядковані зверху вниз;
* атрибути не впорядковані зліва направо;
* всі значення атрибутів атомарні (нормалізоване відношення).

Таким чином, реляційна база даних - це набір нормалізованих відносин. Для того, щоб перейти до видів відносин, введемо поняття змінної відносини. Змінна відносини - це іменований об'єкт, значення якого може змінюватися з плином часу. Змінна відносини в різний час - це різні таблиці бази даних, у яких різні рядки, але однакові стовпці.

*Види відносин:*

* іменоване відношення;
* базове ставлення;
* похідне відношення;
* ставлення;
* уявлення/представлення (view);
* знімки (snapshot);
* результат запиту;
* проміжний результат.

*Іменоване відношення* - це змінна відносини, певна в СУБД (системи управління базами даних) за допомогою оператора **CREATE** (CREATE TABLE, CREATE BASE RELATION, CREATE VIEW, CREATE SNAPSHOT).

*Базове відношення* - це іменоване відношення, яке не є похідним. Існування базового відношення не залежить від існування інших відносин.

*Похідне відношення* - це відношення, яке визначено через інші іменовані відносини. Похідне відношення залежить від існування інших - базових - відносин.

*Представлення* - це представлення, яке можна отримати з набору іменованих відносин за допомогою деякого реляційного вираження. Кожне іменоване відношення є висловлюваним відношенням, але не навпаки. Приклади які висловлюються відношення - базові відносини, уявлення, знімки, проміжні та кінцеві результати. Безліч всіх відносин - це безліч всіх базових і всіх похідних відносин.

*Подання/представлення* - це іменоване похідне відношення. Представлені в базі даних у вигляді визначення. Подання не зберігається у фізичній пам'яті системи управління базою даних (СКБД), а формується з використанням інших іменованих відносин.

*Знімки* (snapshot) - це те саме, що і уявлення, але з фізичним збереженням і з періодичним оновленням.

*Результат запиту* - це неіменоване похідне відношення. СУБД не забезпечує постійного існування результатів запитів. Для збереження результату запиту його можна привласнити будь-якому іменованому відношенню.

*Проміжний результат* - це неіменоване похідне відношення, яке є результатом підзапиту, вкладеного в більший вираз.

***Склад реляційної моделі***

До складу реляційної моделі даних зазвичай включають теорію нормалізації. Крістофер Дейт визначив три складові частини реляційної моделі даних:

* структурна
* маніпуляційна
* цілісна

**Структурна частина** моделі визначає, що єдиною структурою даних є нормалізоване n-арне відношення. Відношення зручно представляти у формі таблиць, де кожен рядок є кортеж, а кожен стовпець — атрибут, визначений на деякому домені. Даний неформальний підхід до поняття відношення дає більш звичну для розробників і користувачів форму представлення, де реляційна база даних являє собою кінцевий набір таблиць.

**Маніпуляційна частина** моделі визначає два фундаментальних механізми маніпулювання даними — реляційну алгебру і реляційне числення. Основною функцією маніпуляційної частини реляційної моделі є забезпечення заходів реляційності будь-якої конкретної мови реляційних БД: мова називається реляційною, якщо вона має не меншу виразність і потужність, ніж реляційна алгебра або реляційне числення.

**Цілісна частина** моделі визначає вимоги цілісності сутностей і цілісності посилань. Перша вимога полягає в тому, що будь-який кортеж будь-якого відношення відмінний від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення має володіти первинним ключем. Вимога цілісності щодо посилань, або вимога зовнішнього ключа полягає в тому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, на яке веде посилання, повинен знайтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинно бути невизначеним (тобто ні на що не вказувати).

Можна провести аналогію між елементами реляційної моделі даних і елементами моделі «сутність-зв'язок». Реляційні відносини відповідають наборам сутностей, а кортежі — сутностям. Тому, як і в моделі «сутність-зв'язок», стовпці в таблиці, що представляє реляційне відношення, називають атрибутами.

Кожен атрибут визначений на домені, тому домен можна розглядати як множина допустимих значень даного атрибуту. Кілька атрибутів одних відношень і навіть атрибути різних відношень можуть бути визначені на одному і тому ж домені.

Іменована множина пар «ім'я атрибута — ім'я домену» називається схемою відношення. Потужність цієї множини — називають ступенем чи «арністю» відносини. Набір іменованих схем відносин являє собою схему бази даних.

Атрибут, значення якого однозначно ідентифікує кортежі, називається ключовим (або просто ключем). Якщо кортежі ідентифікуються тільки зчепленням значень декількох атрибутів, то говорять, що відношення має складовий/складений ключ. Відношення може містити кілька ключів. Завжди один із ключів оголошується первинним, його значення не можуть оновлюватися. Всі інші ключі відносини називаються можливими ключами.

На відміну від ієрархічної і мережної моделей даних в реляційної відсутнє поняття групових відношень. Для відображення асоціацій між кортежами різних відносин використовується дублювання їх ключів.

***Переваги і недоліки реляційної моделі***

**Переваги реляційної моделі**

Переваги реляційної моделі:

* простота і доступність для розуміння користувачем. Єдиною використовуваною інформаційною конструкцією є «таблиця»;
* суворі правила проектування, які базуються на математичному апараті;
* повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;
* для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті.

Перевагами реляційної моделі є те, що вона спрямована на опрацювання документа в цілому, а не окремих його частин. Вона не потребує знань програмування від користувачів. Для використання реляційної моделі достатньо навичок роботи на ПК та знань основ інформаційних технологій. Тому реляційна модель є простою у розумінні та використанні.

**Недоліки реляційної моделі**

Недоліки реляційної моделі:

* далеко не завжди предметна область може бути представлена у вигляді «таблиць»;
* в результаті логічного проектування з'являється множина «таблиць». Це призводить до труднощів розуміння структури даних;
* БД займає відносно багато зовнішньої пам'яті;
* відносно низька швидкість доступу до даних.

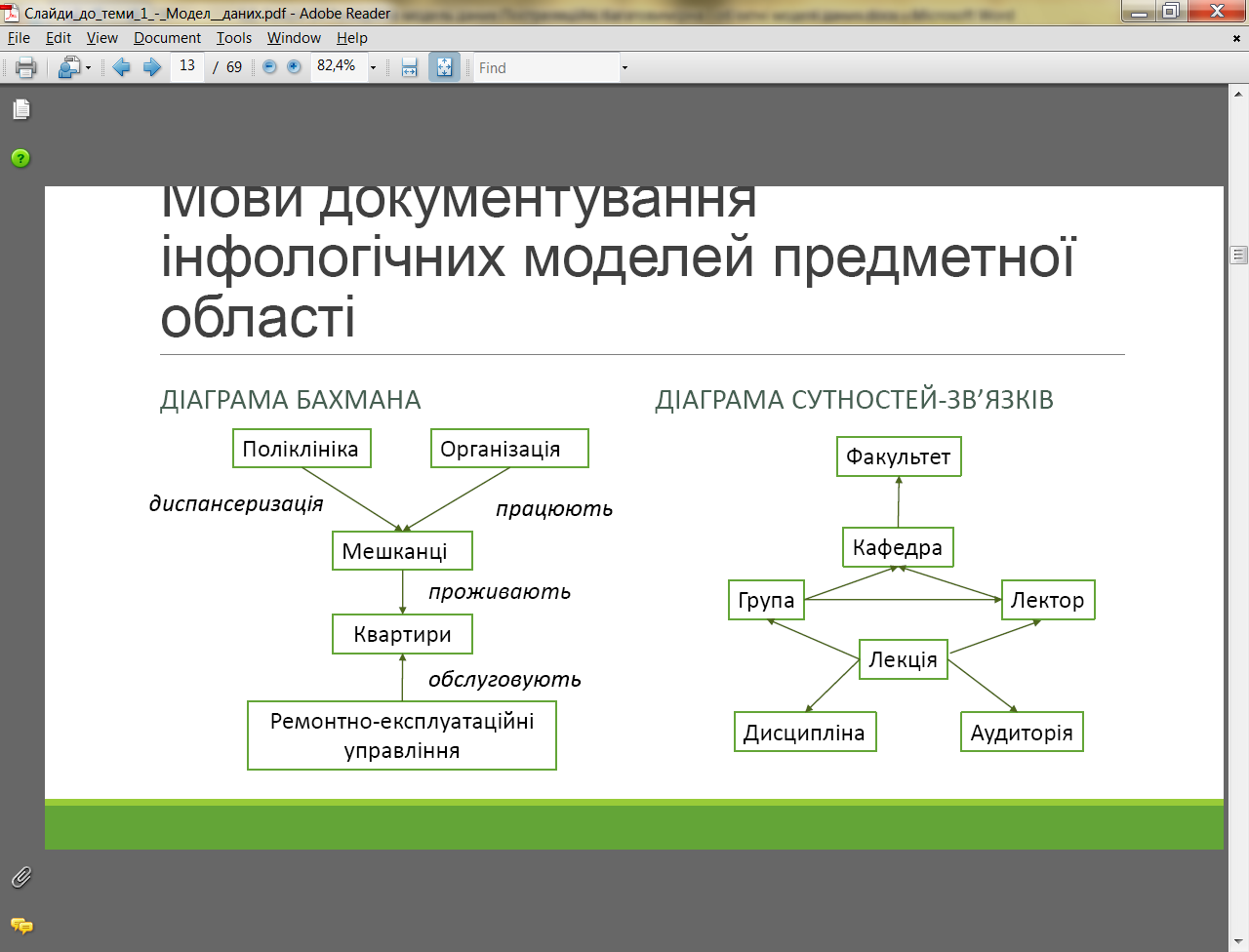
Недоліками реляційної моделі є відносно низька швидкість доступу до даних та використання великого обсягу пам’яті на носіях інформації.

***Відмінності реляційної моделі від інших моделей***

Реляційні, ієрархічні та мережеві моделі даних належать до моделей на основі записів.

Основні відмінності:

* реляційна здійснює зв’язок і пошук між об’єктами зазначенням ключових атрибутів, а ієрархічні та мережева – за структурою;
* у реляційній вищий рівень абстракції даних, ніж в ієрархічній або мережевій;
* простота структур даних в реляційній;
* табличне представлення даних у реляційній, за допомогою графів – у ієрархічній та мережевій моделях;
* в реляційній немає поняття групових відношень, притаманне для ієрархічної та мережевої моделей: для відображення асоціацій між кортежами різних відносин використовується дублювання їх ключів;
* у реляційній моделі подання даних не залежить від способу їх фізичної організації, на відміну від ієрархічної або мережевої;
* реляційна модель дозволяє реалізовувати легкі для створення персональні бази даних, створення яких не передбачене ієрархічною чи мережною моделями взагалі;
* простота та зручність доступу до даних у реляційній моделі.



***Модель вкладених множин*** - техніка для представлення дерев в реляційних базах даних. Стандартна реляційна алгебра та побудовані на ній SQL операції не можуть бути застосовані для всіх потрібних маніпуляцій з деревами (ієрархіями). Якщо дерево має довільну глибину, це не дозволяє використовувати SQL вирази для таких операцій, як порівняння місця в ієрархії для двох елементів або визначення належності елемента до певного під-дерева.

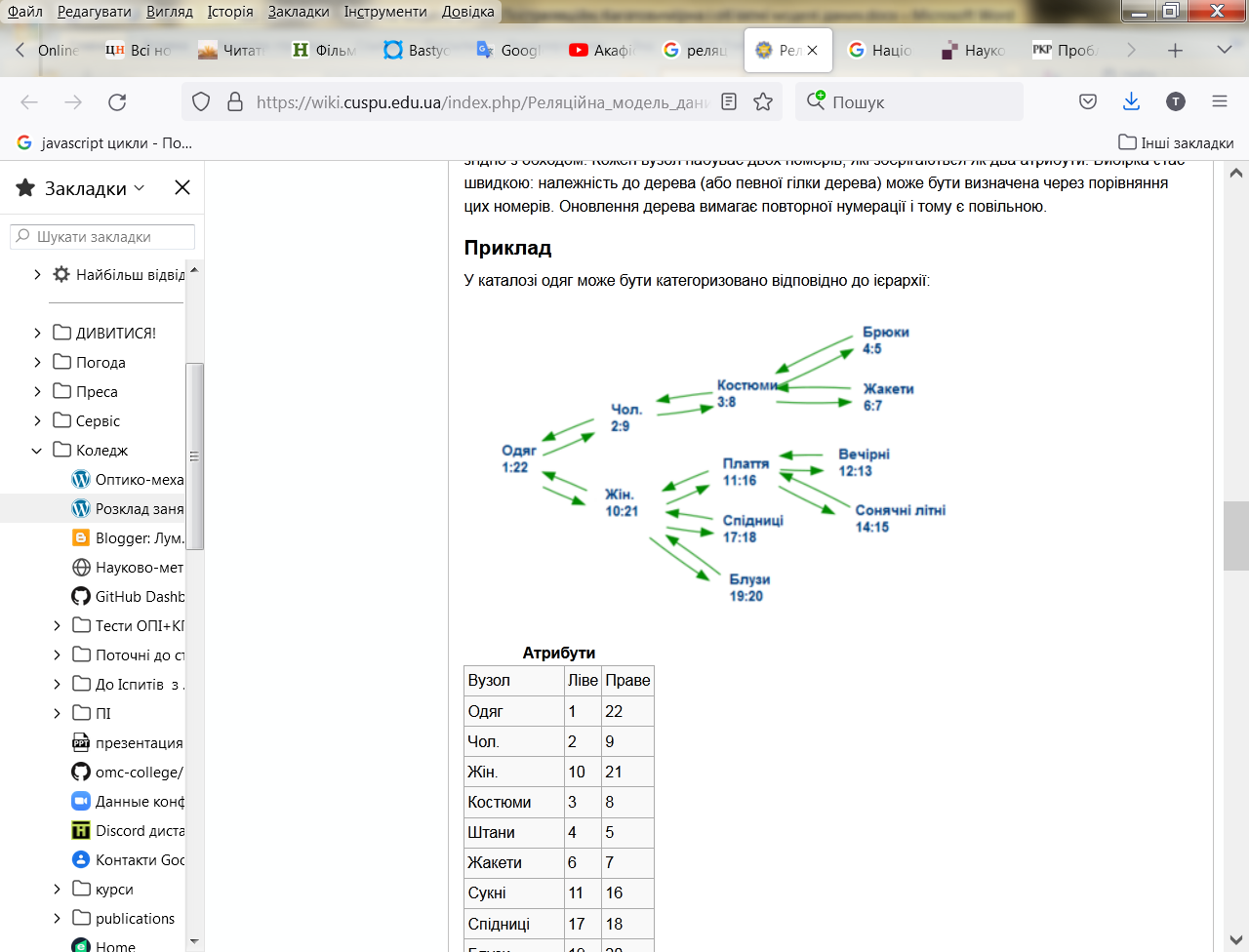
Існує декілька підходів для вирішення проблеми і деякі є доступними в системах керування базами даних:

* підтримка ієрархічних типів даних;
* розширення SQL для маніпуляцій з деревами;
* SQL запити можуть бути виражені мовою програмування, яка підтримує ітерації та дозволяє виконувати реляційні операції, як от [PL/SQL](https://wiki.cuspu.edu.ua/index.php?title=PL/SQL&action=edit&redlink=1), [T-SQL](https://wiki.cuspu.edu.ua/index.php?title=T-SQL&action=edit&redlink=1), або майже будь-якою сучасною мовою програмування;

Техніка **моделі вкладених множин** полягає в нумерації вузлів відповідно до обходу дерева. Кожен вузол оброблюється двічі, кожному вузлу надається номер, відповідний до порядкового номеру згідно з обходом. Кожен вузол набуває двох номерів, які зберігаються як два атрибути. Вибірка стає швидкою: належність до дерева (або певної гілки дерева) може бути визначена через порівняння цих номерів. Оновлення дерева вимагає повторної нумерації і тому є повільною.

**Приклад**

У каталозі одяг може бути категоризовано відповідно до ієрархії:



|  |  |  |
| --- | --- | --- |
| Атрибути | | |
| Вузол | Ліве | Праве |
| Одяг | 1 | 22 |
| Чол. | 2 | 9 |
| Жін. | 10 | 21 |
| Костюми | 3 | 8 |
| Штани | 4 | 5 |
| Жакети | 6 | 7 |
| Сукні | 11 | 16 |
| Спідниці | 17 | 18 |
| Блузи | 19 | 20 |
| Вечірні | 12 | 13 |
| Сонячні літні | 14 | 15 |

Категорія "Одяг", яка має найвищу позицію в ієрархії, включає в себе всі підкатегорії. Атрибути мають значення 1 та 22, останнє значення дорівнює числу вузлів помноженому на 2. Наступний рівень ієрархії включає категорії "Чол." та "Жін.", обидва включають піддерева. На кожному рівні вузлу призначено "праве" та "ліве" значення відповідно до кількості вкладених вузлів. Таким чином, щоб вирахувати чи належить, наприклад, категорія "Блузи" до жіночого одягу треба порівняти відповідні атрибути "Ліве" та "Праве".

***Види ключів***

*Первинний ключ*

Первинний ключ — атрибут, або набір атрибутів, що однозначно ідентифікує кортеж даного відношення. Первинний ключ обов'язково унікальний, він єдиний і найголовніший із унікальних ключів.

В реляційних базах даних первинний ключ задається обмеженням PRIMARY KEY.

*Вторинний ключ*

Зовнішній (вторинний) ключ — це одне або кілька полів (стовпців) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об’єднання таблиць.

З двох логічно пов’язаних таблиць одну називають таблицею первинного ключа або головною таблицею, а іншу таблицею вторинного (зовнішнього) ключа або підпорядкованою таблицею. СУБД дозволяють зіставити споріднені записи з обох таблиць і спільно вивести їх у формі, звіті або запиті.

Існує три типи первинних ключів: ключові поля лічильника (лічильник), простий ключ і складовий/складений ключ.

Поле лічильника (Тип даних «Счетчик»). Для кожного запису цього поля таблиці автоматично заноситься унікальне числове значення.

Простий ключ. Якщо поле містить унікальні значення, такі як коди чи інвентарні номери, то це поле можна визначити як первинний ключ. В якості ключа можна визначити всі поля, що містить дані, якщо це поле не містить повторювані значення або значення Null.

Складові / складені ключі. У випадках, коли неможливо гарантувати унікальність значень кожного поля, існує можливість створити ключ, що складається з декількох полів. Найчастіше така ситуація виникає для таблиці, використовуваної для зв’язування двох таблиць відношенням «багато — до — багатьох».

*В полі первинного ключа повинні бути тільки унікальні значення в кожному рядку таблиці, тобто збіг не допускається, а в полі вторинного або зовнішнього ключа збіг значень у рядках таблиці допускається*.

Якщо виникають труднощі з вибором потрібного типу первинного ключа, то в якості ключа доцільно вибрати поле лічильника.

***Нормалізація баз даних***

***Нормалізація схеми бази даних*** — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Для роботи з реляційними БД застосовують реляційні СУБД. Інакше кажучи, реляційна база даних — це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

Метою нормалізації є усунення недоліків структури БД, які призводять до шкідливої надмірності в даних, яка в свою чергу потенційно призводить до різних аномалій і порушень цілісності даних.

Теоретики реляційних баз даних у процесі розвитку теорії виявили та описали типові приклади надмірності і способи їхнього усунення.

***Нормальні форми***

Нормальна форма — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення. Тобто це - формальна властивість відношення, яка характеризує ступінь надмірності збережуваних даних і можливі проблеми. Кожна наступна нормальна форма в нижченаведеному списку (крім ДКНФ) в деякому сенсі є досконалішою, ніж попередня, з точки зору усунення надмірності.

* Перша нормальна форма (1НФ, 1NF)
* Друга нормальна форма (2НФ, 2NF)
* Третя нормальна форма (3НФ, 3NF)
* Нормальна форма Бойса — Кодда (НФБК, BCNF)
* Четверта нормальна форма (4НФ, 4NF)
* П'ята нормальна форма (5НФ, 5NF)
* Доменно-ключова нормальна форма (ДКНФ, DKNF).

Таким чином, схема [реляційної бази даних](https://wiki.cuspu.edu.ua/index.php?title=%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85&action=edit&redlink=1) переходить у першу, другу, третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми *n* та всіх попередніх нормальних форм, тоді вважається, що це відношення знаходиться у нормальній формі рівня *n*.

**Перша нормальна форма**

Перша нормальна форма (1НФ, 1NF) утворює ґрунт для структурованої схеми бази даних:

* Кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис.
* Уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів в різних записах), правильно визначаючи неключові атрибути.
* Атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень.

Приклад.

За визначенням, сутність, яка не має жодних повторюваних стовпців або груп даних, може бути названа Першою звичайною формою. У першій звичайній формі кожен стовпець унікальний.

**Нижче наведено, як виглядала б наша таблиця співробітників та підрозділів, якби у першій нормальній формі (1NF):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **empNum** | **прізвище** | **ім'я** | **deptName** | **deptCity** | **deptCountry** |
| 1001 | Ендрюс | Джек | Рахунки | Нью-Йорк | Сполучені Штати |
| 1002 | Шватц | Майк | Технологія | Нью-Йорк | Сполучені Штати |
| 1009 | Кубок | Гаррі | HR | Берлін | Німеччина |
| 1007 | Гарві | Паркер | Адміністратор | Лондон | Об'єднане Королівство |
| 1007 | Гарві | Паркер | HR | Лондон | Об'єднане Королівство |

**Друга нормальна форма**

Друга нормальна форма (2НФ, 2NF) вимагає, аби дані, що зберігаються в таблицях із композитним ключем, не залежали лише від частини ключа:

* Схема бази даних повинна відповідати вимогам першої нормальної форми.
* Дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці.

Приклад.

За визначенням, сутність, яка дорівнює 1NF, та один з її атрибутів визначається як первинний ключ, а решта атрибутів залежать від первинного ключа.

**Нижче наведено приклад того, як виглядатиме таблиця співробітників та підрозділів:**

**Таблиця співробітників:**

|  |  |  |
| --- | --- | --- |
| **empNum** | **прізвище** | **ім'я** |
| 1001 | Ендрюс | Джек |
| 1002 | Шватц | Майк |
| 1009 | Кубок | Гаррі |
| 1007 | Гарві | Паркер |
| 1007 | Гарві | Паркер |

**Таблиця відділів:**

|  |  |  |  |
| --- | --- | --- | --- |
| **deptNum** | **deptName** | **deptCity** | **deptCountry** |
| один | Рахунки | Нью-Йорк | Сполучені Штати |
| два | Технологія | Нью-Йорк | Сполучені Штати |
| 3 | HR | Берлін | Німеччина |
| 4 | Адміністратор | Лондон | Об'єднане Королівство |

**Таблиця EmpDept:**

|  |  |  |
| --- | --- | --- |
| **empDeptID** | **empNum** | **deptNum** |
| один | 1001 | один |
| два | 1002 | два |
| 3 | 1009 | 3 |
| 4 | 1007 | 4 |
| 5 | 1007 | 3 |

Тут ми можемо помітити, що ми розділили таблицю у формі 1NF на три різні таблиці. таблиця 'Співробітники' - це сутність, присвячена всім працівникам компанії, а її атрибути описують властивості кожного працівника. Первинним ключем для цієї таблиці є empNum.

Аналогічно, таблиця Департаментів - це сутність усіх відділів компанії, а її атрибути описують властивості кожного відділу. Первинним ключем для цієї таблиці є deptNum.

У третій таблиці ми поєднали первинні ключі обох таблиць. Первинні ключі таблиць службовців та підрозділів у цій третій таблиці називаються зовнішніми ключами.

Якщо користувач хоче вихід, подібний до того, який ми мали в 1NF, тоді користувач повинен об'єднати всі три таблиці, використовуючи первинні ключі.

**Зразок запиту буде виглядати, як показано нижче:**

SELECT empNum, lastName, firstName, deptNum, deptName, deptCity, deptCountry FROM Employees A, Departments B, EmpDept C WHERE A.empNum = C.empNum AND B.deptNum = C.deptNum WITH UR;

**Третя нормальна форма**

Третя нормальна форма (3НФ, 3NF) вимагає, аби дані в таблиці залежали винятково від основного ключа:

* [Схема бази даних](https://wiki.cuspu.edu.ua/index.php?title=%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%B1%D0%B0%D0%B7%D0%B8_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85&action=edit&redlink=1) повинна відповідати всім вимогам другої нормальної форми.
* Будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю.

Уточнення щодо ЗНФ

* Рівень нормалізації змінної відношення визначається семантикою, а не конкретним значенням цієї змінної в певний момент часу. Інакше кажучи, за конкретним значенням деякої змінної відносини неможливо визначити, чи перебуває вона, наприклад, у ЗНФ.
* Для цього необхідно також знати, які функціональні залежності визначені в розглянутій змінній відношенні. Навіть знаючи про залежності в деякій змінній відношенні, не можна на підставі конкретного її значення довести, що вона перебуває в ЗНФ.

Приклад

За визначенням, таблиця вважається третьою нормальною формою, якщо таблиця / сутність вже у другій нормальній формі, а стовпці таблиці / сутності неперехідно залежать від первинного ключа.

Давайте зрозуміємо неперехідну залежність за допомогою наступного прикладу.

**Скажімо таблицю з іменем, Клієнт має наведені нижче стовпці:**

**CustomerID** - Первинний ключ, що ідентифікує унікального клієнта   
Customer**ZIP** - Поштовий індекс населеного пункту, в якому проживає клієнт   
**CustomerCity** - місто, в якому проживає клієнт

У наведеному вище випадку стовпець CustomerCity залежить від стовпця CustomerZIP, а стовпець CustomerZIP залежить від CustomerID.

Наведений вище сценарій називається транзитивною залежністю стовпця CustomerCity від CustomerID, тобто первинним ключем. Після розуміння транзитивної залежності, тепер давайте обговоримо проблему з цією залежністю.

Можливий сценарій, коли до таблиці вноситься небажане оновлення CustomerZIP до поштового індексу іншого міста без оновлення CustomerCity, тим самим залишаючи базу даних у несумісному стані.

Для того, щоб вирішити цю проблему, нам потрібно видалити транзитивну залежність, яку можна було зробити, створивши іншу таблицю, скажімо, таблицю CustZIP, яка містить два стовпці, тобто CustomerZIP (як первинний ключ) та CustomerCity.

**Нормальна форма Бойса — Кодда (**НФБК / BCNF)

Відношення знаходиться в НФБК тоді і лише тоді, коли детермінант кожної функціональної залежності є потенційним ключем. Детермінантою є один атрибут або група атрибутів, від якої повністю функціонально залежить інший атрибут. Якщо це правило не виконується, то, щоб привести вказане відношення до НФБК, його слід розділити на два відношення шляхом двох операцій проекції на кожну функціональну залежність, детермінант якої не є потенційним ключем:

1. Проекція без атрибутів залежної частини такої функціональної залежності;
2. Проекція на всі атрибути цієї функціональної залежності.

Нормальна форма Бойса-Кодда - це **нормальна форма, яка є модифікацією третій нормальній формі, і при цьому відсутні функціональні залежності атрибутів (полів) первинного ключа від не-ключових атрибутів**.

Визначення НФБК не потребує жодних умов попередніх нормальних форм. Якщо проводити нормалізацію послідовно, то в переважній більшості випадків при досягненні 3НФ автоматично будуть задовольнятися вимоги НФБК. 3НФ не збігається з НФБК лише тоді, коли одночасно виконуються такі 3 умови:

1. Відношення має 2 або більше потенційних ключів.
2. Ці потенційні ключі складені (містять більш ніж один атрибут)
3. Ці потенційні ключі перекриваються, тобто мають щонайменше один спільний атрибут.

Приклад 1.

* Дано змінну відношення EXAM з атрибутами S (студент), J (предмет) і Р (позиція).
* Кожний кортеж (s, j, р) змінної відношення EXAM відображає відомості про те, що деякий студент s екзаменується по певному предмету j і займає певну позицію р в екзаменаційній відомості.
* Крім того, ніякі два студенти не можуть займати ту саму позицію в екзаменаційній відомості, яка відноситься до того самого предмету.

Приклад 2.

За визначенням, таблиця вважається нормальною формою Бойса-Кодда, якщо вона вже є у Третій нормальній формі, і для кожної функціональної залежності між A та B, A повинна бути суперключем.

Це визначення звучить дещо складно. ***Спробуймо розбити його, щоб краще зрозуміти це.***

* **Функціональна залежність:** Атрибути або стовпці таблиці називаються функціонально залежними, коли атрибут або стовпець таблиці однозначно ідентифікує інший атрибут (и) або стовпець (и) тієї ж таблиці.   
  **Наприклад,** стовпець empNum або Номер працівника однозначно ідентифікує інші стовпці, такі як Ім'я працівника, Заробітна плата працівника тощо в таблиці Співробітник.
* **Супер ключ:** Один ключ або група декількох ключів, які можуть однозначно ідентифікувати один рядок у таблиці, можна назвати Super Key. Загалом, ми знаємо такі ключі, як складові/складені ключі.

Давайте розглянемо наступний сценарій, щоб зрозуміти, коли виникає проблема з третьою нормальною формою, і як приходить на допомогу звичайна форма Бойса-Кодда.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **empNum** | **ім'я** | **Місто** | **deptName** | **deptHead** |
| 1001 | Джек | Нью-Йорк | Рахунки | Реймонд |
| 1001 | Джек | Нью-Йорк | Технологія | Дональд |
| 1002 | Гаррі | Берлін | Рахунки | Самара |
| 1007 | Паркер | Лондон | HR | Елізабет |
| 1007 | Паркер | Лондон | Інфраструктура | Том |

У наведеному вище прикладі працівники з empNum 1001 та 1007 працюють у двох різних відділах. Кожне відділення має завідувача. Для кожного відділу може бути кілька керівників відділів. Як і для бухгалтерії, Реймонд і Самара є двома керівниками відділів.

У цьому випадку empNum та deptName є суперключами, що означає, що deptName є основним атрибутом. На основі цих двох стовпців ми можемо ідентифікувати кожен рядок однозначно.

Крім того, deptName залежить від deptHead, що означає, що deptHead є непростим атрибутом. Цей критерій дискваліфікує таблицю як частину BCNF.

Для вирішення цього питання ми розіб‘ємо таблицю на три різні таблиці, як зазначено нижче:

**Таблиця співробітників:**

|  |  |  |  |
| --- | --- | --- | --- |
| **empNum** | **ім'я** | **Місто** | **deptNum** |
| 1001 | Джек | Нью-Йорк | D1 |
| 1001 | Джек | Нью-Йорк | D2 |
| 1002 | Гаррі | Берлін | D1 |
| 1007 | Паркер | Лондон | D3 |
| 1007 | Паркер | Лондон | D4 |

**Таблиця відділу:**

|  |  |  |
| --- | --- | --- |
| **deptNum** | **deptName** | **deptHead** |
| D1 | Рахунки | Реймонд |
| D2 | Технологія | Дональд |
| D1 | Рахунки | Самара |
| D3 | HR | Елізабет |
| D4 | Інфраструктура | Том |

**Четверта нормальна форма**

Четверта нормальна форма (4НФ, 4NF) потребує, аби в схемі баз даних не було нетривіальних багатозначних залежностей множин атрибутів від будь чого, окрім надмножини ключа-кандидата. Вважається, що таблиця знаходиться у 4НФ тоді і лише тоді, коли вона знаходиться в НФБК та багатозначні залежності є функціональними залежностями. Четверта нормальна форма усуває небажані структури даних — багатозначні залежності.

Інше визначення; таблиця знаходиться у четвертій нормальній формі, якщо вона не містить двох або більше незалежних даних, що описують відповідну сутність.

**П'ята нормальна форма**

П'ята нормальна форма (5НФ, 5NF, PJ/NF) вимагає, аби не було нетривіальних залежностей об'єднання, котрі б не витікали із обмежень ключів. Вважається, що таблиця в п'ятій нормальній формі тоді і лише тоді, коли вона знаходиться в 4НФ та кожна залежність об'єднання зумовлена її ключами-кандидатами. Тобто може бути розбита на кілька таблиць без втрати будь-яких даних.

**Нормальна форма домен/ключ**

Ця нормальна форма вимагає, аби в схемі не було інших обмежень окрім ключів та доменів.

**Шоста нормальна форма**

Таблиця знаходиться у 6NF, якщо вона знаходиться у 5NF та задовольняє вимозі відсутності нетривіальних залежностей. Зазвичай 6NF ототожнюють з DKNF.

Кожна нормальна форма обмежує певний тип функціональної залежності (ФЗ) і усуває аномалії обробки даних.

***Властивості нормальних форм:***

1НФ - всі атрибути відносини прості;

2НФ - відношення знаходиться в 1НФ і не містить часткових ФЗ;

3НФ - відношення знаходиться у 2НФ і не містить транзитивних ФЗ від ключа;

НФБК - відношення знаходиться в 3НФ і не містить ФЗ ключів від неключових атрибутів;

4НФ, застосовується при наявності більш ніж однієї багатозначної ФЗ - відношення знаходиться в НФБК або 3НФ і не містить незалежних багатозначних ФЗ;

5НФ - відношення знаходиться в 4НФ і не містить ФЗ по з'єднанню.

*Для самостійного вивчення*: Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

*Література*.

1. *Дейт К.* Введение в системы баз данных : пер. с англ. / К. Дейт. – 8-е изд. – М., СПб.: Вильямс, 2005. – 1328 с.

2. *Мейер Д.* Теория реляционных баз данных / Д. Мейер. – М., Мир, 1987. – 608 с.

3. *Кузнецов С.Д.* Основы баз данных / С.Д. Кузнецов. – 2-е изд. – Москва: Бином, 2007. – 251 с.

*4.* Гайдаржи В.І., Ізварін І.В. Бази даних в інформаційних системах. – К.; Університет «Україна», 2018. – 418 с.

*Запитання для самоперевірки.*

1. Дайте визначення реляційної моделі.
2. Що описує реляційна модель?
3. Які були цілі створення реляційної моделі даних?
4. Визначте вимоги до таблиць реляційної моделі.
5. Які передбачаються обов'язкові об'єкти в структурі даних реляційної моделі даних?
6. Які відносини між об’єктами в реляційної моделі даних ви можете визначити?
7. Визначте переваги і недоліки реляційної моделі.
8. В чому полягають відмінності реляційної моделі від інших моделей?
9. Які види ключів ви знаєте і в чому їх особливості?
10. В чому полягає процес нормалізації схеми бази даних?
11. Що таке нормальна форма? Які нормальні форми ви знаєте?