Лекція 16

Основи проєктування БД

Класифікація систем управління базами даних

Системи управління базами даних можуть бути класифіковані на основі декількох критеріїв, таких як:

1 модель даних,
2 кількість користувачів,
3 розподіл баз даних.

1 Класифікація на основі моделі даних

Найпопулярнішою моделлю даних, що використовується сьогодні, є реляційна модель даних. Відомі DBMSS, такі як Oracle, MS SQL Server, DB2 і MySQL підтримують цю модель. Інші традиційні моделі, такі як ієрархічні моделі даних і мережеві моделі даних, все ще використовуються в промисловості в основному на платформах великих ЕОМ. Однак вони не є загальновживаними через їх складність. Всі ці моделі називаються традиційними, тому що вони передували реляційній моделі.

В останні роки були введені новіші об'єктно-орієнтовані моделі даних. Такі моделі є системами управління базами даних, в яких інформація представлена у вигляді об'єктів, так само як в об'єктно-орієнтованому програмуванні. Об'єктно-орієнтовані бази даних відрізняються від реляційних, які орієнтовані на таблиці. Об'єктно-орієнтовані системи управління базами даних (Object-oriented database management systems (OODBMS)) поєднують можливості баз даних з можливостями об'єктно-орієнтованої мови програмування.

Об'єктно-орієнтовані моделі, як і очікувалося, не мають широкого використання. Деякі приклади об'єктно-орієнтованих СУБД – O2, ObjectStore і Jasmine.

2 Класифікація на основі кількості користувачів

СУБД можуть бути класифіковані за кількістю користувачів, одночасну роботу яких вона підтримує. Це може бути однокористувацька система баз даних, яка підтримує роботу одного користувача в певний момент часу, або мультикористувацька система баз даних, яка підтримує роботу декількох користувачів одночасно.

3 Класифікація на основі розподілу баз даних

Існує чотири основні системи розподілу для систем баз даних, які можуть бути використані для класифікації СУБД.

Централізовані системи

За використання централізованої системи база даних і СУБД зберігаються на одному сайті, який використовується також кількома іншими системами. Централізована система неведена на рисунку 6.1.

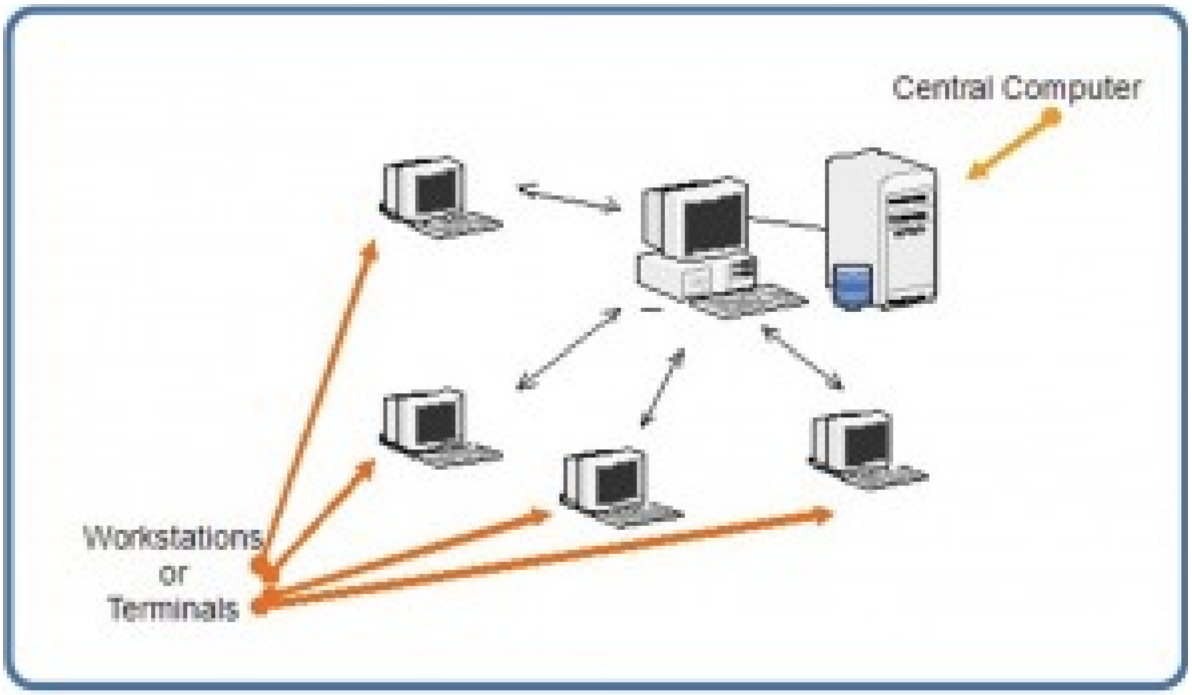

Рисунок 6.1. Приклад централізованої системи бази даних

На початку 1980-х років багато канадських бібліотек використовували GEAC 8000 для перетворення своїх каталогів паперових карток на системи автоматизованих централізованих каталогів. Кожен книжковий каталог мав поле зі штрих-кодом, аналогічним тому, який є на продуктах супермаркету.

Розподілена система баз даних

У розподіленій системі баз даних фактична база даних і програмне забезпечення СУБД розподілені за різних сайтів, які поєднані через комп'ютерну мережу, як показано на рисунку 6.2.

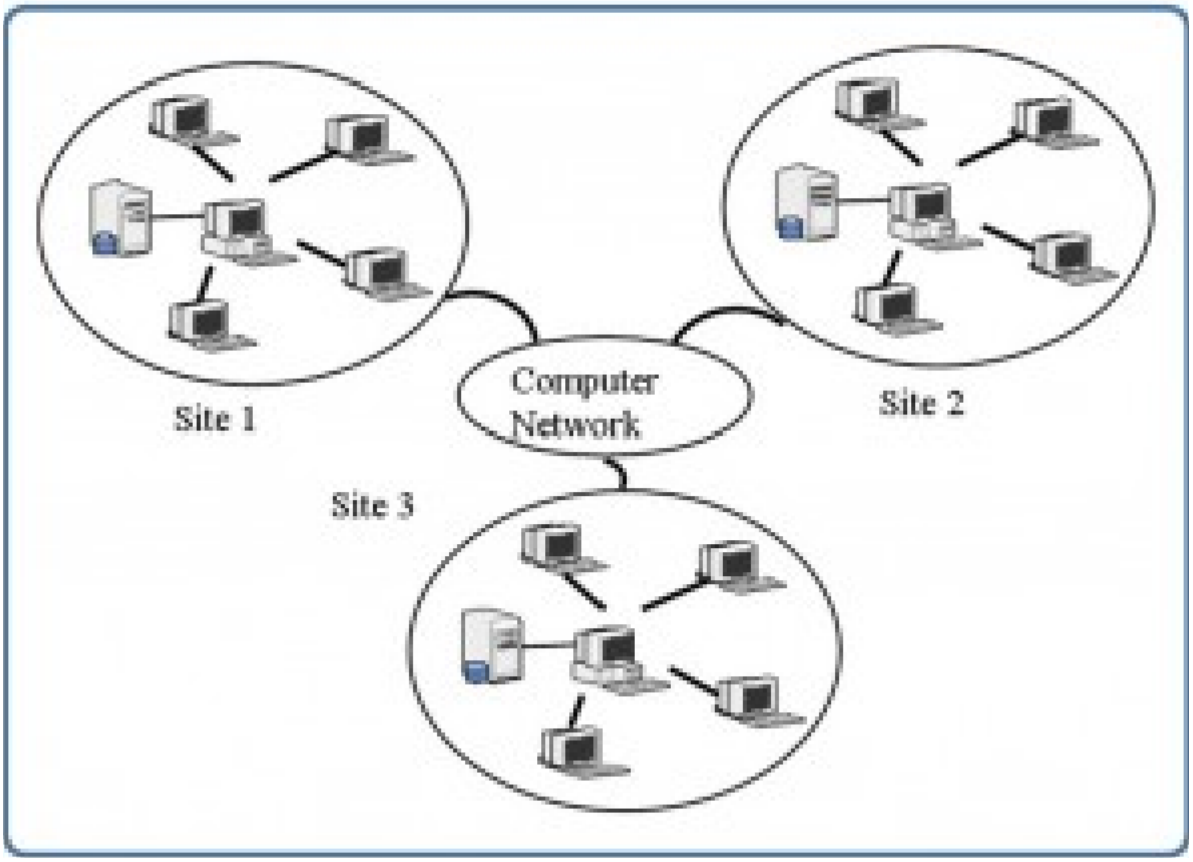

Рисунок 6.2. Приклад розподіленої системи баз даних

Однорідні розподілені системи баз даних

Однорідні розподілені системи баз даних використовують одне і те ж програмне забезпечення СУБД з декількох сайтів. Обмін даними між цими різними сайтами може бути легко реалізованим. Наприклад, бібліотечні інформаційні системи одного й того ж самого постачальника, такі, наприклад, як Geac Computer Corporation, використовують те ж саме програмне забезпечення СУБД, що дозволяє легко обмінюватися даними між різними бібліотечними сайтами Geac.

Різнорідні розподілені системи баз даних

У різнорідній розподіленій системі баз даних, різні сайти можуть використовувати різне програмне забезпечення СУБД, але існує додаткове спільне програмне забезпечення для підтримки обміну даними між цими сайтами. Наприклад, різні системи баз даних бібліотеки використовують той самий формат автоматизованої каталогізації, для підтримки обміну даними з бібліотеками.

Реляційна модель даних

Реляційна модель даних була представлена Едгаром Франком Коддом в 1970 році. В даний час є найбільш широко використовуваною моделлю даних.

Реляційна модель забезпечує основу для:

• Дослідження з теорії даних/зв'язків/обмежень

• Численних методологій проектування баз даних

• Стандартної мови доступу до бази даних під назвою структурована мова запитів (Structured Query Language)

• Майже всіх сучасних комерційних систем управління базами даних

Реляційна модель даних описує світ як «сукупність взаємозв'язків (або таблиць)».

Основні поняття реляційної моделі даних:

Відношення

Відношення – фундаментальне поняття реляційної моделі даних. З цієї причини модель і називається *реляційною* (від англійського *relation* — відношення).

N-арним відношенням R, або відношенням R ступеня n, називають підмножину декартового добутку множин, не обов'язково різних. Вихідні

множини називають в моделі *доменами* (в СУБД використовується поняття *тип даних*).

Відношення має просту графічну інтерпретацію, воно може бути представлено у вигляді таблиці, стовпці (поля, атрибути) якої відповідають входженням доменів у відношення, а рядки (записи, кортежі) – наборам з *n* значень, які взяті з початкових доменів. Кількість рядків, називають *кардинальним числом відношення* або потужністю відношення.

Така таблиця має ряд властивостей:

1. В таблиці немає двох однакових рядків.
2. Таблиця має стовпці, відповідні атрибутам відношення.
3. Кожний атрибут у відношенні має унікальне ім'я.
4. Порядок рядків у таблиці довільний.

Під *атрибутом* розуміють входження домену у відношення. Рядки відношення називаються *кортежами*.

- *Заголовок (схема)* відношення *r* (*Hr*) – скінченна множина впорядкованих пар виду <*A, T*>, де *A* називається іменем атрибута, а *T* означає ім'я деякого базового типу або раніше визначеного домену. Всі імена атрибутів в заголовку мають бути різними.

- *Кортеж tr*, відповідний заголовку *Hr* – множина впорядкованих триплетів (трійок) <*A, T, v*>, по одному такому триплету для кожного атрибута в *Hr*. Третій елемент *v* триплета має бути дозволеним значенням типу даних або домену *T*.

- *Тіло Br* відношення – невпорядкована множина різних кортежів *tr*.

- *Значенням Vr* відношення *r* називається пара множин *Hr* та *Br*.

Припустимо, вміст доменів є наступним:

- = {Бовкун, Вередун, Прядун}
- = {Фізика, Хімія}
- = {3,4,5}

Тоді повний декартів добуток складається з 18 трійок – прізвище, навчальна дисципліна, оцінка.

Тоді відношення R може моделювати реальну ситуацію і містити п'ять рядків, які відповідають результатам сесії (Вередун екзамен з фізики не здавав):

R		
Прізвище	**Предмет**	**Оцінка**
Бовкун	Фізика	4
Бовкун	Хімія	3
Вередун	Хімія	5
Прядун	Фізика	5
Прядун	Хімія	4

Таблиця

База даних складається з декількох таблиць і кожна таблиця містить дані. На рисунку 7.1 зображена база даних, яка містить три таблиці.

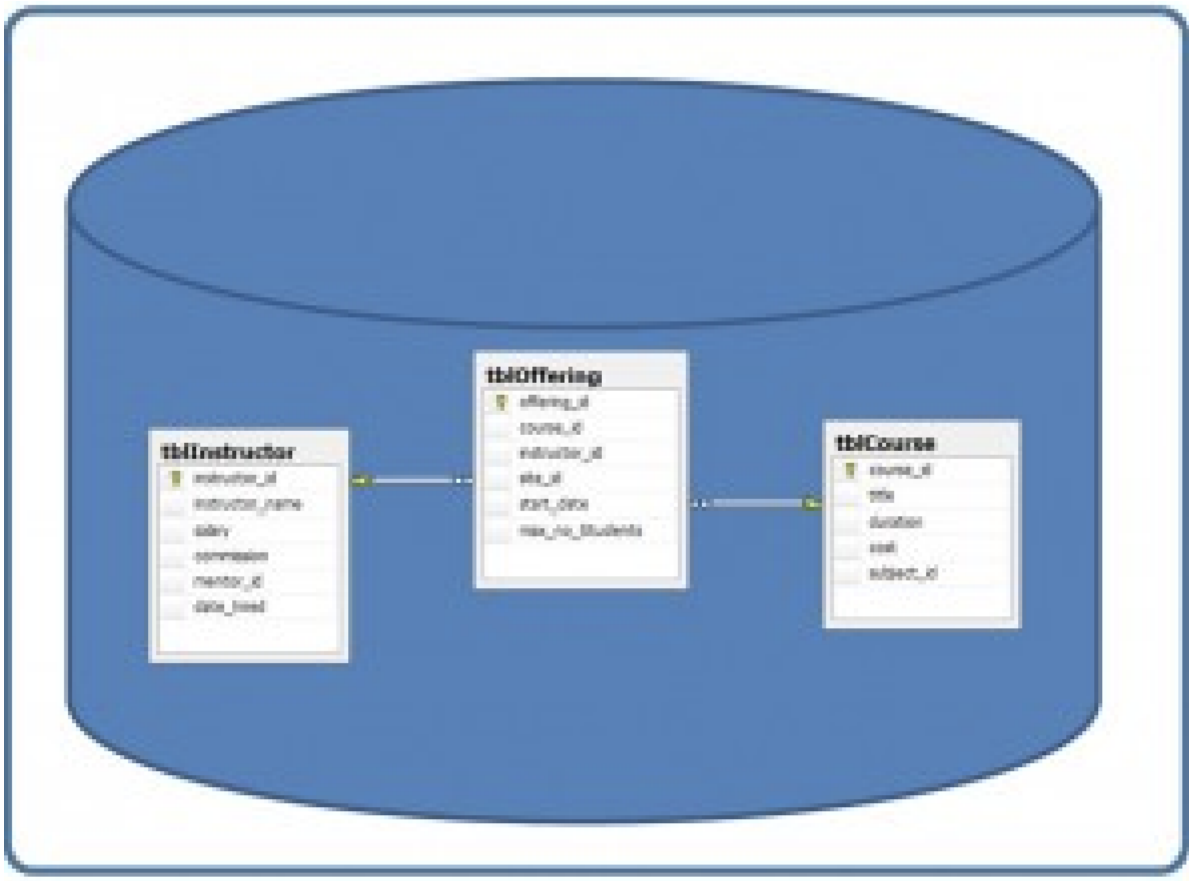

Рисунок 7.1. таблиці БД

Поля

База даних зберігає впорядковано елементи інформації або факти. Розуміння того, як використовувати базу даних у найефективніший спосіб вимагає розуміння цього способу організації зберігання цих фактів.

Основні одиниці зберігання називаються стовпчиками або полями або атрибутами. Поля є основними компонентами даних, на які можна розбити вміст предметної області. Під час вирішення, які саме поля слід створювати в БД, потрібно узагальнено обміркувати всю наявну інформацію, наприклад, виокремити спільні компоненти інформації, яку слід зберігати в базі даних з метою уникнення специфіки, що відрізняє один екземпляр об’єкта від іншого.

На рисунку 7.2 наведено приклад ID-картки, який відображає взаємозв'язок між полями та їх даними.

Field Name	Data
First Name	Isabelle
Family Name	Whelan
Nationality	British
Salary	109,900
Date of Birth	15 September 1983
Marital Status	Single
Shift	Mon, Wed
Place of issue	Addis Ababa
Valid until	17 December 2003

Рисунок 7.2. Приклад ID-картки

Домен

Домен – це початкові набори атомарних (неподільних, елементарних) значень, які використовуються для моделювання даних. Під атомарними значеннями розуміють, значення в домені, які є неподільним щодо реляційної моделі. Наприклад:

• Домен Marital Status має набір можливостей: Margen, Single, Divorced.

• Домен Shift має набір всіх можливих днів: {Mon, Tue, Wed…}.

• Доменом Salary є множина всіх чисел з плаваючою комою більших за 0 і менше 200,000.

• Домен First Name – набір рядків символів, що виражає імена людей.

Таким чином, домен – це набір допустимих значень, які може містити певний стовпчик (певне поле). Це базується на різних властивостях і типі даних поля.

Записи

Так само, як вміст будь-якого одного документа або елемента може бути розбитий на певні складові частини, дані, які зберігаються в полях може бути розподілений по рядках. Проте зв'язок між цими даними повинен бути прийнятний для відтворення з них цілісної картини. Записи задовольняють цим вимогам. Записи (або кортежи) містять поля, які мають відношення, наприклад, клієнт або працівник.

Записи і поля складають основу всіх баз даних. Проста таблиця (рис. 7.3) дає нам найчіткіше зображення того, як записи і поля працюють разом у процесі зберігання даних.

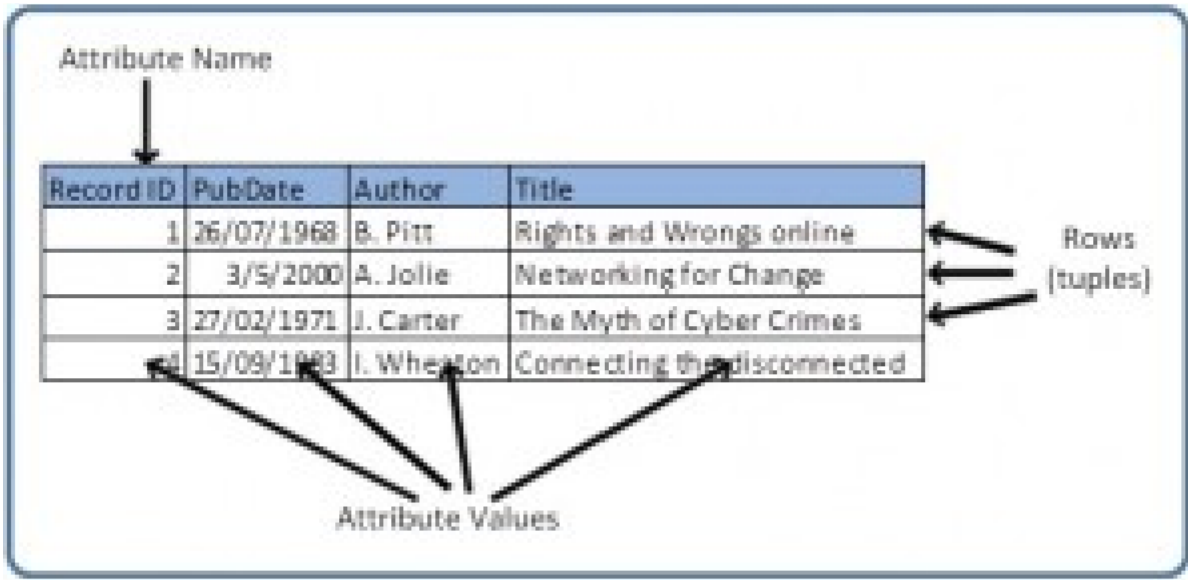

Рисунок 7.3. Приклад таблиці

Простий приклад таблиці на рисунку 7.3 демонструє, що поля можуть містити діапазони різних типів даних:

• Поле ідентифікатора запису: це порядковий номер, тому тип даних – ціле число.

• Поле дати публікації: відображає день/місяць/рік публікації книги, тому тип даних – дата.

• Поле прізвища автора: містить лише текстову інформацію, тому тип даних – текст.

• Поле заголовоку публікації: може містити довільний текст, тип даних – текст.

Завдяки визначенню типів даних записи бази даних можна фільтрувати та сортувати за значеннями певних полів. Наприклад, можна створювати запити для вибірки записів, дата публікації яких: 1 раніше заданої дати, 2 дорівнює заданії даті або 3 знаходиться між двома вказаними датами. Аналогічно, можна відсортувати записи за значеннями поля дата. Таке можливо через те, що завдяки привласненню типу даних дата база даних сприймає інформацію в полі Дата не просто як числа або набір символів розподілених косою рискою а, як дати, які повинні бути впорядковані за календарною системою.

Ступінь таблиці

Ступінь – це кількість атрибутів у таблиці. У нашому прикладі на рисунку 7.3, ступінь дорівнює 4.

Властивості таблиці

• Таблиця має назву, відмінну від усіх інших таблиць у базі даних.

• Немає дублікатів рядків: набір значень атрибутів для кожного рядка є унікальним.

• Дані в колонках є атомарними: таблиця не повинна містити повторюваних груп значень або багатозначних атрибутів (тобто не можливо щоб поле певного рядка містило більше одного значення або значення різних типів).

• Дані певного поля походять з одного домену, який визначає їх тип даних, включаючи:

число (numeric, integer, float, smallint,…)

символ (string)

логічна дата (true або false)

• Операції обробки даних, що поєднують різні типи даних, не допускаються.

• Кожен атрибут має унікальну назву.

• Послідовність стовпчиків та рядків не важливі.

Модель даних взаємозв'язків сутностей (Entity Relationship модель)

Модель даних взаємозв'язків сутностей (ER-модель або діаграма) існує вже понад 40 років. Така модель добре підходить для моделювання даних для використання базами даних, тому що забезпечує досить абстрактне представлення даних, є легкою для обговорення та розуміння. Моделі ER легко перетворюються на відносини для складання таблиць. Моделі ER, або ER схеми, представляються у вигляді ER діагарам.

ER-моделювання базується на двох концепціях:

• Сутності, визначають як таблиці, які містять конкретну інформацію (дані) про об’єкти реального світу.

• Взаємозв'язки, визначені як асоціації або взаємодії між сутностями відбивають зв’язки між об’єктами в предметній області.

Ось приклад того, як ці два поняття можуть бути об'єднані в моделі даних ER: ДОЦЕНТ (сутність) ВИКЛАДАЄ (взаємозв'язок) КУРС СИСТЕМИ БАЗ ДАНИХ (сутність).

Для ілюстрації концепції моделі ER використаємо базу даних БАЗА ДАНИХ КОМПАНІЇ, яка містить інформацію про співробітників, відділи та проекти. До важливих характеристик, які слід відзначити з метою створення моделі ER, належать:

• У компанії є кілька відділів. Кожен відділ має унікальну ідентифікацію, ім'я, місце розташування офісу та конкретного працівника, який керує відділом.

• Відділ контролює ряд проектів, кожен з яких має унікальну назву, унікальний номер і бюджет.

• Кожен працівник має прізвище, ідентифікаційний номер, адресу, заробітну плату та дату народження. Співробітник призначається в один відділ, але може приєднатися до декількох проектів. Потрібно зберігати дату початку роботи працівника в кожному проекті, також безпосереднього керівника кожного співробітника.

• Слід також зберігати дані про тих, хто знаходить під опікою кожного працівника (неповнолітні діти, непрацездатні особи тощо). Слід зберігати ім'я, дату народження та відносини з працівником, для кожної з таких осіб

Сутність, набір сутностей і тип сутності

Сутність – це об'єкт в реальному світі з незалежним існуванням, який в даній предметній області є відмінним від інших об'єктів. Сутністю може бути:

• Об'єкт з фізичним існуванням – реальний об’єкт (наприклад, лектор, студент, автомобіль)

• Об'єкт з концептуальним існуванням – абстрактний об’єкт (наприклад, курс, робота, позиція)

Сутності можуть бути класифіковані виходячи з їх міцності. Сутність вважається слабкою, якщо її таблиці існують лише за умови існування таблиць інших сутностей:

• така сутність не може існувати без зв'язку з іншим об'єктом;

• первинний ключ такої сутності є похідним від первинного ключа батьківської сутності, від існування якої вона залежить.

Таблиця осіб для опіки ОСОБИ ЗАЛЕЖНІ, у базі даних КОМПАНІЯ, є слабкою сутністю, оскільки її основний ключ залежить від таблиці СПІВРОБІТНИК. Без відповідного запису про співробітника не буде й відповідних записів щодо осіб, які знаходяться під опікою цього співробітника.

Сутність вважається сильною, якщо вона може існувати без жодної з пов'язаних з нею сутностей.

Таблиця без зовнішного ключа або таблиця, що містить зовнішній ключ, який може містити значення null – сильна сутність.

Зовнішній ключ – це одне або декілька полів (атрибутів), які є первинними в іншій таблиці і значення яких замінюється значеннями первинного ключа іншої таблиці. Наприклад: нехай між таблицями “Працівник” і “Зарплата” є взаємозв’язок за полем “Табельний номер”. У цьому випадку поле “Табельний номер” таблиці “Працівник” може бути первинним ключем, а поле “Табельний номер” таблиці “Зарплата” зовнішнім ключем. Це означає, що значення поля “Табельний номер” таблиці “Зарплата” замінюються значеннями поля “Табельний номер” таблиці “Працівник”.

Тип сутності визначає сукупність подібних сутностей.

Набір сутностей – сукупність сутностей певного типу в певний момент часу. На діаграмі взаємозв'язку сутностей (ERD) тип сутності представляється назвою поля. Наприклад, на рисунку 8.1 тип сутності є СПІВРОБІТНИК.

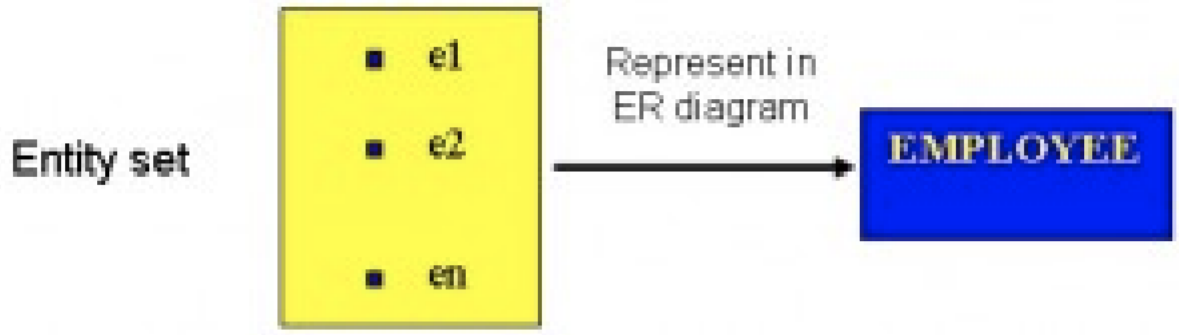

Рисунок 8.1. ERD для сутності СПІВРОБІТНИК

Залежність від існування об’єкта

Існування сутності залежить від існування пов'язаного з нею об'єкта реального світу. Сутність є залежною від існування іншого об’єкта, якщо вона має обов'язковий зовнішній ключ (тобто, атрибут зовнішнього ключа, який не може мати значення null). Наприклад, у БАЗІ

даних КОМПАНІЯ, ОСОБИ ЗАЛЕЖНІ є сутністю залежною від існування сутності СПІВРОБІТНИК.

Види сутностей

Існують різні типи сутностей: незалежні сутності, залежні сутності та характерні сутності.

Незалежні сутності, так звані ядра (kernel), є основою бази даних. На них базуються інші таблиці. Ядра мають такі характеристики:

• Вони є будівельними блоками бази даних.

• Первинний ключ ядра може бути простим або складеним.

• Первинний ключ ядра не може бути зовнішнім ключем.

• існування ядра не залежить від існування жодної іншої сутності.

Прикладами незалежної сутності є таблиця КЛІЄНТ, таблиця РОБОТОДАВЕЦЬ або таблиця ПРОДУКТ.

Залежні сутності або похідні сутності – сутності значення яких залежать від інших сутностей (даних таблиць). Залежні сутності мають такі характеристики:

• Залежні сутності використовуються для з'єднання двох ядер разом.

• Таблиці залежних сутностей існують залежно від існування двох або більше таблиць інших сутностей.

• За існування залежних сутностей відношення багато до багатьох утворюють асоціативні таблиці з принаймні двома зовнішніми ключами.

• Залежні сутності можуть містити інші атрибути.

• Кожен зовнішній ключ залежної сутності ідентифікує іншу пов'язану таблицю.

• Залежні сутності можуть мати три варіанти первинного ключа:

1. За використання композиції з кількох зовнішніх ключів відповідних таблиць, якщо значення такого складеного ключа є унікальним.

2. За використання композиції з одного або кількох зовнішніх ключів та ключового поля цієї сутності.

3. За створення нового простого первинного ключа.

У базі даних КОМПАНІЯ ключами сутностей можуть бути:

* СПІВРОБІТНИК (EID, Name, Address, Age, Salary) – EID (ідентифікаційний номер) є простим первинним ключем.

* КОНТАКТИ СПІВРОБІКНИКІВ(EID, Phone) – EID є частиною складеного первинного ключа. Тут, EID також є зовнішнім ключем для цієї сутності.

Атрибути

Кожен об’єкт реального світу має властивості, важливі з точки зору даної предметної області. Відповідно до цього кожна сутність описується набором атрибутів (наприклад, СПІВРОБІТНИК = (ім'я, адреса, дата народження (вік), зарплата).

Кожен атрибут має ім'я, і пов'язаний з суб'єктом і доменом. Однак інформація про домен атрибутів не представлена на ERD.

Приклад атрибутів діаграми взаємозв'язку сутностей наведено на рисунку 8.2.

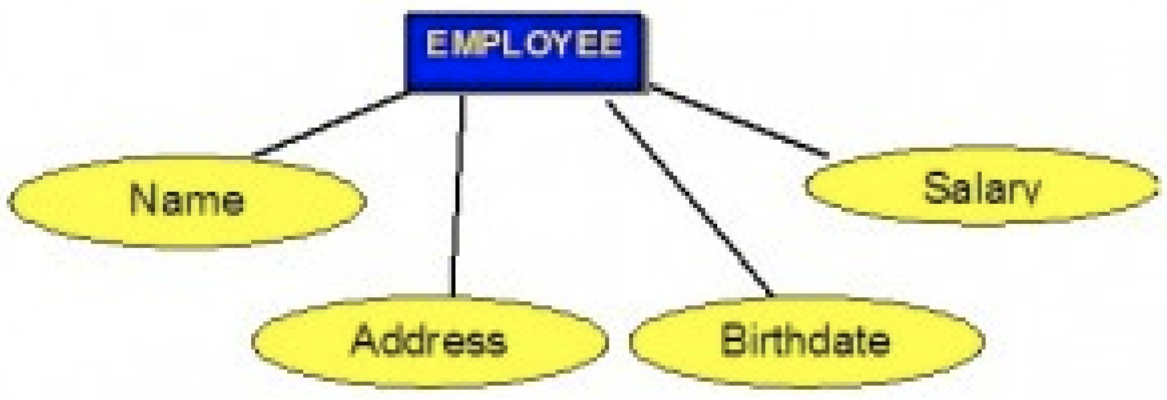

Рисунок 8.2. Приклад атрибутів діаграми взаємозв'язку сутностей

Типи атрибутів

Існує декілька типів атрибутів.

Прості атрибути – це ті, що виокремлені з доменів атомарних значень; їх також називають однозначними атрибутами. У базі даних КОМПАНІЯ прикладом такого атрибута є: Name = {John} ; Age = {23}

Складені атрибути – це ті, що складаються з ієрархії кількох атрибутів. Використовуючи приклад бази даних, показаний на рисунку 8.3, Address може складатися з номеру будинку, вулиці та району. Тобто атрибут АДРЕСА може мати складене значення → Address = {59 + 'Meek Street' + 'Kingsford'}

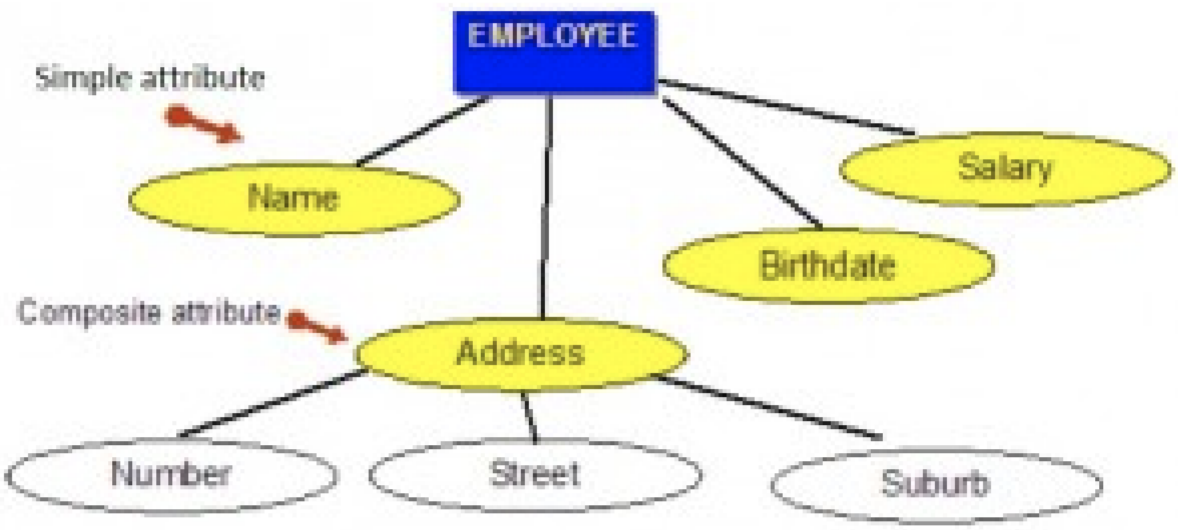

Рисунок 8.3. Приклади простих та складених атрибутів

Багатозначні атрибути – атрибути, що мають обмежений набір значень для кожного елемента сутності. Прикладом багатозначного атрибута у базі даних КОМПАНІЯ є ступені працівника: BSc, MIT, PhD (рис.8.4).

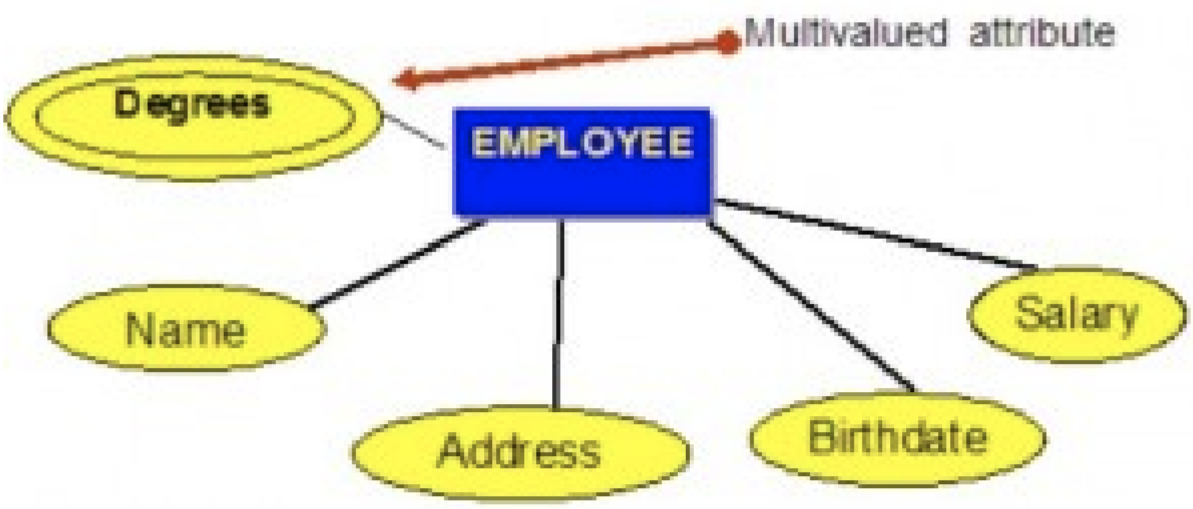

Рисунок 8.4. Приклад багатозначного атрибута

Похідні атрибути – атрибути, що містять значення, розраховані зі значень інших атрибутів. Прикладом такого атрибута є ВІК, який можна отримати з атрибута ДАТА НАРОДЖЕННЯ (рис. 8.5). У цій ситуації ДАТА НАРОДЖЕННЯ є атрибутом, який фізично зберігається у базі даних.

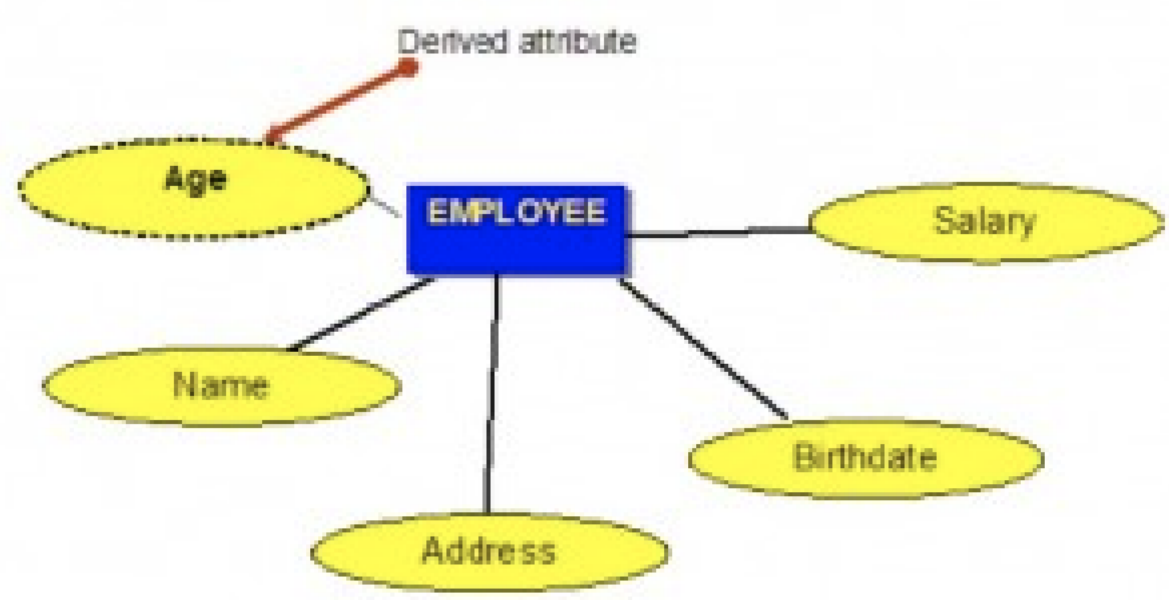

Рисунок 8.5. Приклад похідного атрибута

Ключі

https://elib.lntu.edu.ua/sites/default/files/elib_upload/%D0%95%D0%9D%D0%9F_%D0%A1%D0%B0%D0%B2%D0%B0%D1%80%D0%B8%D0%BD_%D0%9B%D0%B5%D0%BF%D0%BA%D0%B8%D0%B9/teoretic/lec5.html

Важливим питанням під час роботи із сутністю є визначення її ключового поля – ключа. Ключ є атрибутом або групою атрибутів, значення яких можуть бути використані для однозначної ідентифікації окремого екземпляра в наборі сутності.

Існує кілька типів ключів.

Потенційний або можливий *ключ* (Candidate key) – простий або складений ключ, який задовольняє вимогам унікальності та мінімальності. Унікальність ключа визначається тим, що жодні два рядки в таблиці не можуть мати однакове значення ключа в будь-який час. Мінімальність ключа визначається тим, що кожен атрибут, який входить до складу ключа, є необхідним для того, щоб досягти унікальності.

Для бази даних КОМПАНІЯ, якщо сутність є СПІВРОБІТНИК (EID, First Name, Last Name, SIN, Address, Phone, BirthDate, Salary, DepartmentID), можливі потенційні ключі:

• EID

• First Name та Last Name – може бути використаний лише за припущення, що в компанії більше не може бути двох осіб, які мають однакові імена та прізвища;

• Last Name та DepartmentID – може бути використаний лише за припущення, що дві людини з одним прізвищем не можуть працювати в одному відділі.

Складений ключ – ключ, який складається з двох або більше атрибутів, але він повинен бути мінімальним.

Використовуючи приклад для потенційного ключа, можливі складені ключі: First Name + Last Name та Last Name + DepartmentID.

Первинний ключ – це потенційний ключ, який вибирається розробником бази даних для використання під час ідентифікації екземплярів зі всієї множини сутностей. Він повинен однозначно ідентифікувати кортежі в таблиці і не може мати значення null. Первинний ключ вказується в моделі ER підкресленням атрибута.

Прикладі стосовно сутності СПІВРОБІТНИК первинним ключем є EID.

Вторинний ключ є атрибутом, який використовується виключно для цілей пошуку (може бути складеним), наприклад: Телефон і прізвище.

Альтернативними ключами є всі потенційні ключі, які не були визначені як первинний ключ.

Зовнішній ключ є атрибутом в таблиці, який посилається на первинний ключ в іншій таблиці, він може мати значення null. Як зовнішній, так і первинний ключі повинні бути одного типу даних.

Наприклад для бази даних КОМПАНІЯ для сутності СПІВРОБІТНИК DepartmentID є зовнішнім ключем, тоді для сутності ВІДДІЛ цей атрибут буде первинним ключем.

Значення Null

Значення null – значення, яке не залежно від типу даних певного атрибута визначає днаі, які є невідомими або неприйнятними для опрацювання. Null не означає нуль в математичному сенсі або відсутність значення. До особливостей null належать:

• Немає запису даних

• Не допускається в первинному ключі

• Слід уникати в інших атрибутах

• Може відбивати:

  – Невідоме значення атрибута

– Відоме, але відсутнє, значення атрибута
– Умову «не значення не застосовується»

• Може створювати помилки під час застосування деяких математичних функцій: COUNT(), AVERAGE() та SUM()

• Може створювати логічні помилки, під час зв’язування реляційних таблиць

Зауваження: Результат операції порівняння є null, якщо будь-який з аргументів має значення null. Результатом арифметичної операції є null, якщо будь-який з аргументів має значення null (за винятком функцій, які ігнорують значення null).

Приклад використання значення null наведено на рисунку 8.6.

Salary_tbl

emp#	jobName	salary	commission
E10	Sales	12500	32090
E11	Null	25000	8000
E12	Sales	44000	0
E13	Sales	44000	Null

Рисунок 8.6. приклад таблиці з використанням значення null

Наприклад слід знайти всіх співробітників (emp#) у таблиці Sales (за значенням поля JobName), в яких зарплата плюс комісійні є більшими за 30,000.

• SELECT emp# FROM Salary_tbl

• WHERE jobName = Sales AND

• (commission + salary) > 30,000 –> E10 and E12

Результат виконання такого запиту не включає рядок E13 через значення null в полу commission. Для того, щоб рядок з нульовим значенням був включений, потрібно продивитися окремо поля ці поля. Оскільки один з аргументів (commission) має значення null функція додавання комісійних та заробітної плати для працівника E13, буде видавати як результат значення null. Рішення такої проблеми наведено нижче.

• SELECT emp# FROM Salary_tbl

• WHERE jobName = Sales AND

• (commission > 30000 OR

• salary > 30000 OR

• (commission + salary) > 30,000 –>E10 and E12 and E13

Зв’язки

Зв'язки – це механізм, який пов’язує таблиці в одну базу даних. Вони використовуються для з'єднання пов'язаної інформації між таблицями.

Ефективність зв’язків ґрунтується на тому, як визначається первинний ключ пов'язаної сутності. *Слабкий*, або невизначений зв’язок виникає, якщо первинний ключ пов'язаної сутності не містить первинного ключового компонента батьківської сутності. Приклади баз даних компанії:

• Customer(CustID, CustName)

• Order(OrderID, CustID, Date)

Тут пов’язана сутність ЗАМОВЛЕННЯ має власний первинний ключ OrderID.

Сильний, або визначений зв’язок існує тоді, коли первинний ключ пов'язаної сутності містить первинний ключовий компонент батьківської сутності. Наприклад:

• Course(CrsCode, DeptCode, Description)

• Class(CrsCode, Section, ClassTime…)

Типи зв’язків

Нижче наведено описи різних типів відносин.

Один до багатьох (1:M)

Відношення один до багатьох (1:M) є нормою для проектування будь-якої реляційної бази даних, а тому зустрічається в усіх реляційних середовищах баз даних. Наприклад, в одному відділі працює багато співробітників, а один співробітник працює лише в одному відділі. На рисунку 8.7 показано зв'язок типу один до багатьох.

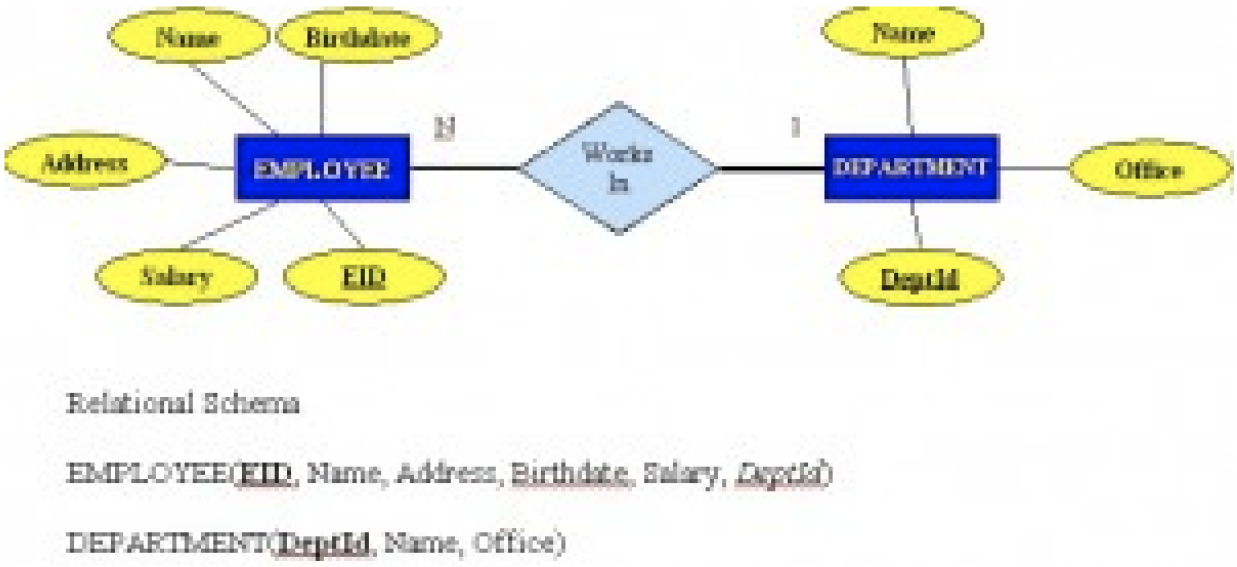

Рисунок 8.7. Приклад відношення один до багатьох

Один до одного (1:1)

Відношення один до одного (1:1) є відношенням одного екземпляра сутності лише до одного екземпляра іншої сутності, і навпаки. Відношення такого типу не є ефективними, тому рідко зустрічаються в реляційній конструкції бази даних. Насправді, це може свідчити про те, що дві сутності насправді можуть бути об’єднані в одній таблиці.

Прикладом з бази даних КОМПАНІЯ є сутності СПІВРОБІТНИК та КОНТАКТИ СПІВРОБІТНИКІВ.

Багато до багатьох (M: M)

Для відношень багато до багатьох визначають наступні моменти:

• Відношення (M: M) не може бути безпосередньо реалізовано в реляційній моделі.

• Відношення (M: M) можна змінити на дві відносини 1:M.

• Відношення (M: M) можна реалізувати, розірвавши кілька зв'язків типу 1:M.

• Відношення (M:M) передбачає створення складеної сутності.

• Відношення (M: M) утворює два або більше 1:M відносин.

• Таблиця складеної сутності повинна містити принаймні первинні ключі таблиць початкових сутностей.

• Таблиця зв'язування містить кілька збігів значень зовнішніх ключів.

• За необхідності до цієї таблиці можуть бути додані інші атрибути.

• Уникнути проблем, притаманних відношенням M:M, можна шляхом створення складеної сутності або сутності для поєднання.

Наприклад, один співробітник може працювати над багатьма проектами, а над одним проектом може працювати багато співробітників (це залежить від правил ведення бізнесу).

На рисунку 8.8 показано ще один аспект відносин M:M: один співробітник має різні дати початку для різних проектів. Тому нам потрібна ТАБЛИЦЯ ПРИЄДНАННЯ, яка містить EID, Code і StartDate.

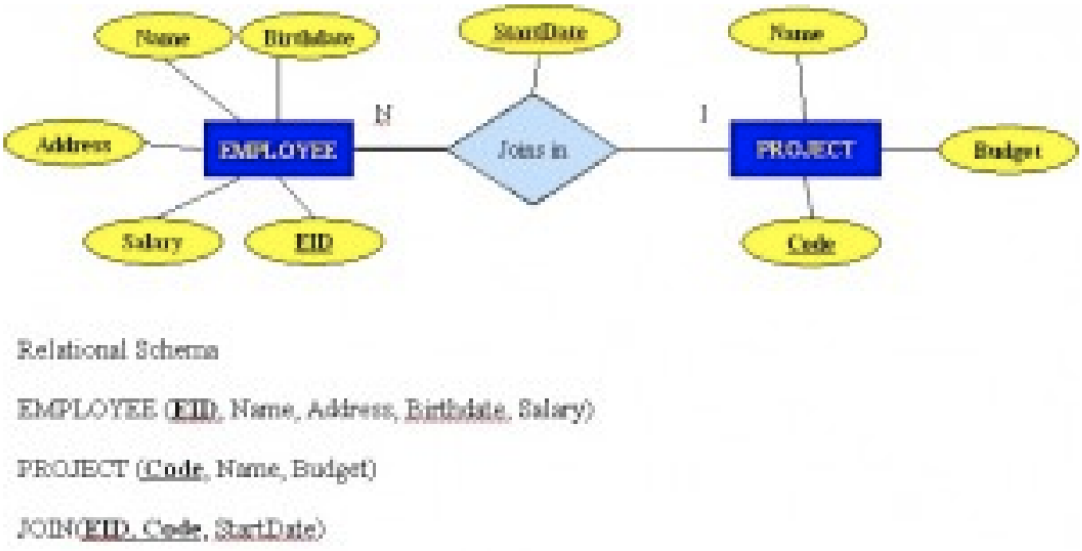

Рисунок 8.8. Приклад відношень M:M

Приклад відображення типу бінарних зв'язків M:N

• Для кожного відношення M:M бінарні відносини, ідентифікують два відношення.

• A і B представляють два типи сутностей, що беруть участь у відношенні R.

• Створити нове відношення S для представлення R.

• S має містити первинні ключі сутностей A та B, які разом можуть складати первинний ключ для таблиці відношення S.

Унарне (рекурсивне) відношення

Рекурсивний зв’язок означає, що потенційний та зовнішній ключі, із встановленою між ними залежністю, належать одній таблиці.

Його ще називають унарним зв’язком.

Існує три класи рекурсивних зв’язків:

а) 1:1 (список);

б) 1:M (дерево);

в) M:M (сітка).

а) Рекурсивний зв’язок «один до одного» (рис.8.9) реалізує структуру даних типу «список».

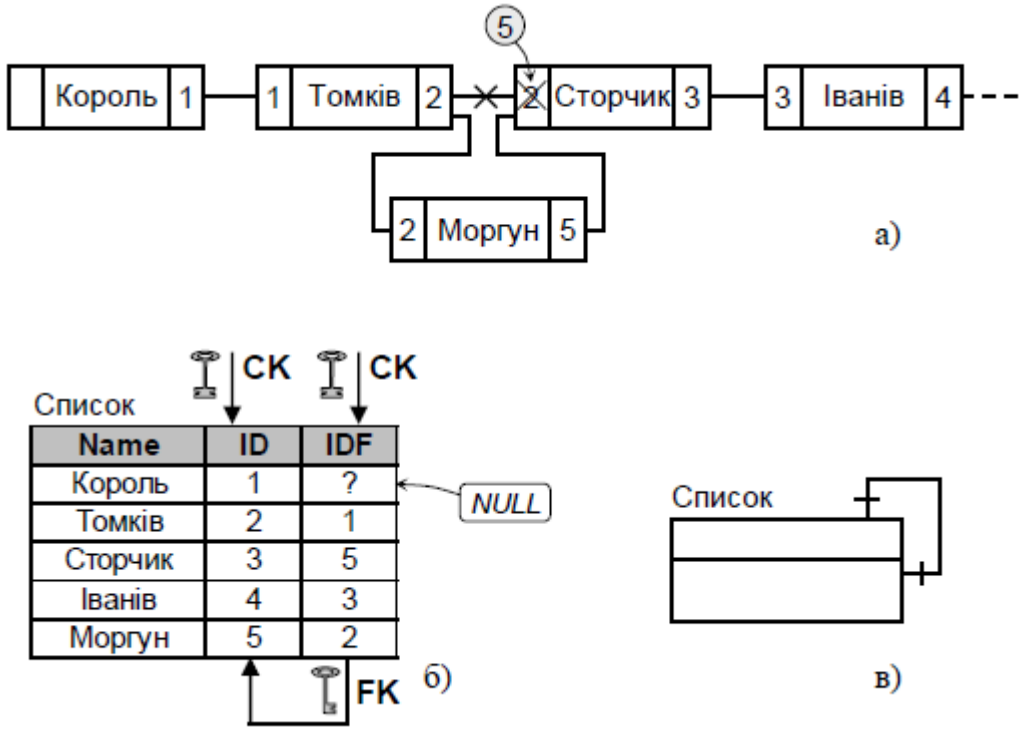

Рисунок 8.9. Рекурсивний зв’язок «один до одного»

а) реалізація; б) альтернативна реалізація;

в) спрощена структурна схема

Рекурсивний зв’язок «один до багатьох» (рис.8.10) реалізує структуру даних типу «дерево».

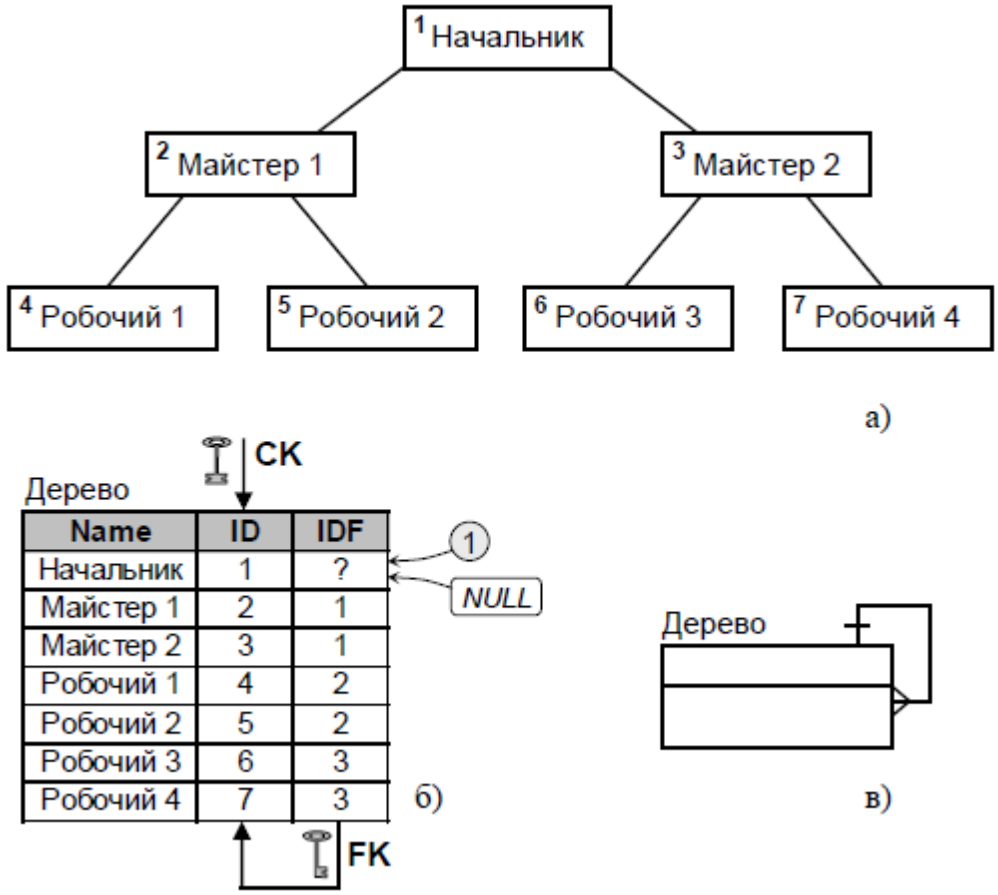

Рисунок 8.10. Рекурсивний зв'язок «один до багатьох»

а) ілюстрація; б) реалізація; в) спрощена структурна схема

Для зв'язків 1:1 та 1:М існує дилема «першого запису» (зовнішній ключ вимагає значення потенційного ключа для уже внесеного у таблицю запису). Є два шляхи вирішення цієї дилеми:

1. Перед створенням зовнішнього ключа необхідно ввести у таблицю перший запис, який потім буде посилатися на себе ж самого, або, навпаки, створити зовнішній ключ, а потім відмінити його дію на деякий час, поки не буде внесено перший запис.

2. Стовпець для зовнішнього ключа повинен дозволяти введення NULL-значення. Таким чином, з'явиться можливість ввести перший запис, що має NULL-значення у стовпці зовнішнього ключа, і тим самим уникнути необхідності примусового введення першого запису.

Рекурсивний зв'язок «багато до багатьох» (рис.8.11) реалізує структуру даних типу «сітка». Наприклад, поліклініка у якій лікарі лікують один одного.

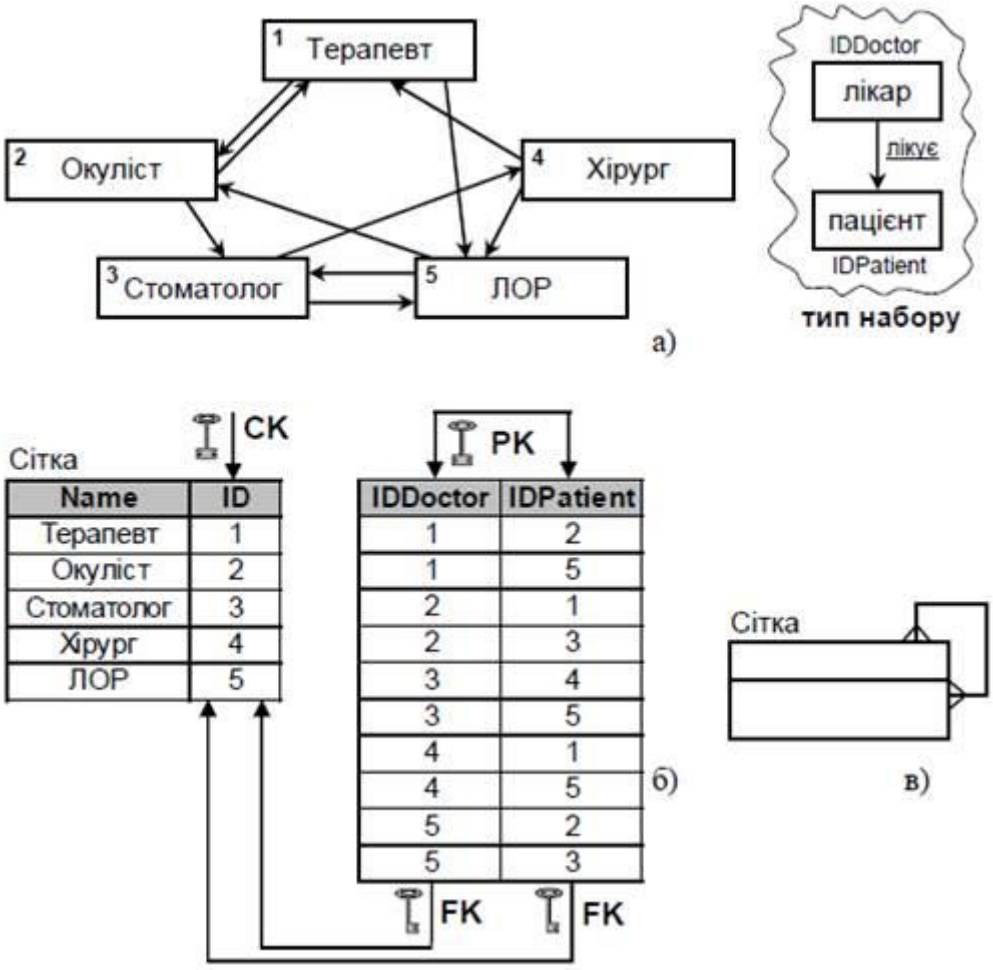

Рисунок 8.11 Рекурсивний зв'язок «багато до багатьох»

а) ілюстрація; б) реалізація; в) спрощена структурна схема

Небінарні відношення

Всі відношення, які були розглянуті до цього належать до унарних (бінарних) відношень, тому що утворюються між двома сутностями.

Небінарні зв'язки – це зв'язки між трьома та більше таблицями.

Приклад потрійного (тернарного) зв'язку (рис. 8.12):

1. є певний асортимент піц;

2. є перелік клієнтів, що замовляють ці піци.

Потрійний зв'язок полягає у тому, що будь-які піци можуть бути доставлені різним клієнтам, і при цьому довільним розношиком.

Аналогія: студент, викладач, знання про певний предмет.

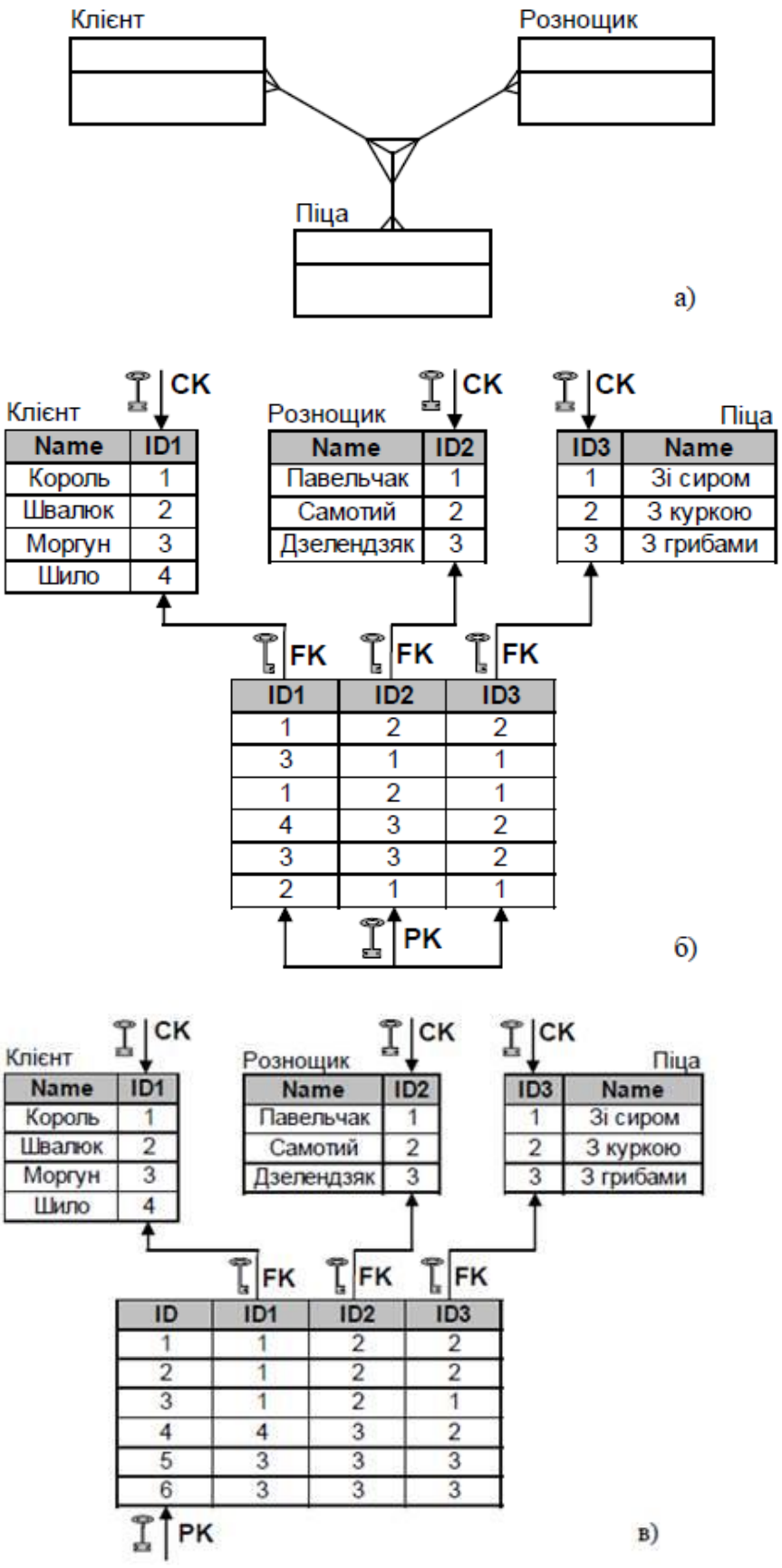

Рис. 9. Тернарний зв'язок а) спрощена структурна схема; б) реалізація; в) альтернативна реалізація