МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

УКРАЇНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

С. І. ДОЦЕНКО

ОРГАНІЗАЦІЯ ТА СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ

Навчальний посібник

Харків 2023

УДК 004.65(075)
Д 714

Рекомендовано вченою радою Українського державного університету залізничного транспорту як навчальний посібник (витяг з протоколу № 4 від 20 квітня 2021 р.)

Рецензенти:
професори Г. А. Кучук (НТУ «ХПІ»),
С. О. Тимчук (ХНТУСГ ім. П. Василенка)

Д 714 Доценко С. І. Організація та системи керування базами даних: Навч. посібник. – Харків: УкрДУЗТ, 2023. – 117 с., рис. 92, табл. 3.

ISBN

Викладено теоретичні основи організації та систем керування базами даних.

Навчальний посібник призначений для здобувачів вищої освіти денної та заочної форм навчання освітніх програм першого рівня, які навчаються за спеціальністю 123 «Комп'ютерна інженерія», 126 «Інформаційні системи і технології», і фахівців у галузі інформаційних технологій.

УДК 004.65(075)

ISBN

ЗМІСТ

ВСТУП

Бази даних та бази знань відіграють вирішальну роль при розробленні та застосуванні інформаційних систем, які є основою для комп'ютеризації в усіх галузях промисловості, сільського господарства, медицини, державному та приватному секторах управління. Розвитку баз даних сприяють успіхи в розробленні її теоретичних основ, а також у практичній реалізації баз даних для різних сфер.

На основі цього зростає потреба у фахівцях, здатних розробляти і застосовувати інформаційні системи на основі баз даних.

Основне призначення навчального посібника полягає у систематичному відображенні основних питань, які зустрічаються при проектуванні баз даних (в першу чергу реляційних баз даних), а також у розгляді актуальних напрямів розвитку баз даних.

Розділ 1. Загальні положення теорії баз даних

1.1. Технологічні основи баз даних

Виникнення технологій баз даних припадає на початок 1960-х років. Їхньому швидкому розвитку сприяли потреби в обробці інформації, досягнення в суміжних галузях інформаційних технологій, таких як операційні системи, мови програмування, технічне забезпечення. Спочатку зароджувалися певні ідеї щодо керування ресурсами даних, формувалися основи методології побудови систем баз даних. Із самого початку було зрозуміло, що цей напрям має самостійне значення і буде відігравати одну з ключових ролей у побудові інформаційних систем різного призначення.

На рис. 1.1 наведено склад інформаційних задач і функцій, виконуваних інформаційною системою.

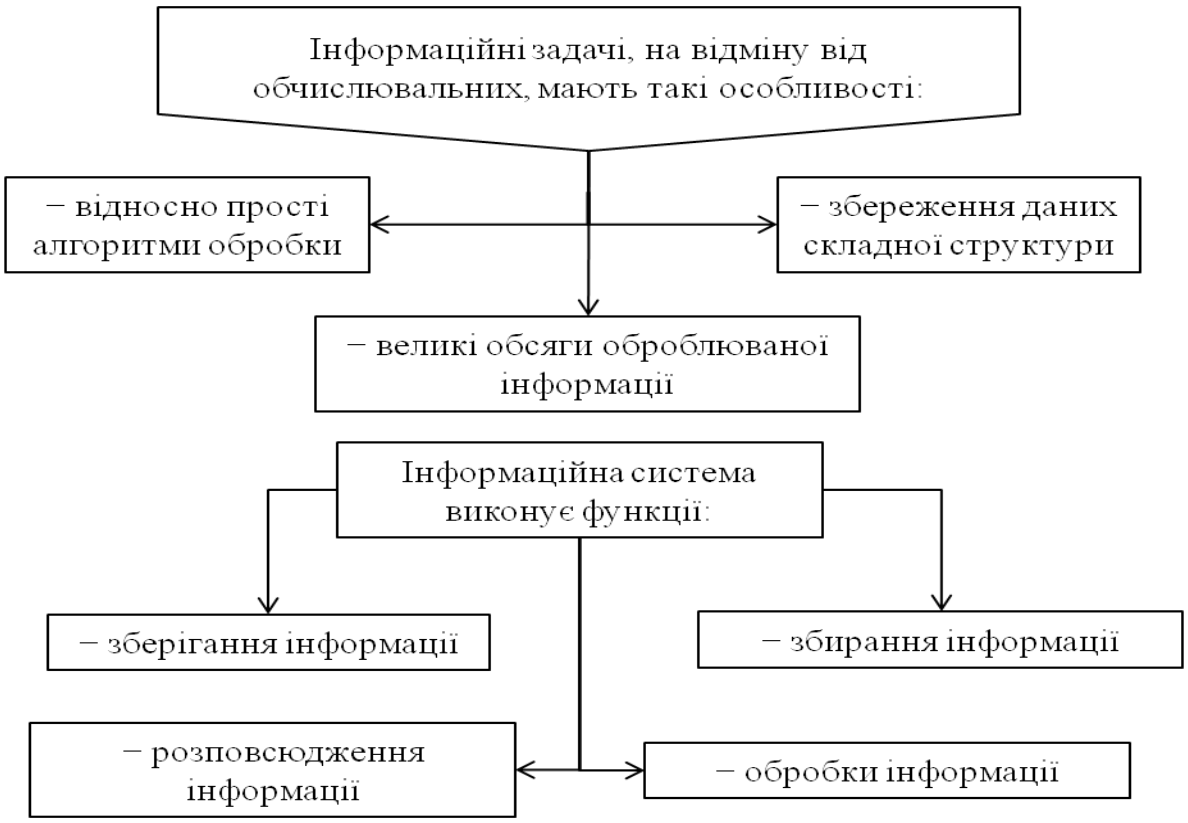


Рис. 1.1. Склад інформаційних задач і функцій, виконуваних інформаційною системою

Під *інформацією* розуміють будь-які відомості про будь-яку подію, сутність, процес, що є об'єктом певних операцій: передачі, перетворення, зберігання або використання.

Дані визначаються як інформація, зафіксована в певній формі, придатна для подальшої обробки, зберігання і передачі (рис. 1.2).

Керування в інформаційних системах виконують системи керування базами даних (СУБД).

На рис. 1.2 наведено форми зв’язків поміж елементами подання предметної галузі та відповідними інформаційними технологіями [1, 2].

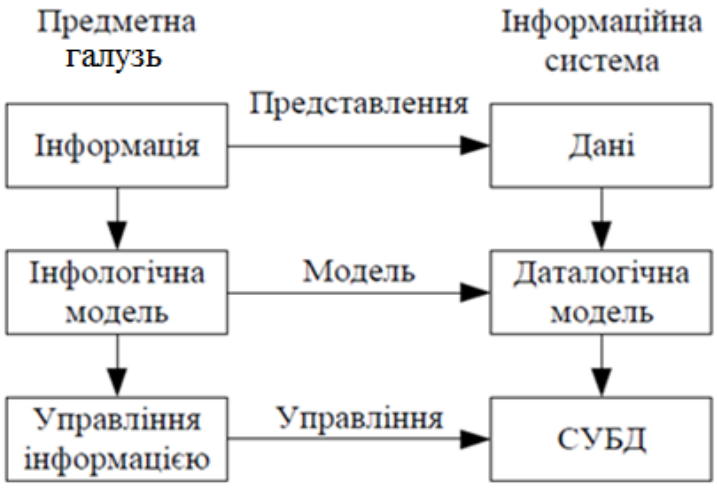


Рис. 1.2. Зв’язок між інформацією і даними

Предметна галузь – галузь застосування конкретної бази даних.

Інфологічне подання розглядає питання, пов'язані зі змістом інформації, незалежно від подання її в пам'яті комп'ютера.

Даталогічна модель розглядає питання подання даних у пам'яті комп'ютера.

Класифікація теоретико-графових моделей наведена на рис. 1.3. **До переваг** цих БД можна віднести ефективне використання пам'яті комп'ютера, швидке виконання основних операцій над даними. **Недоліками** цих БД є:

− труднощі при обробці інформації з достатньо складними зв'язками;

− складність розуміння змісту звичайним користувачем;

− залежність від фізичної реалізації.

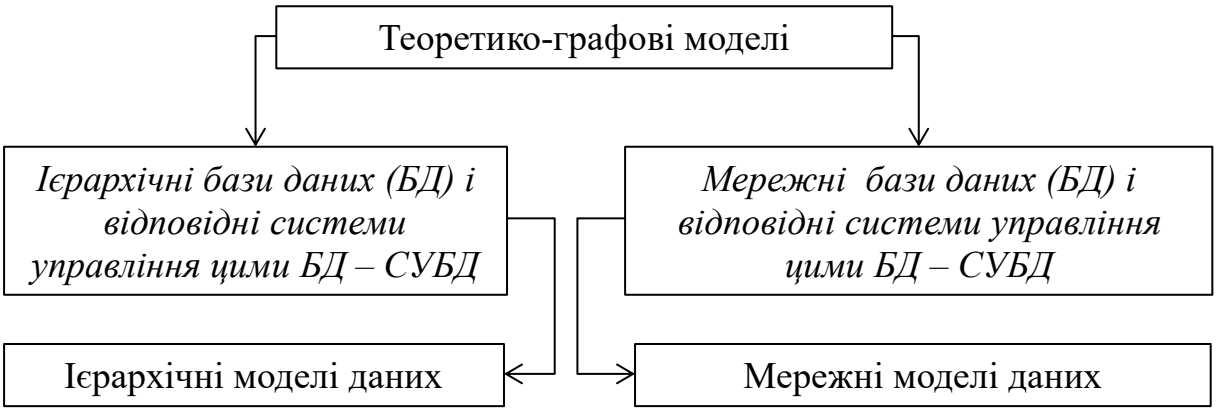


Рис. 1.3. Класифікація теоретико-графових моделей

В основі визначених форм БД лежали відповідні моделі даних: ієрархічні і мережеві (рис. 1.4) [1, 2].

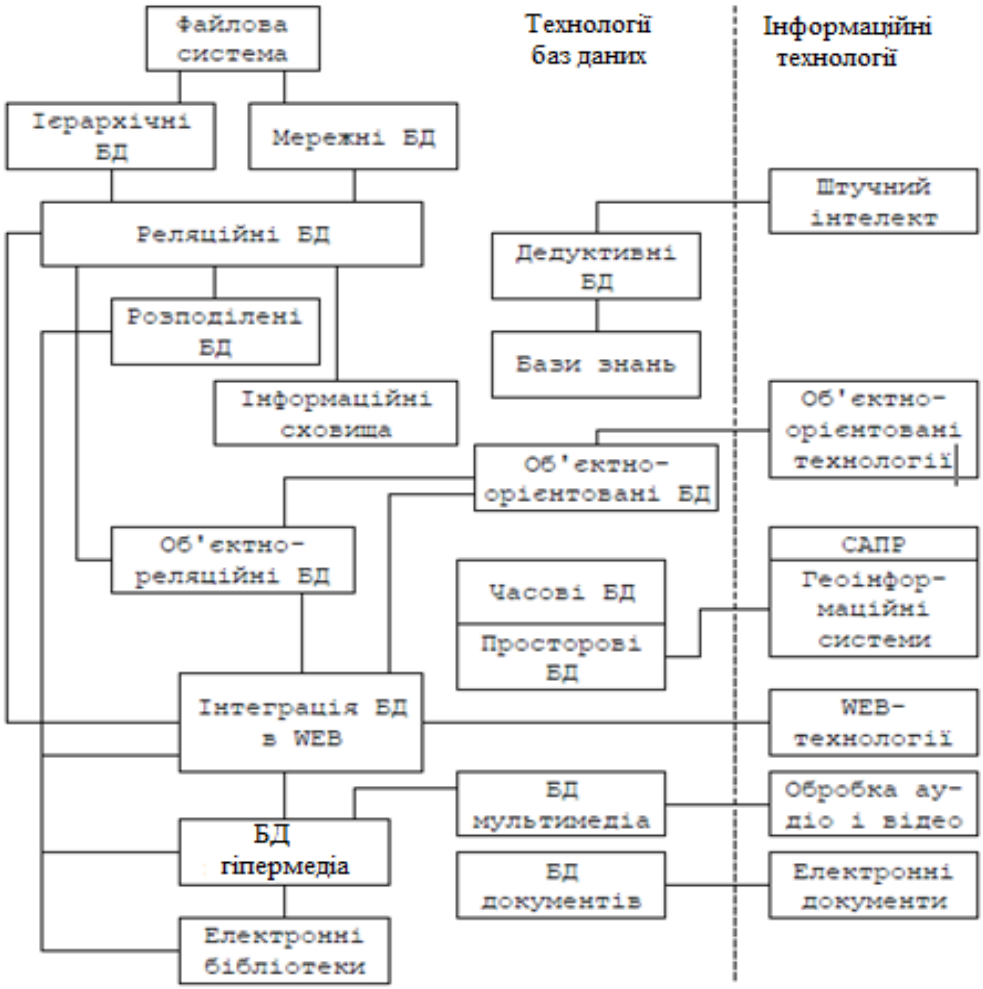


Рис. 1.4. Етапи розвитку баз даних

На рис. 1.5 наведено класифікацію математичних моделей даних.

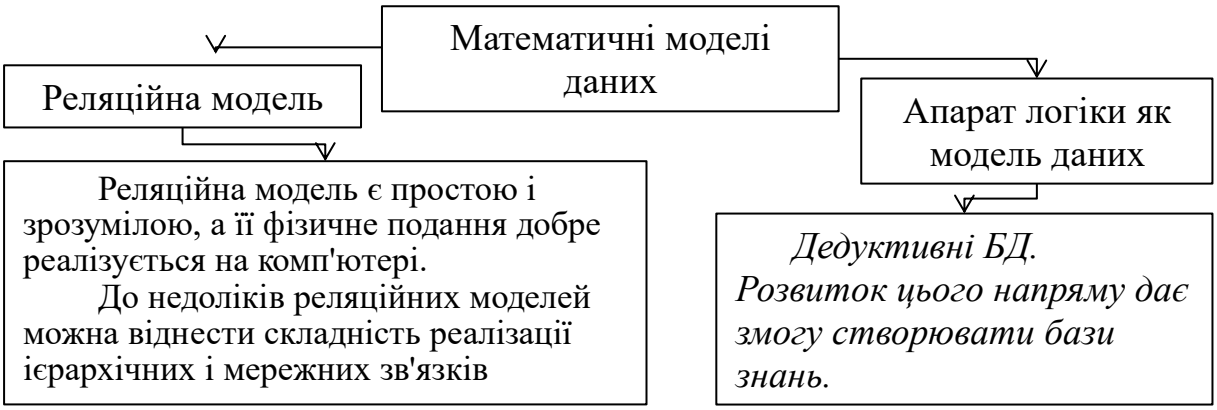


Рис. 1.5. Класифікація математичних моделей даних

На рис. 1.6 наведено класифікацію математичних моделей даних з використанням об’єктно-орієнтованого програмування.

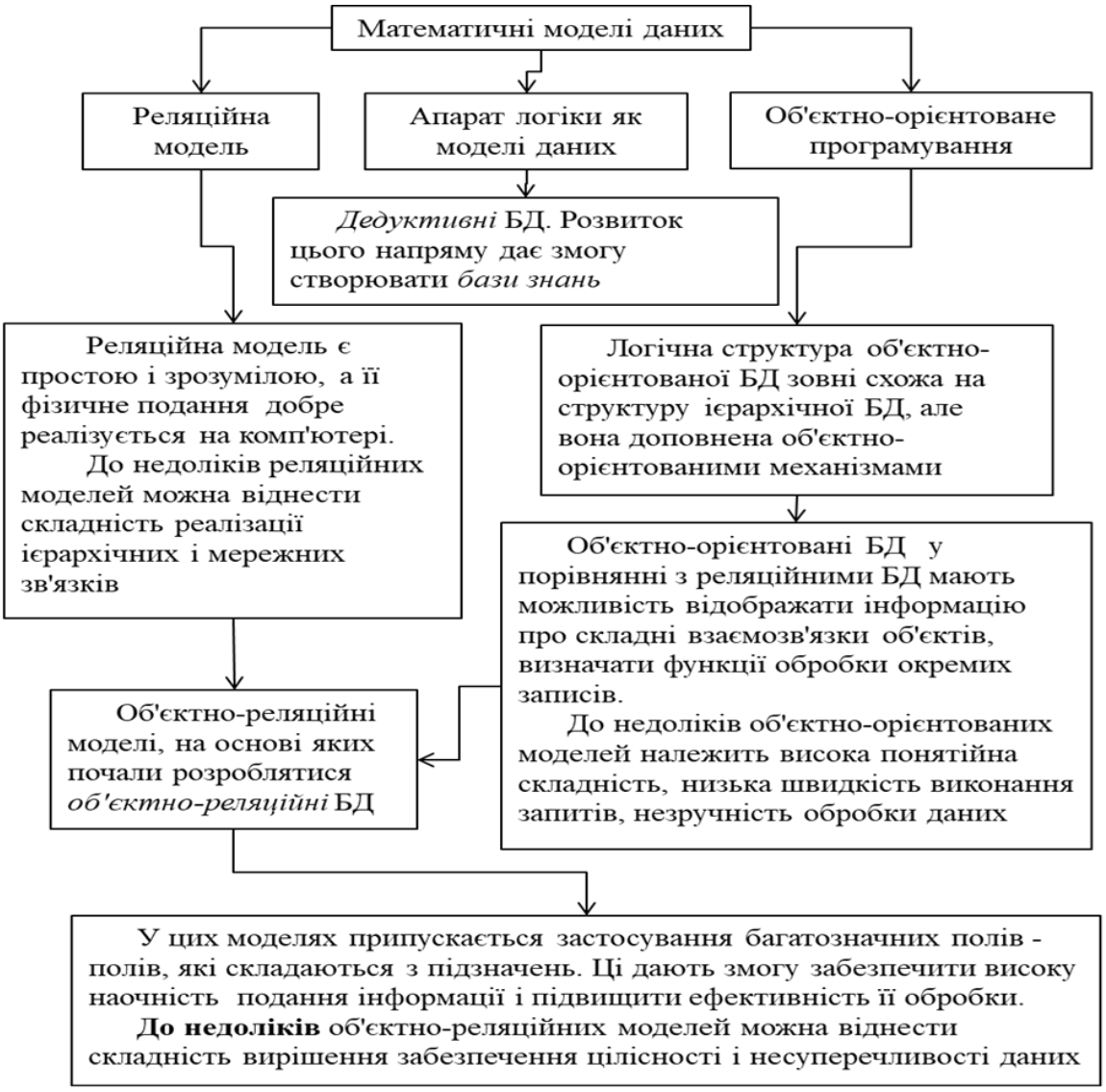


Рис. 1.6. Класифікація математичних моделей даних з використанням об’єктно-орієнтованого програмування

У 1980-х роках реляційні БД набули домінуючого положення. Однак уже тоді існувало і постійно розширювалося коло застосувань, для яких ця технологія була неадекватною. Це стосується мультимедійних систем, що оперують просторовими даними, і т. ін. У середині 1980-х років успіхів досягло об'єктно-орієнтоване програмування.

На рис. 1.7 наведено склад технологій, застосовуваних в інформаційних сховищах.

Рис. 1.7. Склад технологій, застосовуваних в інформаційних сховищах

Одним з найбільших досягнень 1990-х років у галузі інформаційних технологій стало створення відкритої глобальної розподіленої неоднорідної гіпермедійної інформаційної системи, що використовує комунікаційну мережу Internet. Ця система отримала назву WWW (Web).

З самого початку були спроби інтегрувати системи БД у Web. Одним з напрямів роботи є ***інтеграція структурованих даних*** *БД і слабкоструктурованих даних Web*, проводяться роботи зі створення БД мовою XML.

На сьогодні в багатьох комп'ютерних компаніях здійснюються роботи зі створення ***цифрових бібліотек*** інформаційних систем, призначених для зберігання, пошуку, обробки, аналізу і розповсюдження інформаційних ресурсів різної природи – структурованих і слабкоструктурованих даних, мультимедійної інформації, повнотекстових документів.

1.2. Банки даних

Банк даних – це система спеціальним чином організованих даних (баз даних), програмних, мовних, технічних, організаційно-методичних засобів призначених для підтримки інформаційної моделі предметної галузі з метою забезпечення інформаційних потреб користувачів (рис. 1.8) [1, 2].

Рис. 1.8. Банк даних

На рис. 1.9 наведено визначення елементів банку даних.

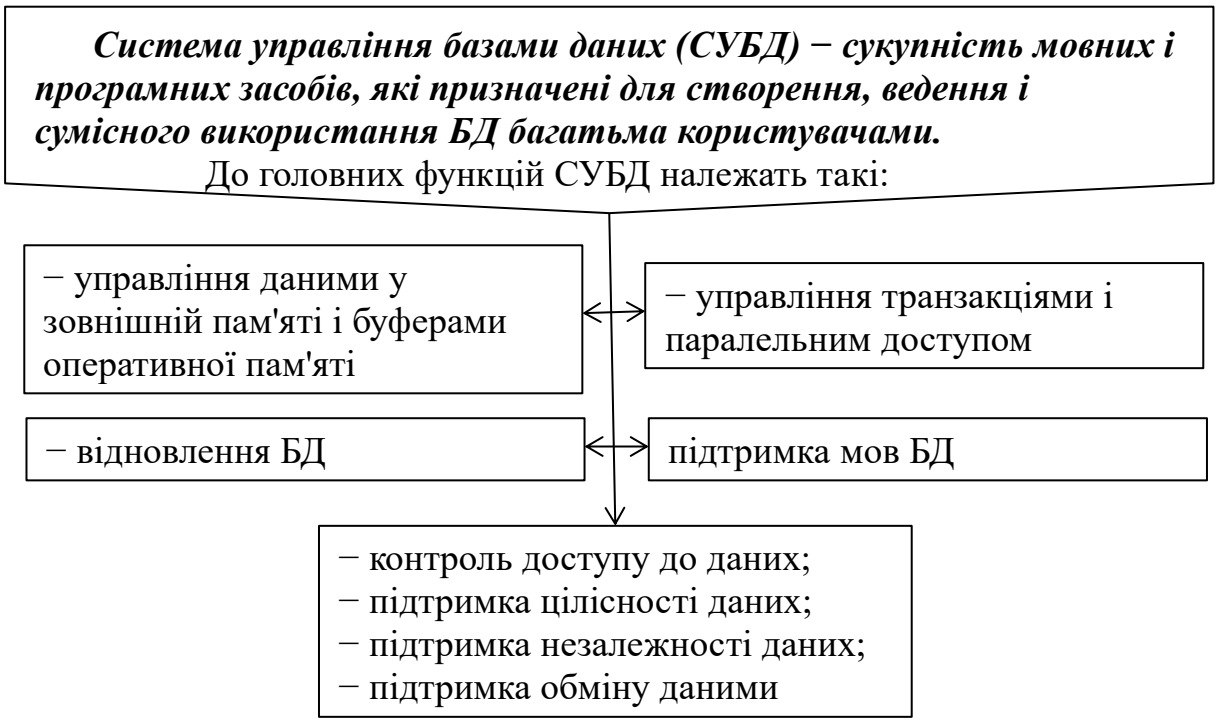


Рис. 1.9. Визначення елементів банку даних

Переваги і недоліки застосування СУБД наведені в табл. 1.1 [1, 2].

Таблиця 1.1

Переваги і недоліки застосування СУБД

Переваги СУБД	Недоліки СУБД
Мінімізація збитковості даних	Використання частини ресурсів на потреби СУБД
Несуперечливість даних і контроль їхньої цілісності	Вартість СУБД
Незалежність прикладних програм від даних	Підвищені вимоги до технічного і програмного забезпечення
Підвищена безпека	Продуктивність
Розвинені засоби резервного копіювання і відновлення	Підвищені вимоги до кваліфікації працівників
Багатокористувацький режим роботи	Наслідки збоїв

Банк даних − це система спеціальним чином організованих даних (баз даних), програмних, мовних, технічних, організаційно-методичних засобів, призначених для підтримки інформаційної моделі предметної галузі з метою забезпечення інформаційних потреб користувачів (рис. 1.3).

База даних − сукупність взаємопов'язаних даних, що знаходяться під керуванням СУБД. У БД зберігаються дані, логічно пов'язані між собою.

До головних властивостей БД належать такі:

− ***цілісність*** означає, що в будь-який момент часу відомості в БД мають бути несуперечливими;

− ***безпека*** означає, що виконується захист даних від санкціонованого і несанкціонованого доступу;

− ***відновленість*** означає можливість відновлення БД після збоїв роботи системи.

На рис. 1.10 наведено склад елементів бази даних і їхні характеристики.

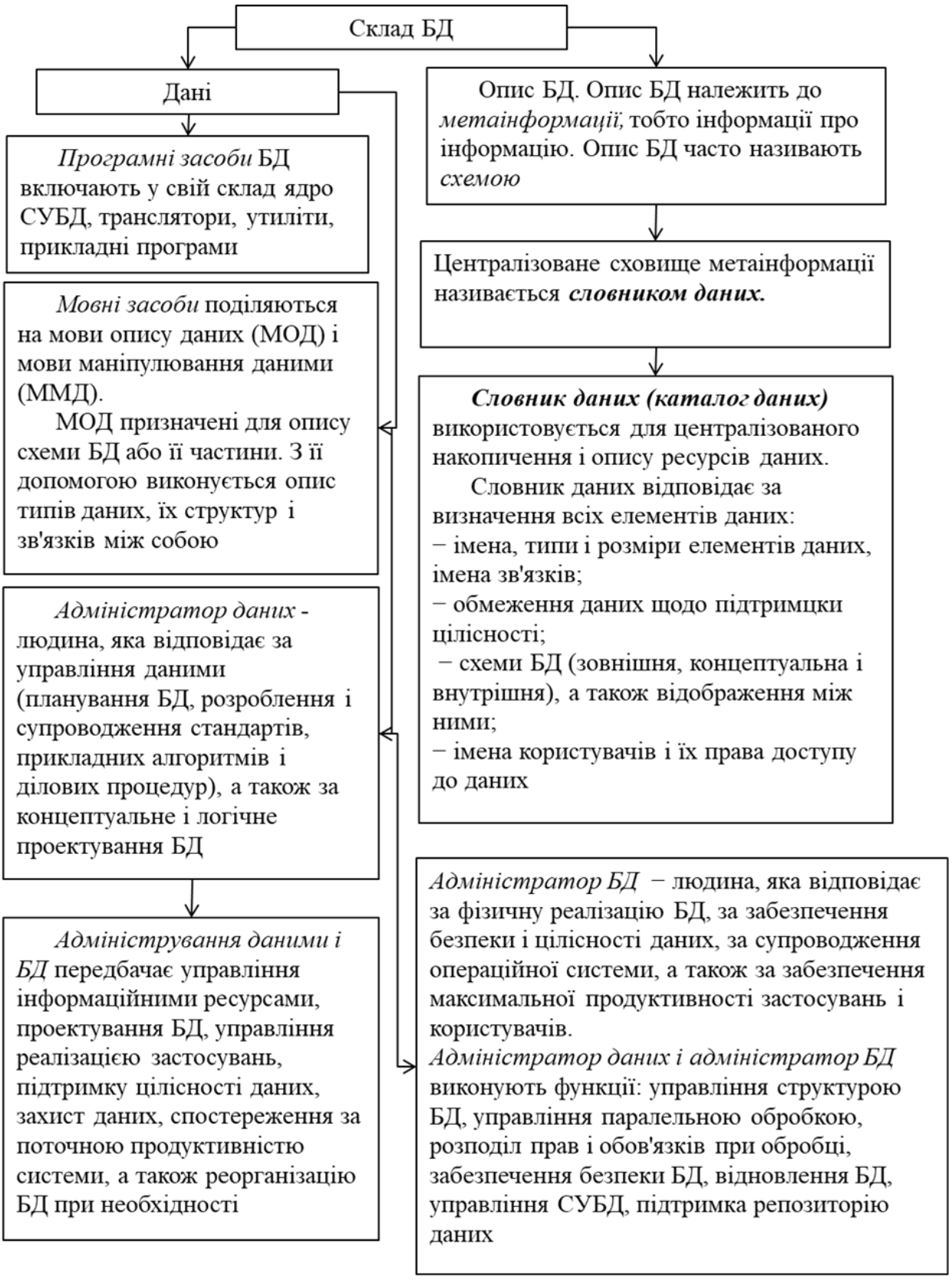


Рис. 1.10. Склад елементів бази даних і їхні характеристики

1.3. Складові елементи системи баз даних

На рис. 1.11 наведено склад компонентів системи баз даних.

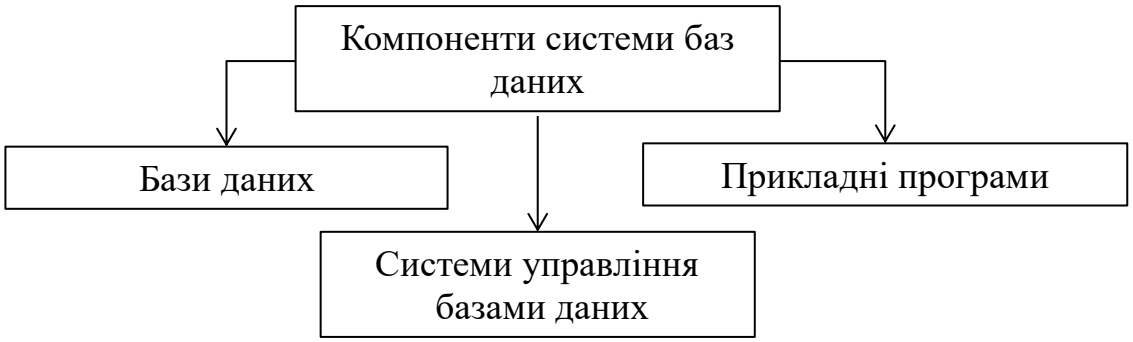


Рис. 1.11. Склад компонентів системи баз даних

Сучасні БД переважно подають користувачу дані у вигляді таблиць.

Ядро СУБД містить сукупність базових механізмів СУБД, використовуваних при будь-яких варіантах конфігурації системи.

Ядро СУБД виконує функцію посередника між підсистемами засобів проєктування і обробки даних.

Ядро СУБД отримує запити від інших компонентів у термінах таблиць, стовпців, рядків і перетворює ці запити на команди операційної системи, що виконують запис і читання з фізичних носіїв інформації.

Ядро СУБД задіяне в керуванні транзакціями, блокуваннях, резервному копіюванні і відновленні.

На рис. 1.12 наведено елементи системи, що скдадають базу даних [2].

Склад елементів ядра системи керування бази даних наведено на рис. 1.13.

Додатково елементи ядра СУБД включено до складу керування транзакціями, блокуванням, резервним копіюванням і відновленням. До ядра СУБД входять менеджери буферів, даних, транзакцій, журналів.

Підсистема засобів проєктування являє собою набір інструментів, що спрощують проєктування і реалізацію баз даних і їх застосування. Як правило, цей набір містить засоби для

створення таблиць, форм, запитів і звітів. В СУБД є також мови програмування та інтерфейси до них.

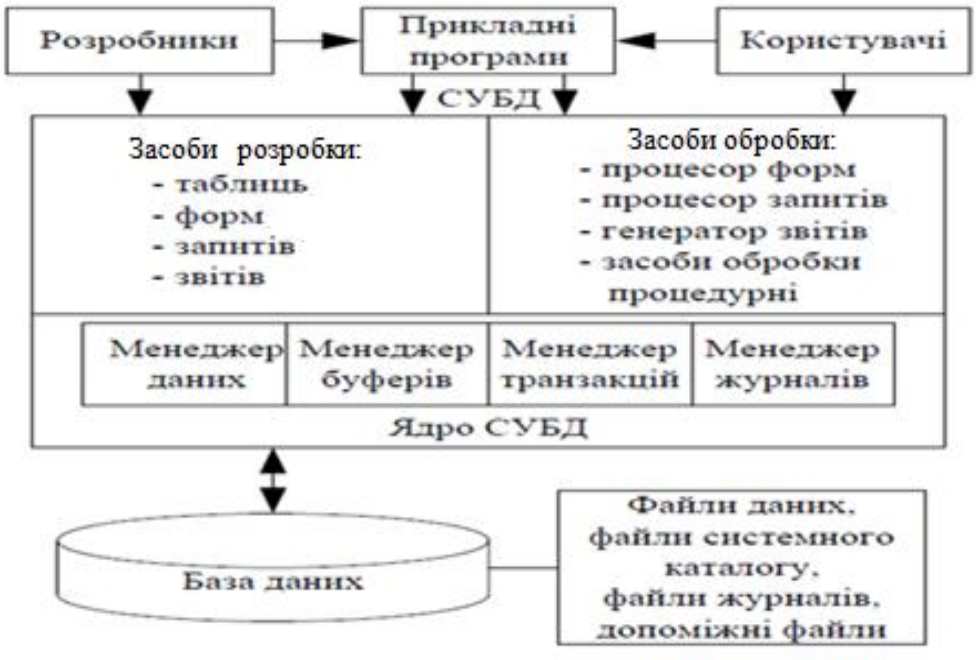


Рис. 1.12. Компоненти системи бази даних

Рис. 1.13. Склад елементів ядра системи керування бази даних

Підсистема обробки здійснює обробку компонентів застосування, створених за допомогою засобів проєктування.

Застосування БД складається з форм, запитів, звітів, меню і прикладних програм.

Форми, запити і звіти можна створювати за допомогою засобів, що постачаються в комплекті з СУБД.

Прикладні програми мають бути написані або вхідною мовою СУБД, або однією зі стандартних мов, а потім за допомогою СУБД з'єднані з БД.

Контрольні запитання до розділу 1

1. Навести визначення поняття «технології баз даних».
2. Навести склад компонентів банків даних.
3. Навести склад компонентів системи баз даних.
4. Навести визначення поняття «дані».
5. Навести визначення поняття «інформація».
6. Навести визначення поняття «база даних».
7. Навести визначення поняття «банк даних».
8. Навести етапи розвитку баз даних.

Розділ 2. Елементи баз даних

2.1. Моделі баз даних

Для організації роботи з БД необхідно забезпечити ***незалежність прикладних програм від даних***. Це обумовлено тим, що при зміні системи, а також для забезпечення ефективного обслуговування користувачів необхідно виконувати роботи щодо зміни методів зберігання даних у БД, шляхів доступу до даних, змінювати структури і формати даних і зв'язки між ними.

Незалежність застосувань від даних забезпечується засобами СУБД. Цей підхід базується на тому, що користувачі, застосовуючи БД, не знають внутрішнього подання даних.

На рис. 2.1 показана трирівнева модель архітектури СУБД, запропонована Комітетом планування стандартів і норм SPARC (Standarts Planning and Requirements Committee) Американського національного інституту стандартів ANSI (American National Standarts Institute) [2].

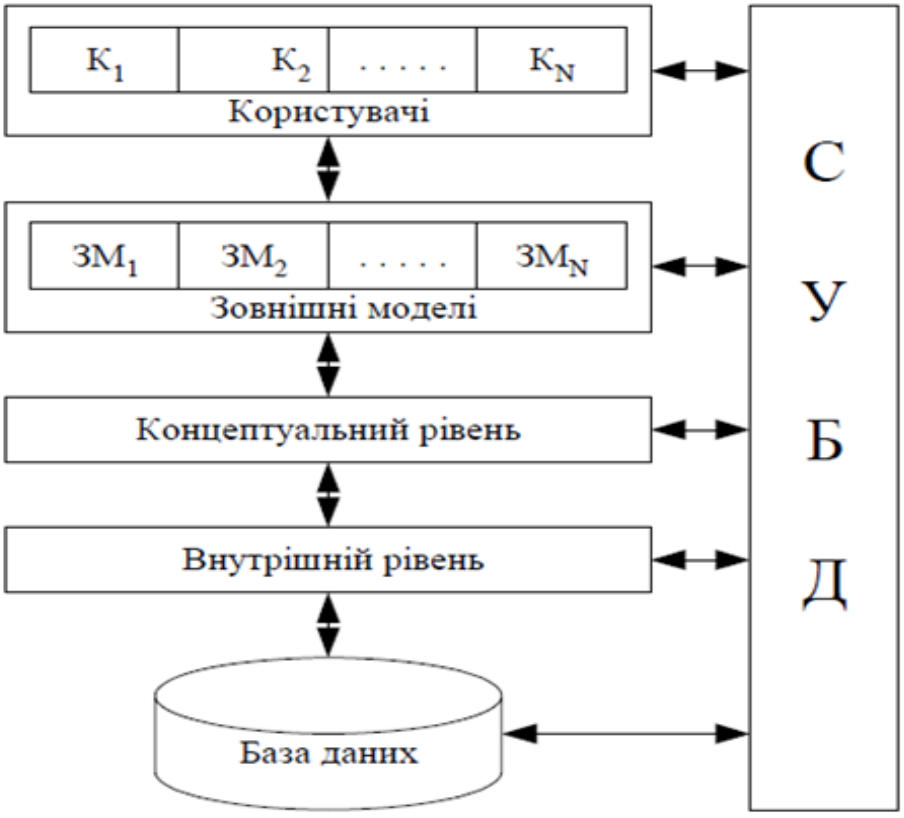


Рис. 2.1. Трирівнева модель архітектури СУБД

Опис структури даних називається *схемою.* На концептуальному рівні опис БД називається *концептуальною схемою.* Тут БД подано в найбільш загальному вигляді, що об'єднує дані, використовувані всіма застосуваннями, які працюють з БД. Фактично концептуальний рівень відображує модель предметної галузі, для якої створювалася БД.

На внутрішньому рівні опис БД називається *внутрішньою схемою.* Тут БД подано у вигляді безпосередньо даних, розташованих у файлах, що відповідають *фізичній організації* БД.

Крім трьох названих рівнів абстрагування, у БД існує ще один рівень, що передує їм. Цей рівень відображує інформацію про предметну галузь, а модель цього рівня називається *інфологічною моделлю* предметної галузі.

Перехід від одного рівня абстрагування до наступного і складає в загальному вигляді процес проєктування БД.

Трирівнева архітектура СУБД дає змогу забезпечити незалежність від даних. Це означає, що зміни на нижніх рівнях не впливають на верхні рівні. ***Розрізняють логічну і фізичну незалежність при роботі з даними***.

Логічна незалежність від даних означає захищеність зовнішніх схем від змін, що вносять до концептуальної схеми. Зміни концептуальної схеми БД не викликають необхідності в корегуванні існуючих зовнішніх схем для користувачів і відповідно зміну в застосуваннях, що працюють з цими схемами.

Фізична незалежність від даних означає захищеність концептуальної і зовнішніх схем від змін, що вносять до внутрішньої схеми. До змін внутрішньої схеми належить використання різних файлових систем або структур даних, різних пристроїв зберігання, модифікація пошукових структур тощо.

2.2. Моделювання даних

На рис. 2.2 наведено опис процесу моделювання даних з визначенням складу та характеристик моделі даних.

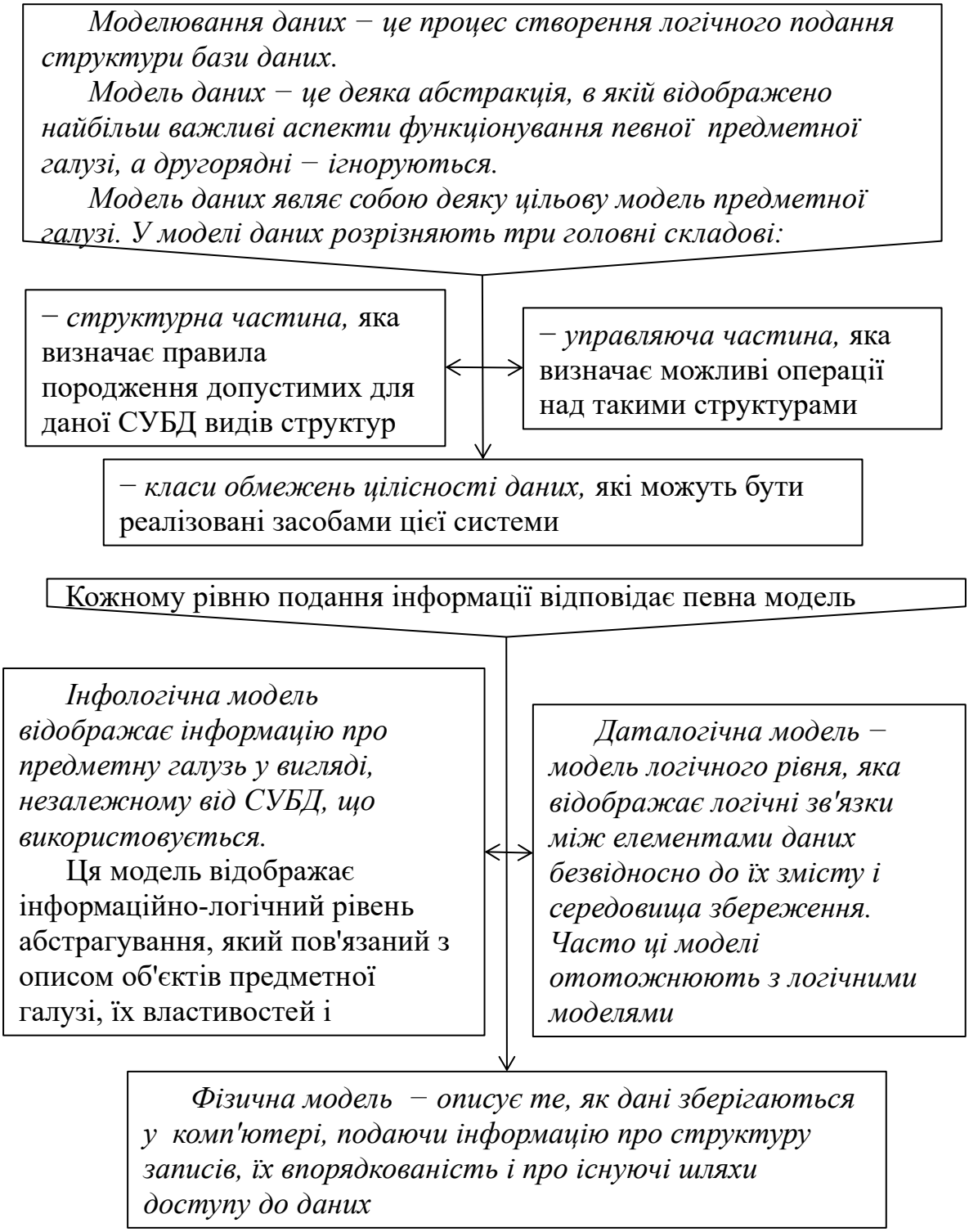


Рис. 2.2. Опис процесу моделювання даних з визначенням складу та характеристик моделі даних

На рис. 2.3 показана класифікація моделей даних [2, 3].

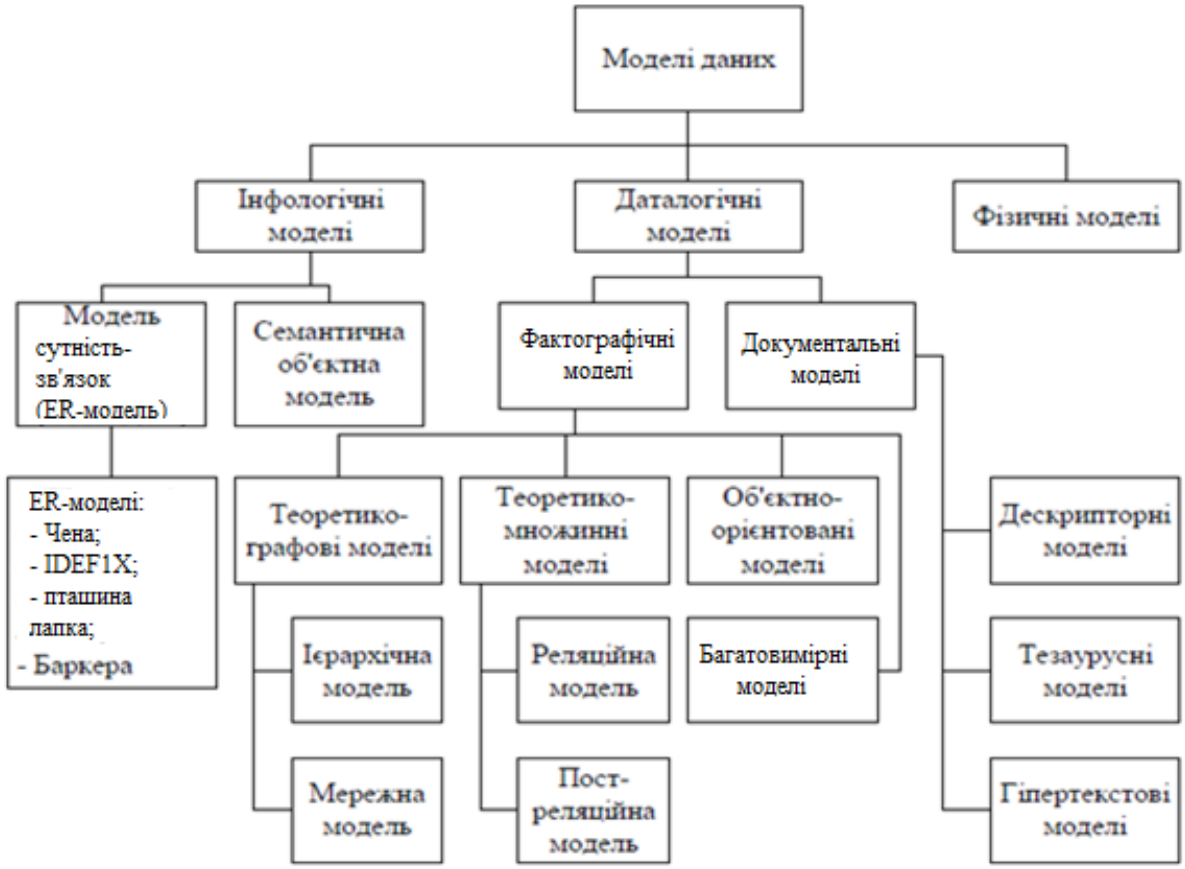


Рис. 2.3. Класифікація моделей даних

На рис. 2.4 наведено опис *інфологічних* моделей, а саме моделі «*сутність-зв'язок*» (ER-моделі).

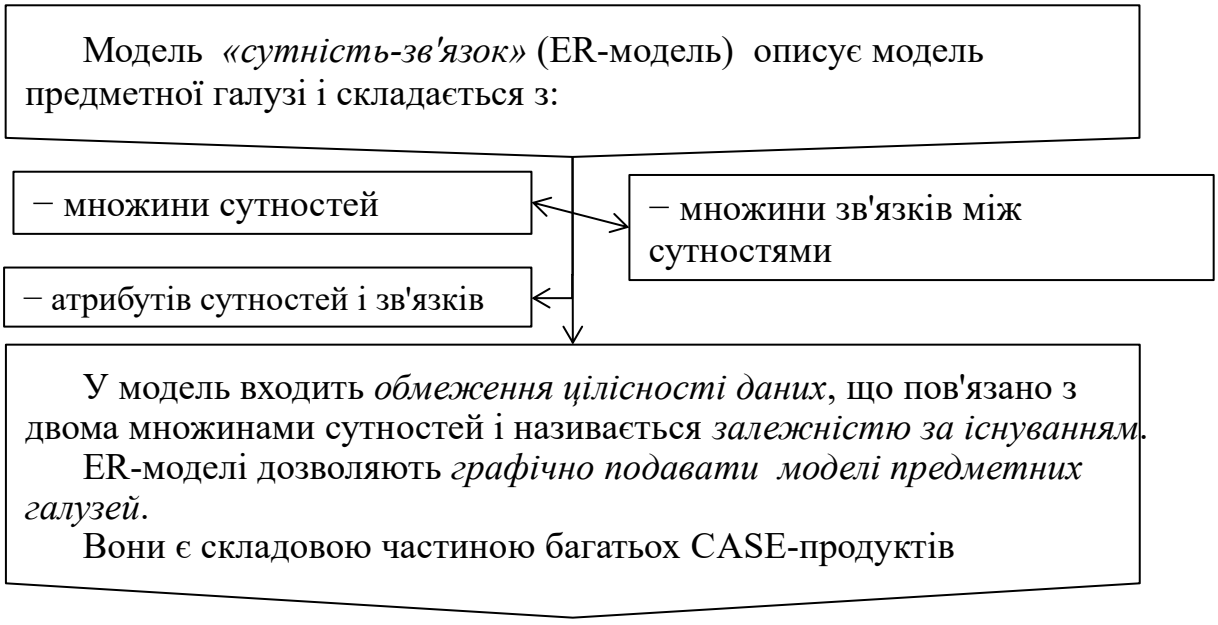


Рис. 2.4. Опис моделі «*сутність-зв'язок*» (ER-моделі)

Семантична об'єктна модель описує модель предметної галузі і являє собою *модель даних*.

Ця модель складається з *семантичних об'єктів*, що містять сукупність *атрибутів*. Атрибути групуються у *класи*.

Даталогічні моделі подано як дві моделі *логічного* рівня: *фактографічна* та *документальна*.

Теоретико-графова модель − модель даних, у якій дозволені структури даних можуть бути подані у вигляді *графа загального або спеціального вигляду*, наприклад *дерева*.

Необхідна група операцій мовою маніпулювання даними, що засновані на цій моделі, - це *навігаційні* операції.

Теоретико-множинна модель − модель даних, у якій використовується *математичний апарат реляційної алгебри*, реляційного обчислення, а операції над даними маніпулюють таблицями.

Мережева модель − модель даних, у якій *дозволені структури даних* можуть бути *подані у вигляді графа загального вигляду*.

Вершинами такого графа можуть бути дані різних типів − від атомарних елементів даних до записів складної структури. На відміну від ієрархічної моделі, наступник у цій моделі може мати довільну кількість батьків.

До *класу фактографічних моделей* належать *об'єктно-орієнтовані моделі*.

Об'єктно-орієнтована модель – модель даних, яка базується на понятті об'єкта, тобто сутності, що має стан і поведінку. Стан об'єкта визначається його атрибутами, а поведінка визначається сукупністю операцій, визначених для цього об'єкта. Також перед-бачається можливість підтримки зв'язків між типами об'єктів.

До *класу фактографічних моделей* належать також *теоретико-множинні моделі*.

Теоретико-множинна модель − модель даних, у якій використовується *математичний апарат реляційної алгебри*, реляційного обчислення, а операції над даними маніпулюють таблицями.

Реляційна модель − модель даних, заснована на математичному понятті відношення і поданні відношень у формі таблиць.

Постреляційна модель − розширена реляційна модель, яка *знімає обмеження неподільності даних*, що зберігаються в записах таблиць.

Ця модель допускає *багатозначні поля* − поля, значення яких складається з підзначень.

Набір значень багатозначних полів вважається самостійною таблицею, вбудованою в основну таблицю. Часто ці моделі ототожнюють з *об'єктно-реляційними моделями.*

Особливим класом фактографічних моделей є клас багатовимірних моделей.

Багатовимірна модель – модель даних, що оперує багатовимірним поданням даних (у вигляді *гіперкуба)* і орієнтована на підтримку аналізу даних.

Передбачається конструювання різноманітних агрегацій даних у межах *гіперкуба*, побудова різних його проєкцій – підмножин гіперкуба, деталізація і обертання даних, а також цілий ряд інших операцій.

Документальні моделі передбачають, що як одиничний елемент інформації виступає неподільний на менші складові частини документ, а інформація про документ, як правило, не структурується або структурується в обмеженому вигляді. У цих моделях в основному розглядаються тексти природною мовою, формати документів є вільними.

Дескрипторна модель описує кожен документ за допомогою дескриптора. Дескриптор має жорстку структуру і являє собою набори деяких лексичних одиниць (слів, словосполучень, термінів), потрібних для роботи з документами. Дескриптори між собою не пов'язані.

Тезаурусна модель описує кожен документ за допомогою дескрипторів, а також змістових відношень між лексичними одиницями (ціле-частина, род-вид, клас-підклас і т. ін.).

Ці моделі дають змогу підвищити ефективність дескрипторних моделей шляхом більш ефективного відображення предметної галузі.

Гіпертекстова модель − модель, заснована на розмічанні документа за допомогою спеціальних навігаційних конструкцій, що відповідають змістовим зв'язкам між різними документами або окремими фрагментами одного документа. Такі конструкції утворюють деяку семантичну мережу в базі документів.

2.3. Засоби баз даних

Важливим елементом баз даних є система керування базою даних, що у своєму складі містить низку складових частин (рис. 2.5).

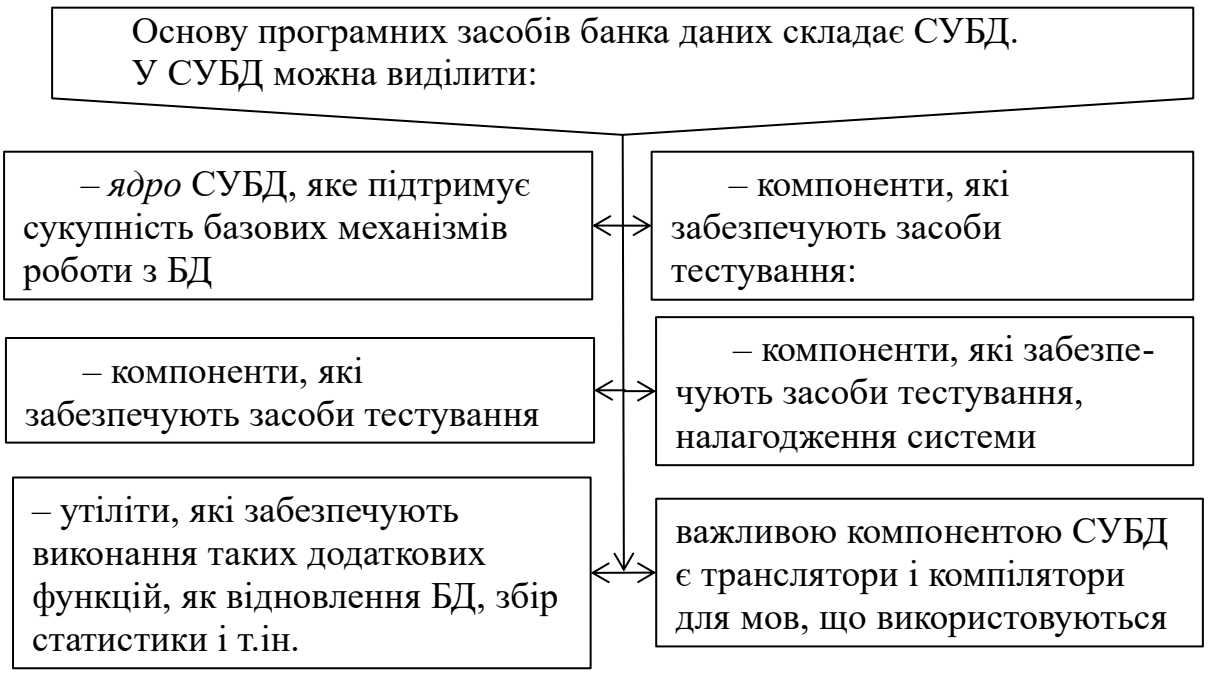


Рис. 2.5. Опис основних елементів системи керування базою даних

Для роботи з БД розробляються застосування.

Застосування – програма, що призначена для вирішення деякої сукупності завдань у певній предметній галузі або являє собою типовий інструментарій, застосовуваний у різних предметних галузях.

Застосування може використовувати різні джерела даних (фактографічні, документальні, АУЕВ і т. ін.), мати різну архітектуру (дволанкову, триланкову, розподілену).

Застосування бази даних – це застосування, що використовує ресурси деякої системи баз даних. Для доступу до БД використовується інтерфейс прикладного програмування СУБД, у середовищі якої він підтримується.

Застосування можуть бути написані стандартною алгоритмічною мовою програмування (Pascal, C, Basic тощо) із вбудованими операторами.

Важливим елементом баз даних є мовні засоби, застосовувані для розроблення баз даних. На рис. 2.6 наведено опис основних типів мов, застосовуваних у базах даних.

Рис. 2.6. Опис основних типів мов, застосовуваних у базах даних

Для формування запиту за допомогою різних СУБД найчастіше використовуються дві основні мови опису запитів:

– *SQL (Structured Query Language)* – структурована мова запитів;

– *QBE (Query By Example)* – мова запитів за зразком.

Головна відмінність між цими мовами полягає в тому, що мова *QBE* передбачає ручне або візуальне формування запиту, а мова *SQL* – програмування запиту.

Мова *SQL* є найбільш поширеною мовою для роботи з БД. На сьогодні існують такі міжнародні стандарти на мову *SQL: SQL1, SQL2, SQL3*.

Будь-яке *SQL*-застосування реляційної БД складається з трьох частин: інтерфейсу користувача, набору таблиць у БД і *SQL*-машини.

2.4. Моделі архітектури баз даних

Ефективність функціонування бази даних залежить від моделі її архітектури. Функціональні частини бази даних можуть розміщуватися на одному або декількох комп'ютерах. У разі, якщо база даних розміщується на одному комп'ютері, можливі такі варіанти використання програмних засобів [2]:

– застосування і СУБД;

– застосування і ядро СУБД;

– незалежне застосування.

У першому випадку взаємодія користувача і СУБД виконується або напряму через користувацький інтерфейс СУБД, або за допомогою застосування (рис. 2.7) [2].

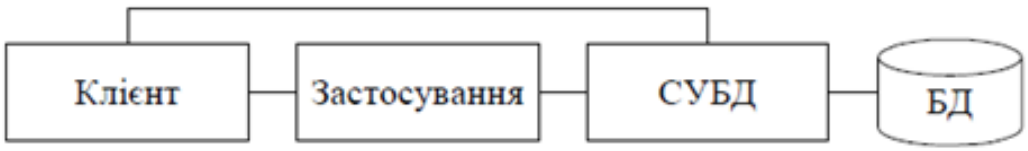


Рис. 2.7. Використання застосування і СУБД

У другому випадку взаємодія користувача і СУБД виконується за допомогою застосування (рис. 2.8). Такий підхід дозволяє підвищити швидкість роботи застосування, зменшити об'єм необхідної пам'яті [2].

Рис. 2.8. Використання застосування і ядра СУБД

Створення незалежних застосувань дає змогу звертатися до БД без СУБД (рис. 2.9). Такий підхід дає змогу ще більше підвищити швидкість роботи застосування, зменшити об’єм необхідної пам'яті. Недоліки такого підходу пов'язані з трудомісткістю доопрацювання застосувань, відсутністю стандартних засобів СУБД із обслуговування БД [2].

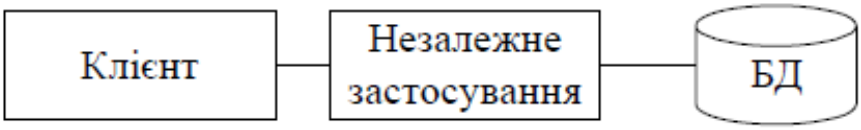


Рис. 2.9. Використання незалежного застосування

Найбільш поширеною є модель архітектури, у якій користувач маю свою персональну БД (КБД) і звертається до серверної БД (СБД) (рис. 2.10) [2].

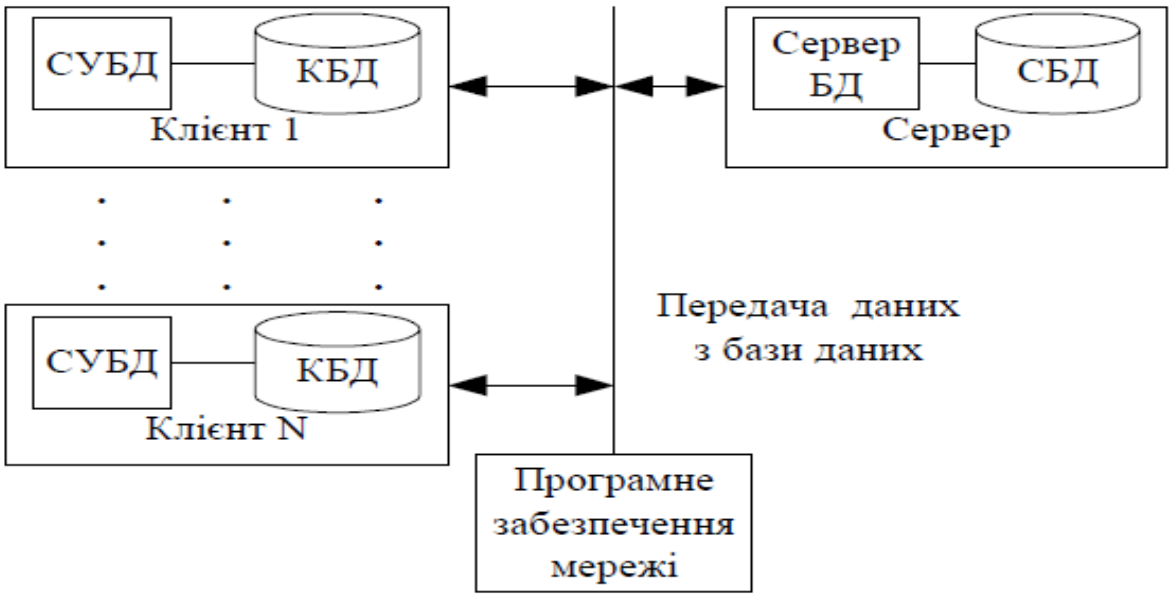


Рис. 2.10. Використання сервера БД

Під *сервером* розуміється комп'ютер або програма, що керують певними ресурсами.

Клієнт − це теж комп'ютер або програма, що використовують цей ресурс.

Контрольні запитання до розділу 2

1. Архітектура бази даних.
2. Назвати рівні архітектури бази даних.
3. Навести визначення концептуальної схеми БД.
4. Навести визначення внутрішньої схеми БД.
5. Пояснити механізми логічної незалежності від даних.
6. Пояснити механізми фізичної незалежності від даних.
7. Навести визначення поняття «модель даних».
8. Навести програмні і мовні засоби баз даних.
9. Навести архітектуру інформаційної системи.

Розділ 3. Системи керування базами даних

3.1. Місце систем керування базами даних у системах обробки інформації

На рис. 3.1 наведено опис функцій систем керування базами даних.

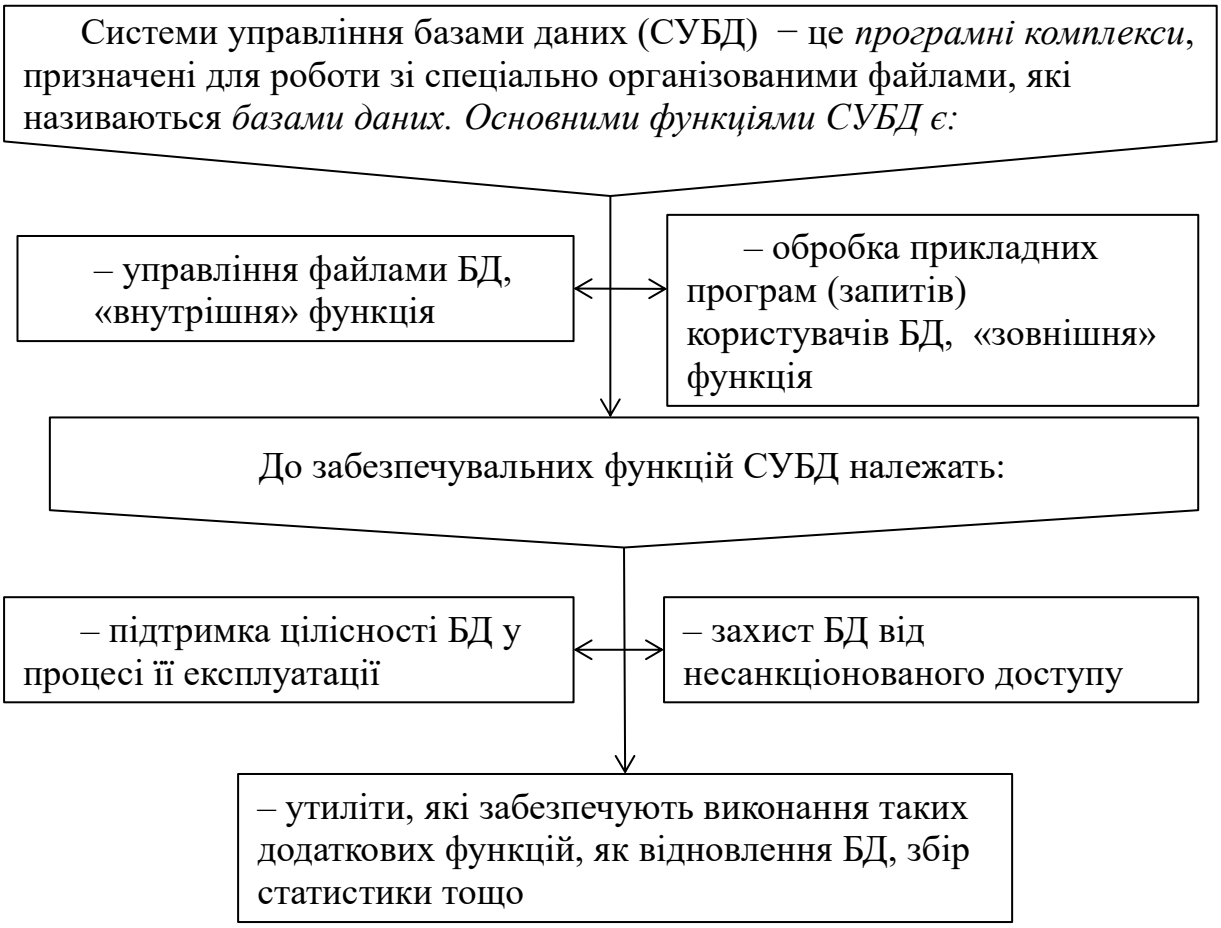


Рис. 3.1. Опис функцій систем керування базами даних

Крім того, сучасні СУБД часто оснащуються додатковими інструментальними засобами розроблення додатків (прикладних задач обробки даних).

СУБД як програмний продукт характеризується великою трудомісткістю виготовлення, високою наукомісткістю, тривалими термінами розроблення, значною вартістю.

У наш час на ринку інформаційних технологій пропонуються десятки різних СУБД, доведених до

«комерційного» (тобто рекомендованого до промислової експлуатації) зразка. Відповідно перед споживачем постає проблема усвідомленого вибору необхідної йому моделі, версії і конфігурації СУБД.

На рис. 3.2 наведено опис класів інформаційних систем, до складу яких входять бази даних.

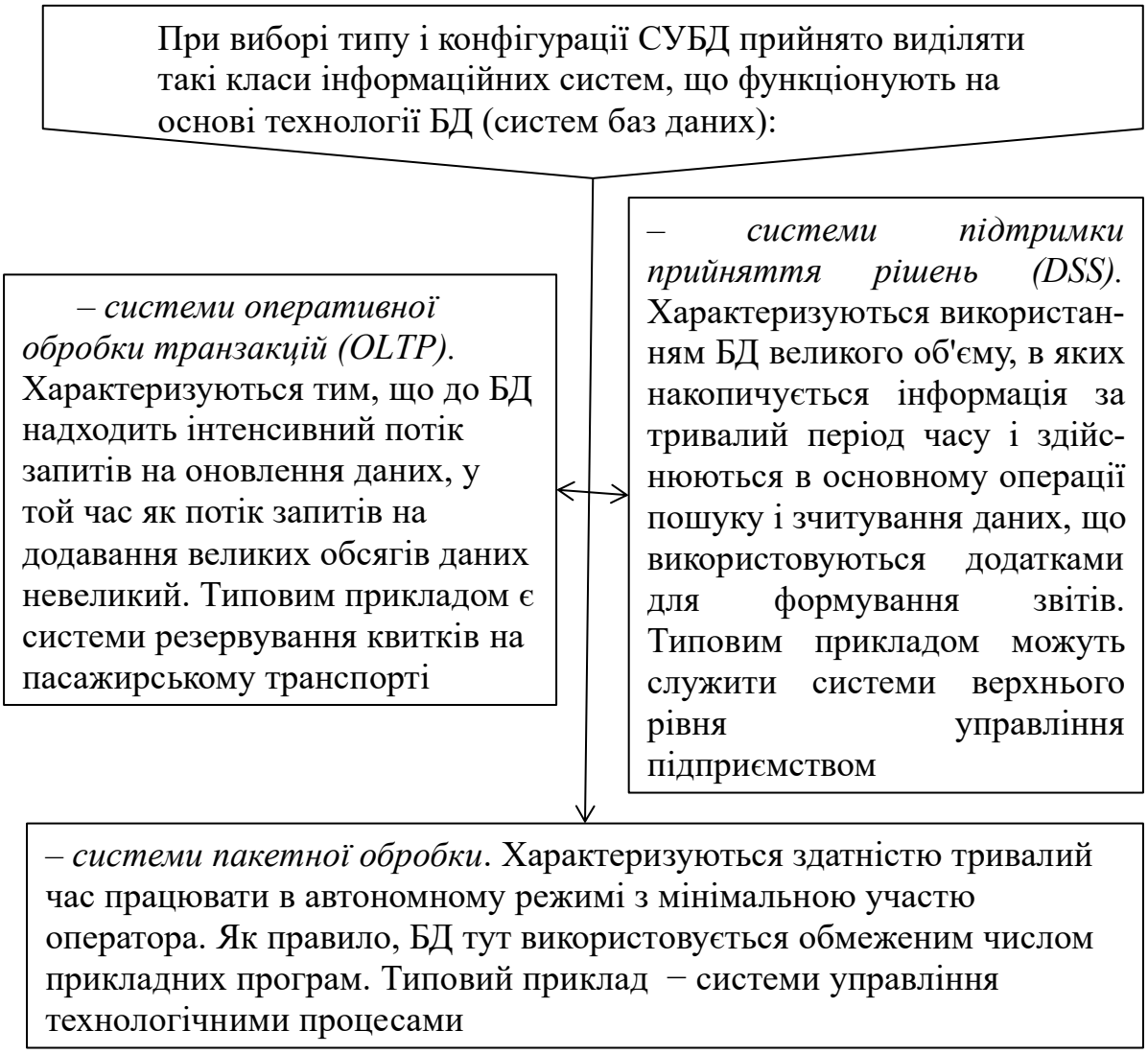


Рис. 3.2. Опис класів інформаційних систем, до складу яких входять бази даних

3.2. Функціонально-технологічні характеристики систем керування базами даних

Склад функціонально-технологічних характеристик систем керування базами даних наведено на рис. 3.3.

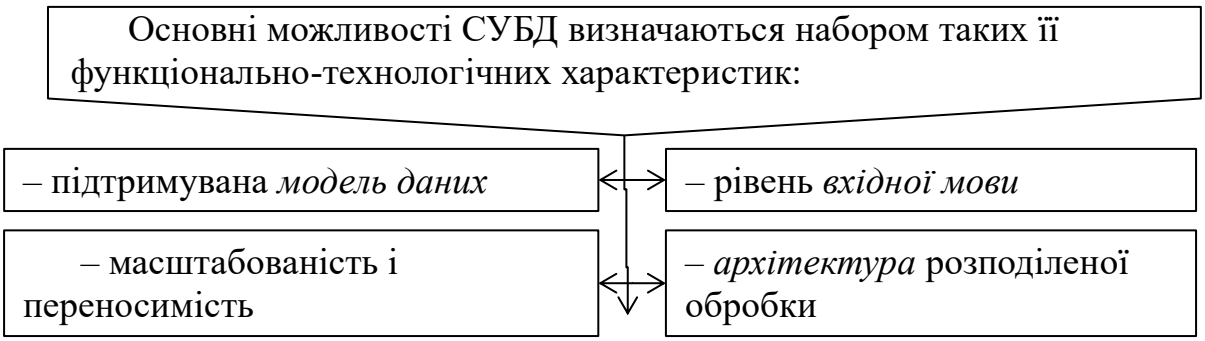


Рис. 3.3. Склад функціонально-технологічних характеристик систем керування базами даних

На рис. 3.4 наведено визначення змісту поняття «модель даних».

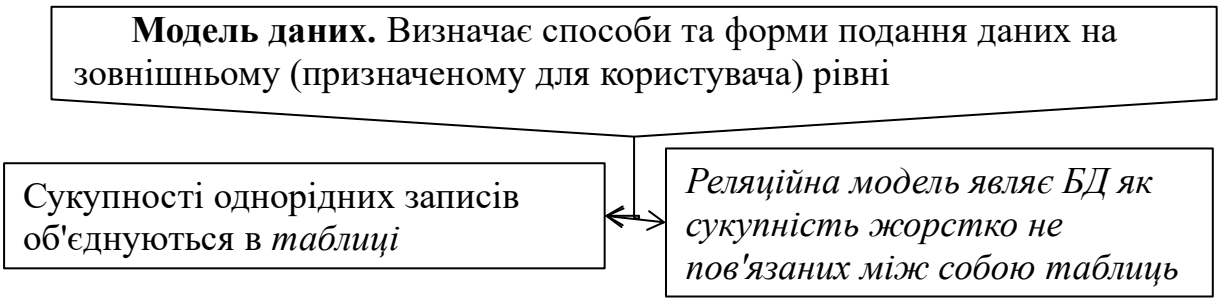


Рис. 3.4. Зміст поняття «модель даних»

Рівень вхідної мови. Підтримка моделі даних полягає в реалізації засобами СУБД мови обробки даних, що відповідає цій моделі. У загальному випадку СУБД можуть оснащуватися декількома вхідними мовами, відмінними за рівнем деталізації команд на обробку даних.

Мови дуже високого рівня реалізують обмежений набір операцій реляційної моделі та орієнтовані на кінцевого користувача. Вони дають змогу з найменшими затратами праці формулювати прості запити на обробку даних (набір яких, втім, може виявитися достатнім для вирішення всіх необхідних користувачеві завдань).

Мови, що реалізують операції реляційної моделі в повному обсязі, також характеризуються малим ступенем деталізації виконуваних команд і належать до мов обробки даних високого рівня.

Зараз в індустрії систем обробки даних склався стандарт реляційної мови високого рівня, що називається непроцедурним

SQL, підтримуваний більшістю сучасних СУБД (як правило, з деякими відхиленнями від стандарту).

Масштабованість і здатність до перенесення. Залежно від функціональних можливостей СУБД, застосування яких з технічних і економічних міркувань доцільно і можливо як в умовах скромних можливостей наявного парку ОТ, так і для побудови великомасштабних систем баз даних, можуть виникнути такі ситуації:

– існуюча СУБД піддається перенастроюванню, а додатки і БД залишаються без змін (найкращий варіант);

– встановлюється нова версія СУБД без (явної) модернізації БД і додатків;

– встановлюється нова версія СУБД з перезавантаженням БД, але без модифікації додатків;

– встановлюється нова версія СУБД, перезавантажується БД і перепрограмовуються додатки (найменш прийнятний варіант).

На рис. 3.5 наведено характеристики двох найважливіших архітектур розподілених СУБД.

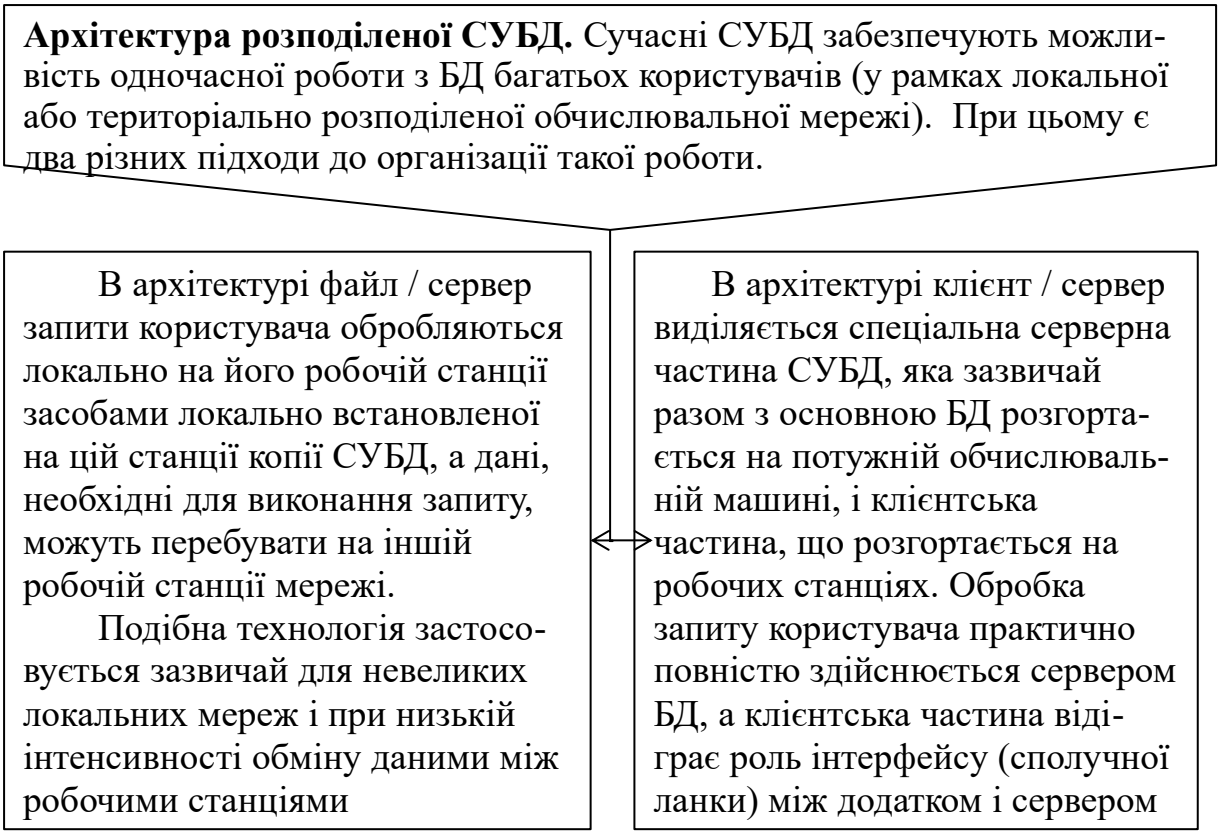


Рис. 3.5. Характеристики двох найважливіших архітектур розподілених СКБД

3.3. Вимоги до елементів систем керування базами даних

Якість СУБД визначається сукупністю характеристик, склад яких наведено на рис. 3.6.

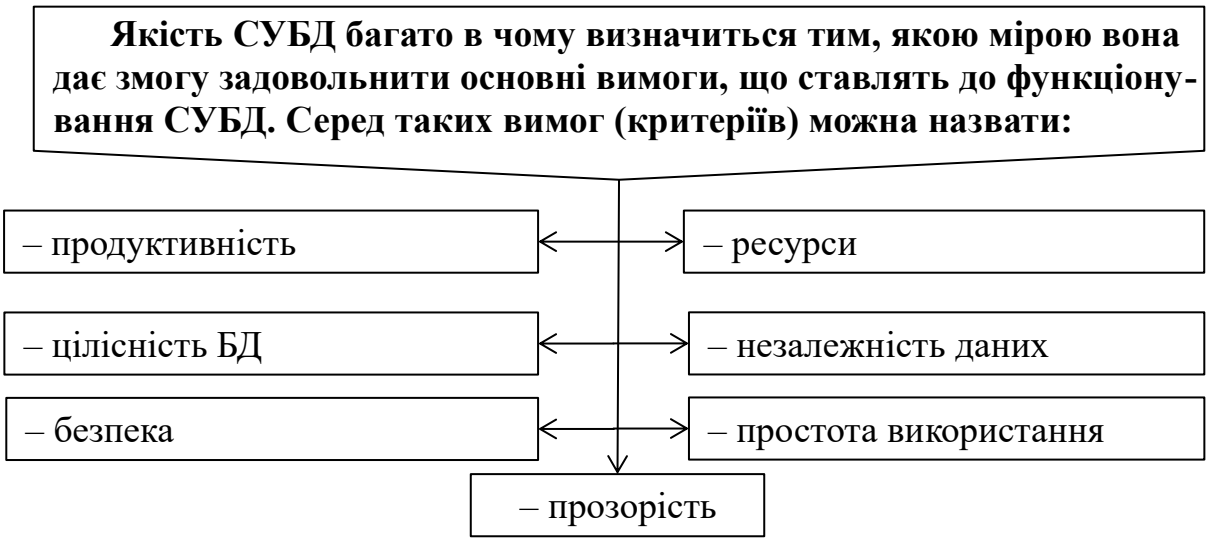


Рис. 3.6. Склад характеристик, що визначають якість СУБД

Дамо коротку характеристику кожного з цих критеріїв.

Продуктивність кількісно можна оцінити як ***час реакції системи на запит*** користувача. За інших рівних умов продуктивність забезпечується здатністю СУБД оптимізувати процес пошуку і оновлення даних у файлах БД, знаходити раціональний план обчислення запиту.

Для *ефективної роботи* СУБД, як правило, потрібні значні обчислювальні **ресурси**. У цьому випадку особливо критичними виявляються об’єм оперативної пам'яті, пропускна здатність каналів введення/виведення і каналів обміну, продуктивність процесора, об’єм зовнішньої пам'яті.

З метою *раціонального розподілу ресурсів* СУБД має бути оснащена засобами моніторингу і статистики.

Цілісність БД є основною умовою безвідмовного функціонування СУБД. Практично всі сучасні СУБД оснащені потужним арсеналом засобів підтримки цілісності і парирування аварійних ситуацій.

Тут головна роль відводиться засобам автоматичного копіювання і відновлення БД.

Незалежність даних розуміється як можливість внесення змін у структуру БД, що зберігається без змін у «логічній» структурі даних і прикладних програмах (запитах). На практиці неможливо досягти абсолютної незалежності даних.

Безпека БД розуміється як її захищеність від несанкціонованого використання. Кожен користувач (прикладна програма) у СУБД має певні права доступу до тих чи інших фрагментів БД.

Простота використання – неформальний критерій, що виявляється у вимогах до кваліфікації обслуговуючого персоналу, зручність роботи користувачів різних категорій.

Вимога **прозорості даних** полягає у наданні користувачам можливості звернення до розподілених в мережі даних без зміни технології своєї роботи. СУБД має по можливості «приховувати» від користувача додаткові технологічні деталі операцій звернення до віддалених даних.

Контрольні запитання до розділу 3

1. Пояснити місце систем керування базами даних у системах обробки інформації.
2. Навести склад забезпечувальних функцій СУБД.
3. Для чого застосовують сучасні БД?
4. Що визначає модель даних?
5. Навести функціональні характеристики систем керування базами даних.
6. У чому полягає підтримка моделі даних засобами СУБД?
7. У чому полягає властивість масштабованості і здатності до перенесення СУБД?
8. Визначити основні вимоги до систем керування базами даних.

Розділ 4. Склад і зміст функції систем керування базами даних

На рис. 4.1 наведено склад основних функції систем керування базами даних.

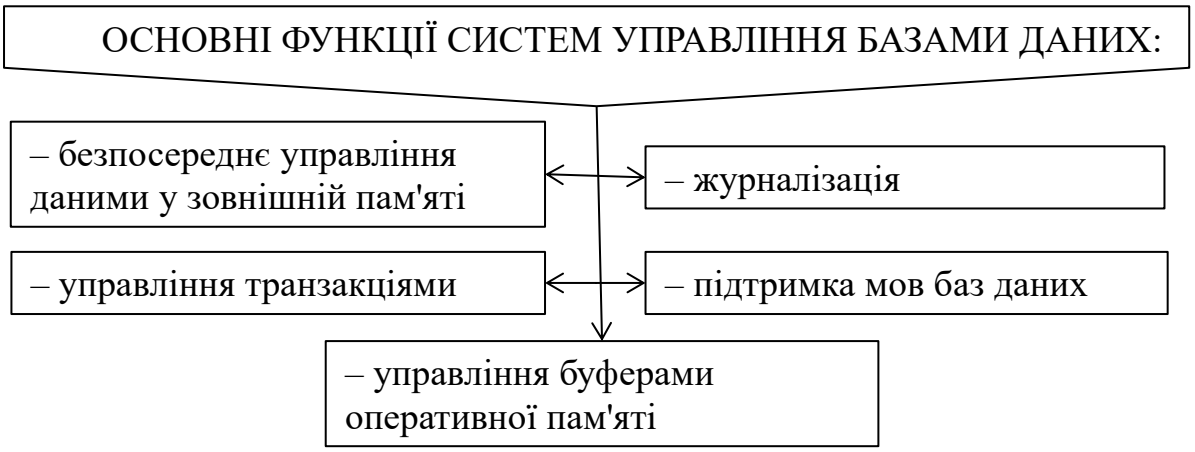


Рис. 4.1. Склад основних функції систем керування базами даних

4.1. Керування даними в зовнішній пам'яті

Безпосереднє керування даними в зовнішній пам'яті [2] включає забезпечення необхідних структур зовнішньої пам'яті як для зберігання даних, що безпосередньо входять до БД, так і для службових цілей.

У розвинених СУБД користувачі в будь-якому разі не зобов'язані знати, чи використовує СУБД файлову систему, і якщо використовує, то як організовані файли. Зокрема СУБД підтримує власну систему іменування об'єктів БД.

4.2. Керування буферами оперативної пам'яті

Керування буферами оперативної пам'яті [2]. СУБД зазвичай працюють з БД значного розміру; принаймні цей розмір зазвичай істотно більше доступного об’єму оперативної пам'яті.

Зрозуміло, що якщо при зверненні до будь-якого елемента даних буде здійснюватися обмін із зовнішньою пам'яттю, то вся

система буде працювати зі швидкістю пристрою зовнішньої пам'яті. Практично єдиним способом реального збільшення цієї швидкості є буферизація даних в оперативній пам'яті. При цьому в розвинених СУБД підтримується власний набір буферів оперативної пам'яті з власною дисципліною заміни буферів.

4.3. Керування транзакціями

Керування транзакціями [2]. Транзакція – це послідовність операцій над БД, розглянутих СУБД як єдине ціле. Або транзакція успішно виконується, і СУБД фіксує зміни БД, вироблені цією транзакцією, у зовнішній пам'яті, або жодна з цих змін ніяк не позначається на стані БД.

Поняття транзакції необхідне для *підтримки логічної цілісності БД*. Підтримка механізму транзакцій є обов'язковою умовою навіть одного користувача СУБД. Але поняття транзакції набагато важливіше в багатокористувацьких СУБД.

З керуванням транзакціями в багатокористувацькій СУБД пов'язані важливі поняття *серіалізації транзакцій* і *серіального плану* виконання суміші транзакцій.

При використанні будь-якого алгоритму серіалізації можливі ситуації *конфліктів* між двома або більше транзакціями з доступу до об'єктів БД. У цьому випадку для підтримки серіалізації необхідно виконати *повернення однієї або більше транзакцій*.

4.4. Журналізація

Журналізація [2]. Однією з основних вимог до СУБД є *надійність* зберігання даних у зовнішній пам'яті. Під надійністю зберігання розуміється те, що СУБД має бути здатна *відновити останній узгоджений стан* перебування БД після будь-якого апаратного або програмного збою.

Зазвичай розглядаються два можливі види апаратних збоїв:

– *м'які збої*, що можна трактувати як раптову зупинку роботи комп'ютера (наприклад аварійне вимкнення живлення);

– *жорсткі збої*, що характеризуються втратою інформації на носіях зовнішньої пам'яті.

Зрозуміло, що в будь-якому випадку для відновлення БД слід мати у своєму розпорядженні деяку *додаткову інформацію*. Найбільш поширеним методом підтримання такої надмірної інформації є ведення *журналу* змін БД.

Журнал – це особлива частина БД, не доступна користувачам СУБД і підтримувана з особливою ретельністю, до якуої надходять записи про всі зміни основної частини БД. У різних СУБД зміни БД журналізуются на різних рівнях:

– запис у журналі відповідає деякій логічній операції зміни БД;

– мінімальна внутрішня операція модифікації сторінки зовнішньої пам'яті;

– у деяких системах одночасно використовуються обидва підходи.

У всіх випадках дотримуються стратегії «попереджувального» запису до журналу (так званого протоколу Write Ahead Log – WAL).

При **м'якому** збої в зовнішній пам'яті основної частини БД можуть перебувати об'єкти, модифіковані транзакціями, що не закінчилися до моменту збою, і можуть бути відсутніми об'єкти, модифіковані транзакціями, які до моменту збою успішно завершилися. При дотриманні протоколу WAL у зовнішній пам'яті журналу мають бути гарантовано перебувати записи, що належать до операцій модифікації обох видів об'єктів.

Для відновлення БД після **жорсткого** збою використовують журнал і архівну копію БД. Звичайно, для нормального відновлення БД після жорсткого збою необхідно, щоб журнал не пропав. Як вже зазначалося, до збереження журналу в зовнішній пам'яті в СУБД висувають особливо підвищені вимоги.

4.5. Підтримка мов баз даних

На рис. 4.2 наведено характеристики мов, застосовувані в базах даних.

У сучасних СУБД зазвичай підтримується єдина інтегрована мова, що містить усі необхідні засоби для роботи з

БД, починаючи від її створення, і забезпечує базовий призначений для користувача інтерфейс з базами даних. Стандартною мовою найбільш поширених у наш час реляційних СУБД є мова SQL (Structured Query Language).

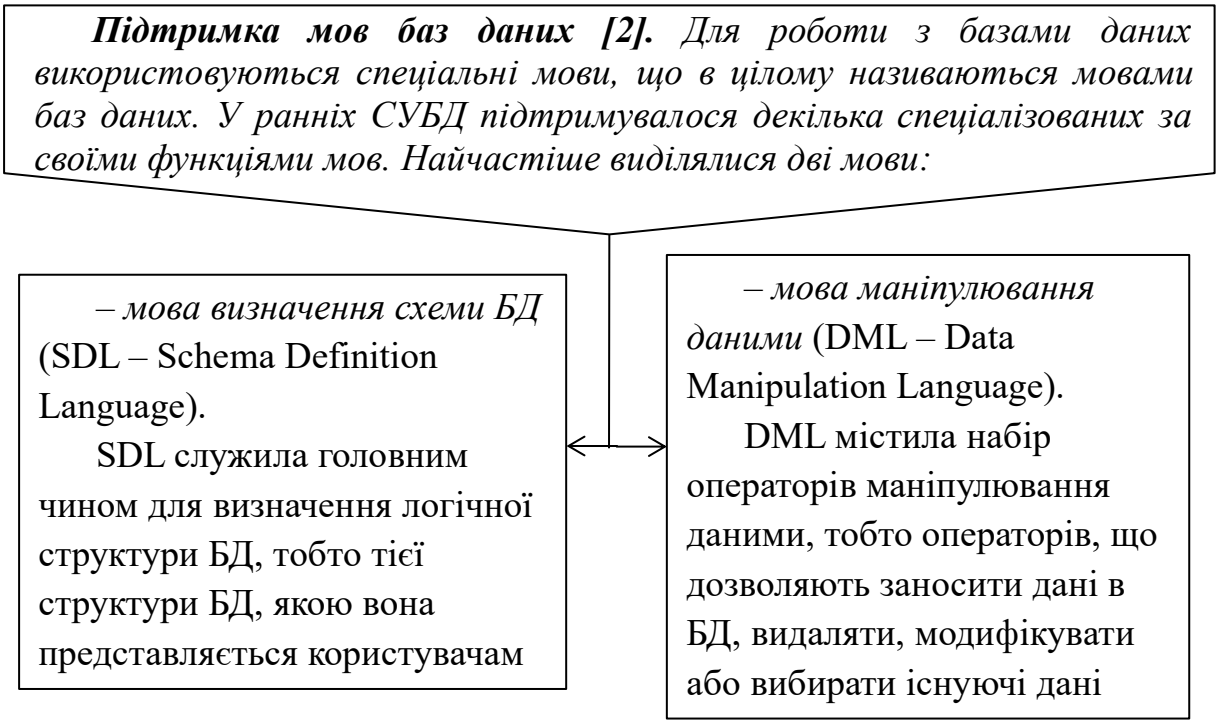


Рис. 4.2. Характеристики мов, застосовувані в базах даних

Спеціальні оператори мови SQL дають змогу визначати так звані *подання БД*, що фактично є збереженими в БД запитами з іменованими стовпцями.

Для користувача подання є такою самою таблицею, як будь-яка базова таблиця, що зберігається в БД, але за допомогою цього можна обмежити або, навпаки, розширити видимість БД для конкретного користувача. Підтримка подань проводиться також на мовному рівні.

Авторизація доступу до об'єктів БД формується також на основі спеціального набору операторів SQL. Ідея полягає в тому, що для виконання операторів SQL різного виду користувач має володіти різними повноваженнями. Користувач, який створив таблицю БД, володіє повним набором повноважень для роботи з цією таблицею.

Контрольні запитання до розділу 4

1. У чому полягає зміст безпосереднього керування даними в зовнішній пам'яті?

2. Для яких даних застосовується безпосереднє керування даними?

3. Навести зміст керування буферами оперативної пам'яті.

4. Навести спосіб реального збільшення швидкості обробки даних.

5. На чому ґрунтується напрям СУБД, орієнтований на постійну присутність в оперативній пам'яті всієї БД?

6. У чому полягає зміст керування транзакціями?

7. За яких умов починається кожна транзакція?

8. Що розуміють під серіалізацією транзакцій?

9. Для чого застосовується журналізація?

Розділ 5. Елементи теорії реляційних моделей даних

5.1. Основи реляційних баз даних

Ми приступаємо до вивчення реляційних баз даних і систем керування базами даних. Цей підхід є найбільш поширеним у наш час, хоча разом з загальновизнаними перевагами має і ряд недоліків [4–6].

На рис. 5.1 наведено склад і зміст переваг і недоліків реляційних баз даних.

Рис. 5.1. Склад і зміст переваг і недоліків реляційних баз даних

Зараз основним предметом критики реляційних СУБД є не їхня *недостатня ефективність*, а притаманна цим системам деяка обмеженість при використанні в так званих нетрадиційних галузях, у яких потрібні гранично складні структури даних.

Ще одним недоліком, часто зазначуваним для реляційних баз даних, є неможливість адекватного відображення *семантики* предметної галузі.

Сучасні дослідження в галузі *постреляційних систем* переважно присвячені саме усуненню цих недоліків.

5.2. Склад і зміст понять реляційних баз даних

Основними поняттями реляційних баз даних є тип даних, домен, атрибут, кортеж, первинний ключ і відношення.

Для початку покажемо зміст цих понять на прикладі відношення *Співробітники*, що містить інформацію про співробітників деякої організації (рис. 5.2).

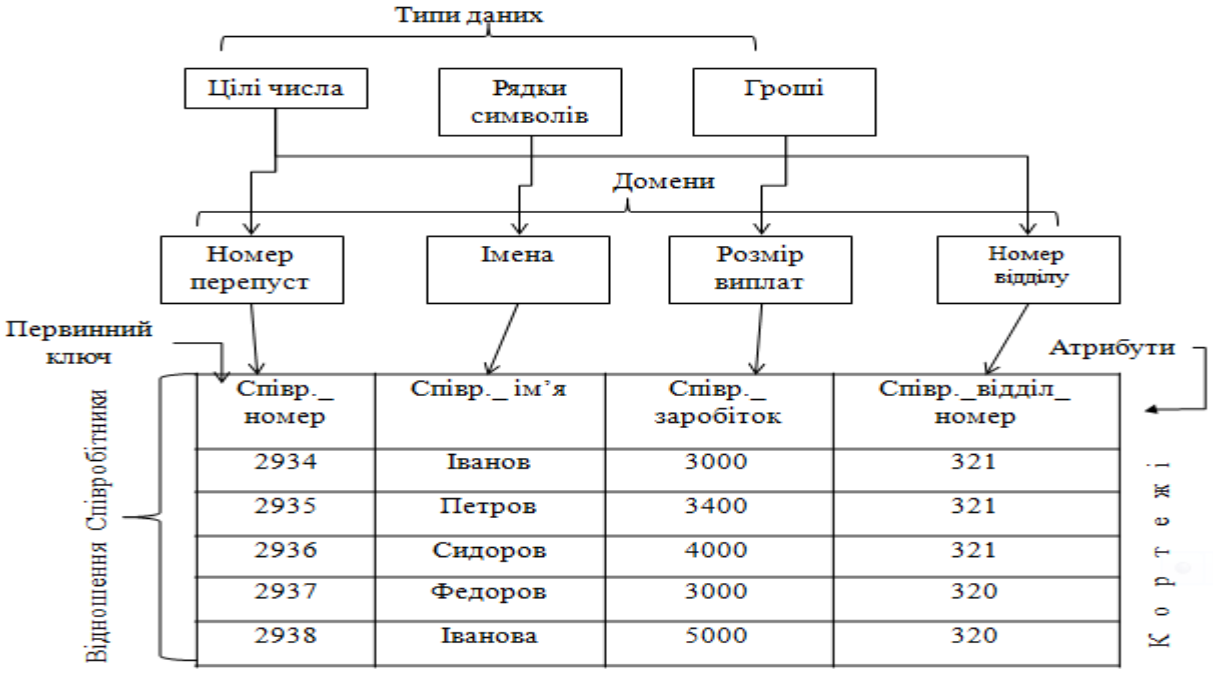


Рис. 5.2. Основні поняття реляційних баз даних

На рис. 5.3 наведено зміст типів даних, застосовуваних у реляційних базах даних.

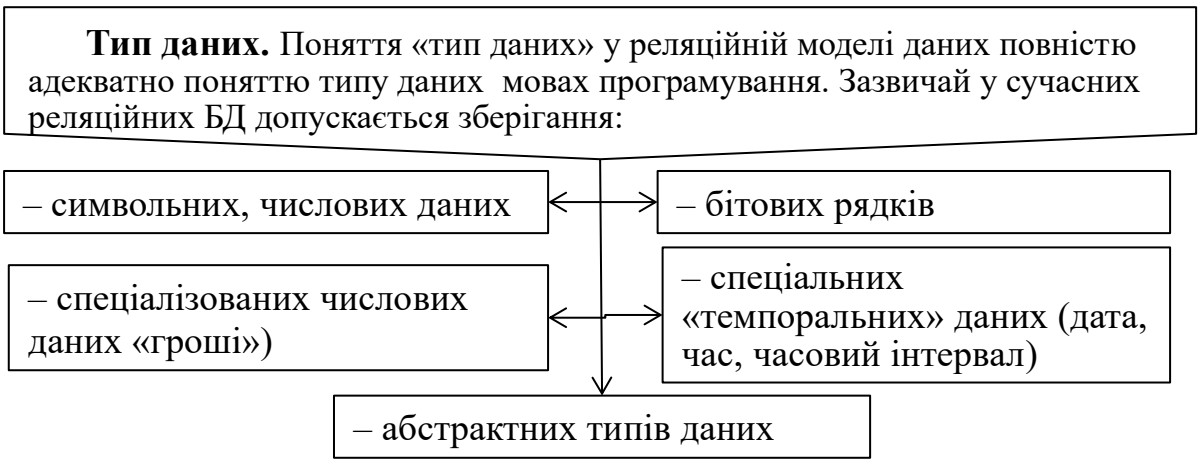


Рис. 5.3. Зміст типів даних реляційних баз даних

Домен. Поняття домену більш специфічно для баз даних, хоча і має деякі аналогії з підтипами в деяких мовах програмування. У найзагальнішому вигляді *домен визначається задаванням деякого базового типу даних*, до якого належать елементи домену, і довільного логічного виразу, застосовуваного до елемента типу даних. Якщо обчислення цього логічного виразу дає результат «істина», то елемент даних є елементом домену.

Найбільш правильним інтуїтивним трактуванням поняття домену є розуміння домену як *допустимої потенційної безлічі значень певного типу*. Слід зазначити також семантичне навантаження поняття домену: дані вважаються порівнянними тільки в тому випадку, коли вони належать до одного домену.

Схема відношення, схема бази даних. Схема відношення – це іменована безліч пар {ім'я атрибута, ім'я домену (або типу, якщо поняття домену не підтримується)}. Ступінь, або «арність» схеми відношення − потужність цієї множини. Якщо всі атрибути одного відношення визначені на різних доменах, логічно використовувати для іменування атрибутів імена відповідних доменів (не забуваючи, звичайно, про те, що це є лише зручним способом іменування і не усуває відмінності між поняттями домену і атрибута). **Схема БД** (у структурному сенсі) – це набір іменованих схем відносин. Важливим поняттям для реляційних баз даних є поняття «кортеж» (рис. 5.4).

Рис. 5.4. Визначення змісту поняття «кортеж»

Відношення – це *безліч кортежів*, що відповідають *одній схемі відношення*. Іноді, щоб не плутатися, говорять «*відношення-схема*» і «*відношення-екземпляр*», іноді схему відношення називають *заголовком відношення*, а відношення як набір кортежів – *тілом відносини*.

Насправді поняття схеми відношення найближче до поняття структурного типу даних у мовах програмування. Було б цілком логічно дозволяти окремо визначати схему відношення, а потім одне або кілька відношень з певною схемою. Однак у реляційних базах даних це не прийнято. Ім'я схеми відношення в таких базах даних завжди збігається з ім'ям відповідного відношення-екземпляра.

Звичайним практичним поданням відношення є таблиця, заголовком якої є схема відношення, а рядками – кортежі відношень-екземплярів; у цьому випадку імена атрибутів іменують стовпці цієї таблиці.

Контрольні запитання до розділу 5

1. Навести переваги реляційних баз даних.
2. Навести недоліки реляційних баз даних.
3. Навести визначення поняття «дані».
4. Навести визначення поняття «домен».
5. Навести визначення поняття «атрибут».
6. Навести визначення поняття «кортеж».
7. Навести визначення поняття «первинний ключ».
8. Навести визначення поняття «вторинний ключ».
9. Навести визначення поняття «відношення».

Розділ 6. Елементи реляційної алгебри

Доступ до реляційних даних здійснюється за допомогою реляційної алгебри або еквівалентного їй реляційного числення.

У реалізаціях конкретних реляційних СУБД не використовується в чистому вигляді ані реляційна алгебра, ані реляційне числення.

Фактичним стандартом доступу до реляційних даних стала мова SQL (Structured Query Language).

Мова SQL являє собою *суміш операторів реляційної алгебри і виразів реляційного числення,* що використовує синтаксис, близький до фраз англійської мови і розширений додатковими можливостями, відсутніми в реляційній алгебрі та реляційному численні.

Взагалі мова доступу до даних називається *реляційно повною*, якщо вона за виразною силою *не поступається реляційній алгебрі* (або, що те саме, *реляційному численню*). Саме такою і є мова SQL.

У цьому розділі будуть розглянуті основні елементи реляційної алгебри.

Для поглибленого вивчення теорії реляційної алгебри рекомендуються джерела [4–6].

6.1. Аксіома замкненості реляційної алгебри

Реляційна алгебра являє собою набір операторів, що використовують відношення як аргументи і повертають відношення як результат. Отже, реляційний оператор виглядає як функція з відношеннями як аргументами [4–6]

$$R = f(R_1, R_2, \ldots, R_n) \quad . \tag{6.1}$$

Реляційна алгебра є замкненою, тому що як аргументи в реляційних операторах можна підставляти інші реляційні оператори, прийнятні за типом [4–6],

$$R = f(f_1(R_{11}, R_{12}, \ldots), f_2(R_{21}, R_{22}, \ldots), \ldots). \tag{6.2}$$

Отже, у реляційних виразах можна використовувати вкладені вирази як завгодно складної структури.

Кожне відношення повинно мати *унікальне ім'я в межах бази даних*.

Ім'я відношення, отриманого в результаті виконання *реляційної операції*, визначається в *лівій частині рівності*.

Однак *можна не вимагати наявності імен від відношень*, отриманих у результаті реляційних виразів, якщо ці відношення підставляються як аргументи в інші реляційні вирази.

Такі відношення будемо називати *неіменованими відношеннями*.

Неіменовані відношення *реально не існують у базі даних*, а тільки обчислюються в момент обчислення значення реляційного оператора.

На рис. 6.1 наведено склад і зміст реляційних операторів.

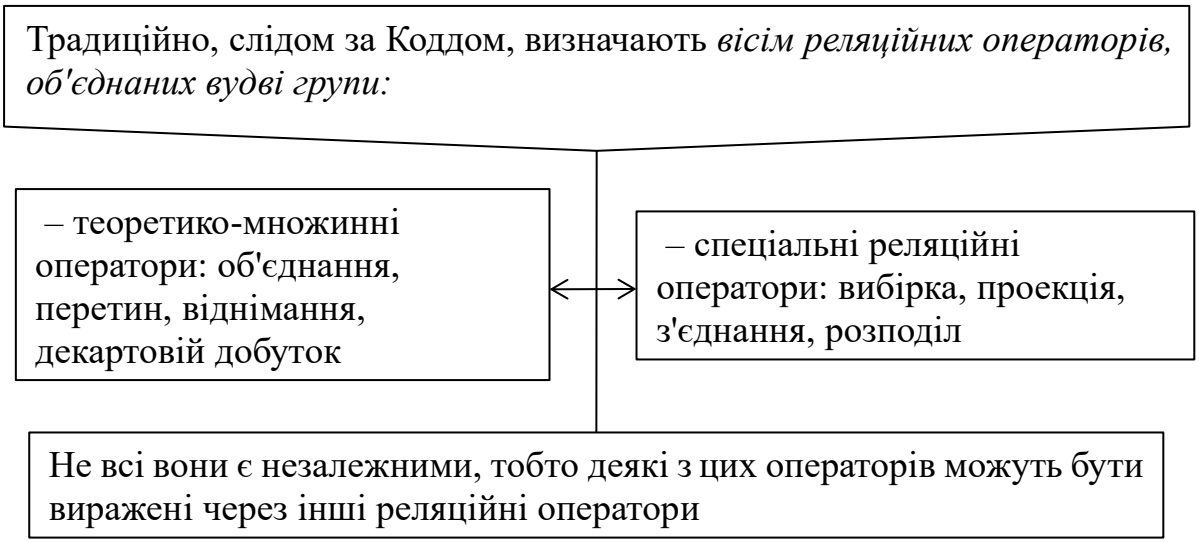


Рис. 6.1. Склад і зміст реляційних операторів

6.2. Властивості відношень, сумісних за типом

Деякі реляційні оператори (наприклад об'єднання) вимагають від відношень однакових заголовків. Дійсно, відношення складаються з заголовка і тіла.

На рис. 6.2 наведено умови реалізації операції об’єднання двох відношень.

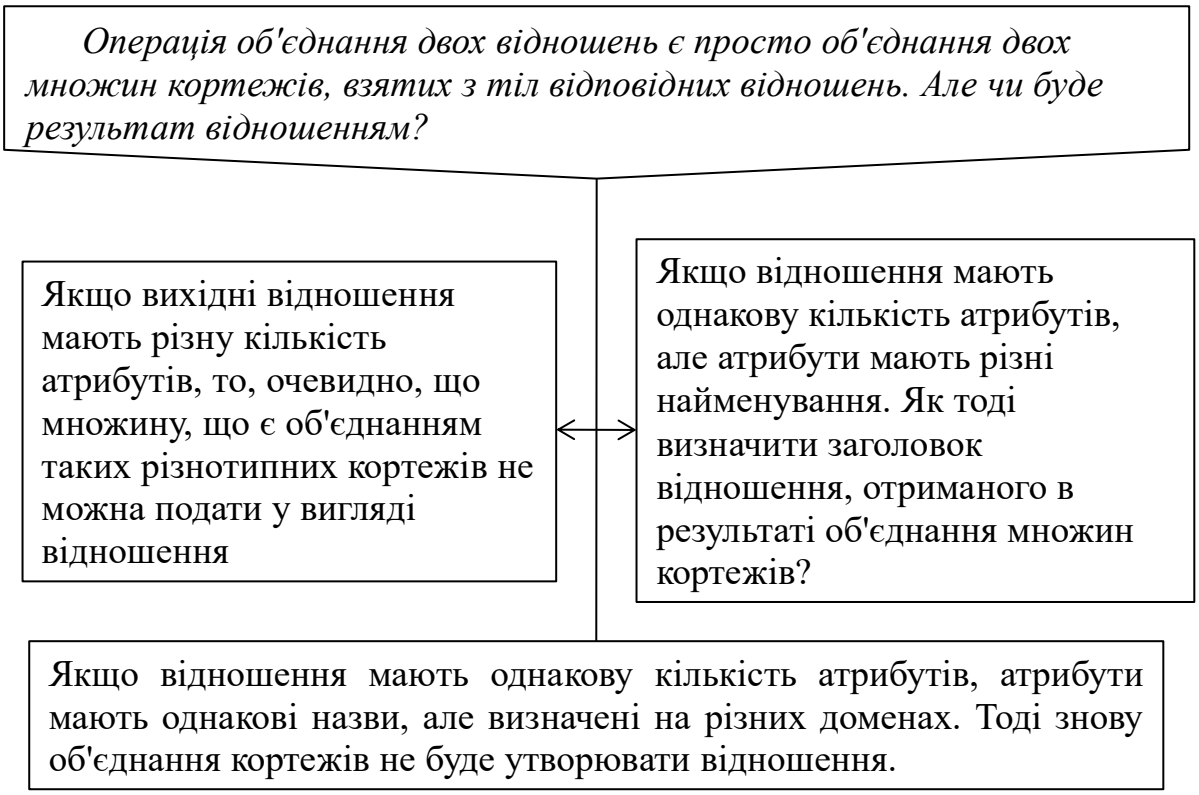


Рис. 6.2. Умови реалізації операції об’єднання двох відношень

На рис. 6.3 наведено визначення поняття «відношення, сумісного за типом».

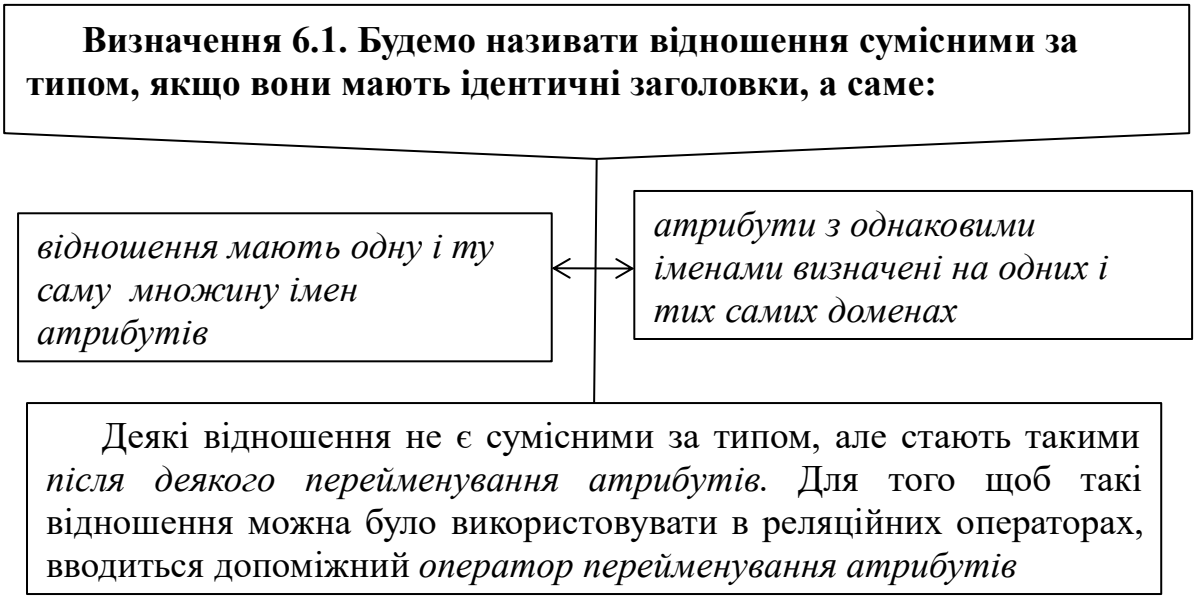


Рис. 6.3. Визначення змісту поняття «відношення, сумісного за типом»

6.3. Властивості оператора перейменування атрибутів

Оператор перейменування атрибутів має такий синтаксис [4–6]:

$$R\,\mathrm{RENAME}\,Atr_1, Atr_2, \ldots \mathrm{AS}\,NewAtr_1, NewAtr_2, \ldots, \qquad (6.3)$$

Де R – відношення;

$Atr_1, Atr_2, \ldots$ – вихідні імена атрибутів;

$NewAtr_1, NewAtr_2, \ldots$ – нові імена атрибутів.

У результаті застосування оператора перейменування атрибутів отримуємо *нове відношення зі зміненими іменами атрибутів*.

6.4. Властивості оператора об’єднання відношень

Визначення 6.2. Об’єднанням двох сумісних за типом відношень *A* і *B* називається відношення з тим самим заголовком, що і у відношень *A* і *B*, і тілом, що складається з кортежів, які належать або *A*, або *B*, або обом відношенням

Синтаксис операції об’єднання [4–6]

$$A\ \mathrm{UNION}\ B\ . \qquad (6.4)$$

Зауваження. Об'єднання, як і будь-яке відношення, не може містити однакових кортежів. Тому, якщо деякий кортеж входить і у відношення *A* і відношення *B*, то в об'єднання він входить *один* раз

Приклад 6.1. Нехай дано два відношення *A* і *B* з інформацією про співробітників:

– відношення *A*

Табельний номер	Прізвище	Зарплата
1	Іванов	1000
2	Петров	2000
3	Сидоров	3000

– відношення *B*

Табельний номер	Прізвище	Зарплата
1	Іванов	1000
2	Пушников	2500
4	Сидоров	3000

– *об'єднання* відношень *A* і *B*

Табельний номер	Прізвище	Зарплата
1	Іванов	1000
2	Петров	2000
3	Сидоров	3000
2	Пушников	2500
4	Сидоров	3000

Зауваження. Як видно з наведеного прикладу, потенційні ключі, які були у відношеннях *A* і *B*, не успадковуються об'єднанням цих відношень.

6.5. Властивості оператора перетину відношень

Визначення 6.3. Перетином двох сумісних за типом відношень *A* і *B* називається відношення з тим самим заголовком, що і у відношень *A* і *B*, і тілом, що складається з кортежів, які належать одночасно обом відношенням *A* і *B*

Синтаксис операції перетину [4–6]

A INTERSECT *B*.

Приклад 6.2. Для тих самих відношень *A* і *B*, що і в попередньому прикладі, перетин має вигляд відношення *A* INTERSECT *B*.

Табельний номер	Прізвище	Зарплата
1	Іванов	1000

Зауваження. Здавалося б, що, на відміну від операції об'єднання, потенційні ключі могли б успадковуватися перетином відношень. Однак це не так. Взагалі жодні реляційні оператори не залишають поза передачею результуючому відношенню ніяких даних про потенційні ключі. Як причину цього можна було б навести тривіальне міркування, що так є простіше і симетричніше – усі оператори влаштовані однаково.

Насправді причина глибша і полягає в тому, що потенційний ключ – семантичне поняття, що відображує розрізнення об'єктів предметної галузі.

Наявність потенційних ключів не виводиться зі структури відношення, а явно задається для кожного відношення виходячи з його змісту.

Реляційні ж оператори є формальними операціями над відношеннями і виконуються однаково незалежно від змісту даних, що містяться у відношеннях.

Тому реляційні оператори нічого не можуть «знати» про сенс даних. Трактування результату реляційних операцій – справа користувача.

6.6. Властивості оператора віднімання відношень

Визначення 6.4. ***Відніманням двох сумісних за типом відношень A і B*** **називається відношення з тим самим заголовком, що і у відношень *A* і *B*, і тілом, яке складається з кортежів, що належать відношенню *A* і не належать відношенню *B***

Синтаксис операції віднімання [4–6]

$$A \text{ MINUS } B \,. \tag{6.5}$$

Приклад 6.3. Для тих самих відношень *A* і *B*, що і в попередньому прикладі, віднімання має вигляд відношення *A* MINUS *B*.

Табельний номер	Прізвище	Зарплата
2	Петров	2000
3	Сидоров	3000

6.7. Властивості оператора декартового добутку відношень

Визначення 6.5. Декартовим добутком двох відношень [4–6]

$$A(A_1, A_2, \ldots, A_n) \text{ і } B(B_1, B_2, \ldots, B_m)$$

називається відношення, заголовок якого є зчепленням заголовків відношень *A* і *B*

$$(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m),$$

а тіло складається з кортежів, що є зчепленням таких кортежів відношень A і B, що

$$(a_1, a_2, \ldots, a_n) \in A, (b_1, b_2, \ldots b_m) \in B \,.$$

Синтаксис операції декартового добутку [4–6]

$$A \text{ TIMES } B. \tag{6.6}$$

Зауваження. Потужність добутку дорівнює добутку потужностей відношень *A* і *B*, тому що кожен кортеж відношення *A* з'єднується з кожним кортежем відношення *B*. Якщо у відношеннях *A* і *B* є атрибути з однаковими найменуваннями, то перед виконанням операції декартового добутку такі атрибути необхідно перейменувати. Перемножати можна будь-які два відношення, сумісність за типом при цьому не потрібна

Приклад 6.4. Нехай дано два відношення *A* і *B* з інформацією про постачальників і деталі: відношення *A* (Постачальники), відношення *B* (Деталі)

Номер постачальника	Постачальник
1	Іванов
2	Петров
3	Сидоров

Відношення *A* (Постачальники)

Номер деталі	Найменування деталі
1	Болт
2	Гайка
3	Гвинт

Відношення *B* (Деталі)

Декартовий добуток відношень *A* і *B* матиме вигляд відношення *A X B*

Номер постачальника	Постачальник	Номер деталі	Найменування деталі
1	Іванов	1	Болт
1	Іванов	2	Гайка
1	Іванов	3	Гвинт
2	Петров	1	Болт
2	Петров	2	Гайка
2	Петров	3	Гвинт
3	Сидоров	1	Болт
3	Сидоров	2	Гайка
3	Сидоров	3	Гвинт

Зауваження. Сама по собі операція декартового добутку не дуже важлива, тому що вона не дає ніякої нової інформації у порівнянні з вихідними відношеннями. Для реальних запитів ця операція майже ніколи не використовується. *Однак операція декартового добутку важлива для виконання спеціальних реляційних операцій*, про які йтиметься нижче

Контрольні запитання до розділу 6

1. У чому полягає замкненість реляційної алгебри?
2. Навести визначення відношення, сумісного за типом.
3. Навести визначення оператора перейменування атрибутів.
4. Навести визначення операції об’єднання відношень.
5. Навести визначення оператора перетину відношень.
6. Навести визначення оператора віднімання відношень.
7. Навести визначення оператора декартового добутку відношень.
8. До якої групи належать оператори об'єднання, перетину, віднімання, декартового добутку?
9. До якої групи належать оператори вибірки, проєкції, з'єднання, розподілу?

Розділ 7. Життєвий цикл розроблення бази даних

7.1. Елементи життєвого циклу бази даних [2]

На рис. 7.1 наведено склад чинників, за допомою яких визначають ефективність роботи інформаційної системи.

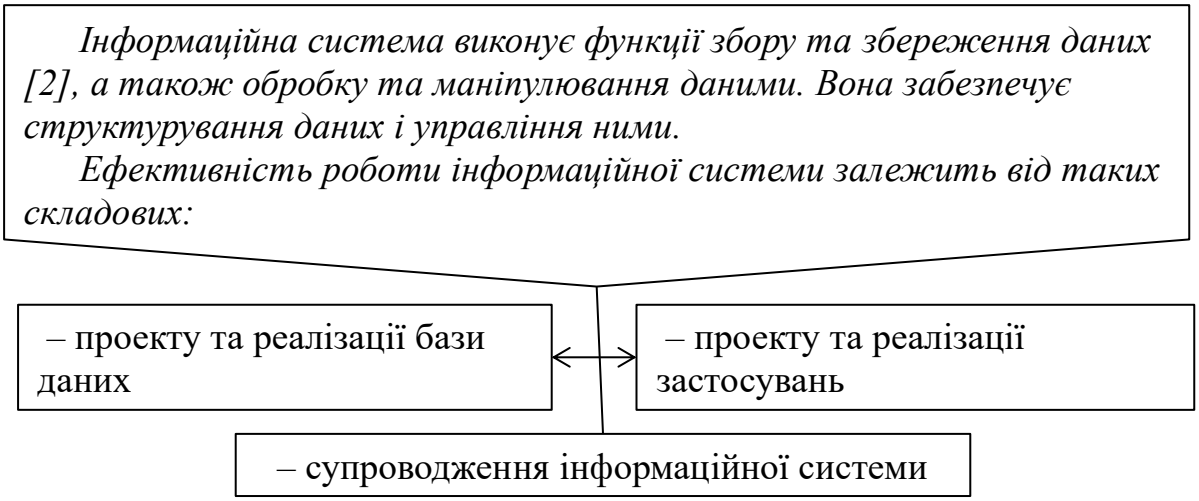


Рис. 7.1. Склад чинників, що визначають ефективність роботи інформаційної системи

База даних є *фундаментальним компонентом інформаційної системи*, і проєктування БД виконується *в рамках проєктування інформаційної системи.*

На рис. 7.2 наведено склад етапів життєвого циклу інформаційної системи.

Рис. 7.2. Етапи життєвого циклу інформаційної системи

Ці етапи не є строго послідовними і передбачають повернення на попередні етапи за допомогою зворотних зв'язків. БД як частина інформаційної системи має свій життєвий цикл (рис. 7.3) [2].

Рис. 7.3. Етапи життєвого циклу бази даних

Конкретне наповнення кожного етапу значною мірою залежить від складності продукту, що розробляється. Для невеликих інформаційних систем кількість етапів може бути зменшена. Розглянемо більш детально зміст кожного етапу [2].

7.2. Елементи розроблення стратегічного плану

На рис. 7.4 наведено склад і зміст етапів розроблення стратегічного плану.

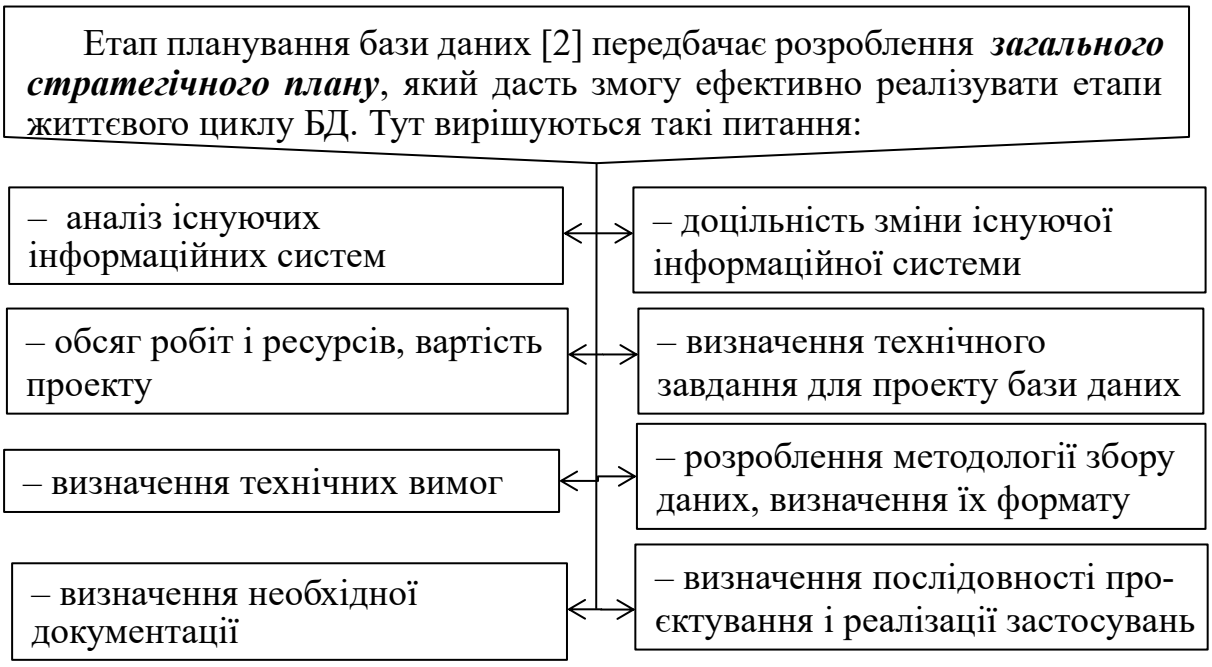


Рис. 7.4. Склад і зміст етапів розроблення стратегічного плану

7.3. Аналіз вимог до елементів бази даних

На рис. 7.5 наведено склад і зміст завдань для етапу аналізу вимог до баз даних [2].

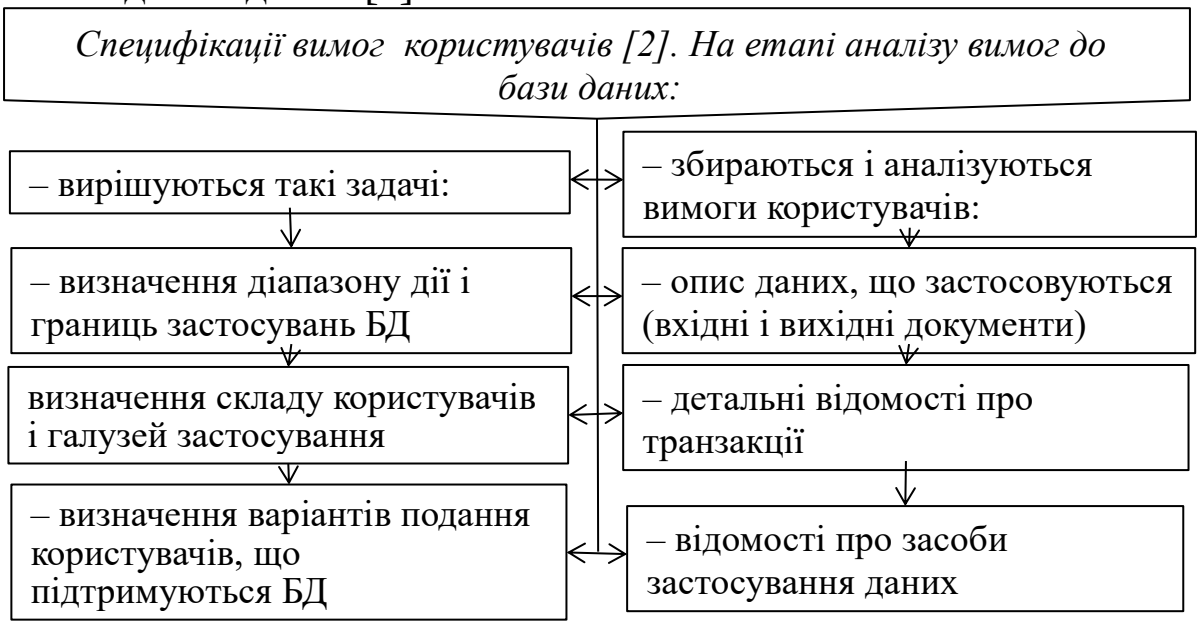


Рис. 7.5. Склад і зміст завдань, вирішуваних на етапі аналізу вимог до баз даних

7.4. Елементи процесу проєктування бази даних

На рис. 7.6 наведено склад елементів процесу проєктування бази даних.

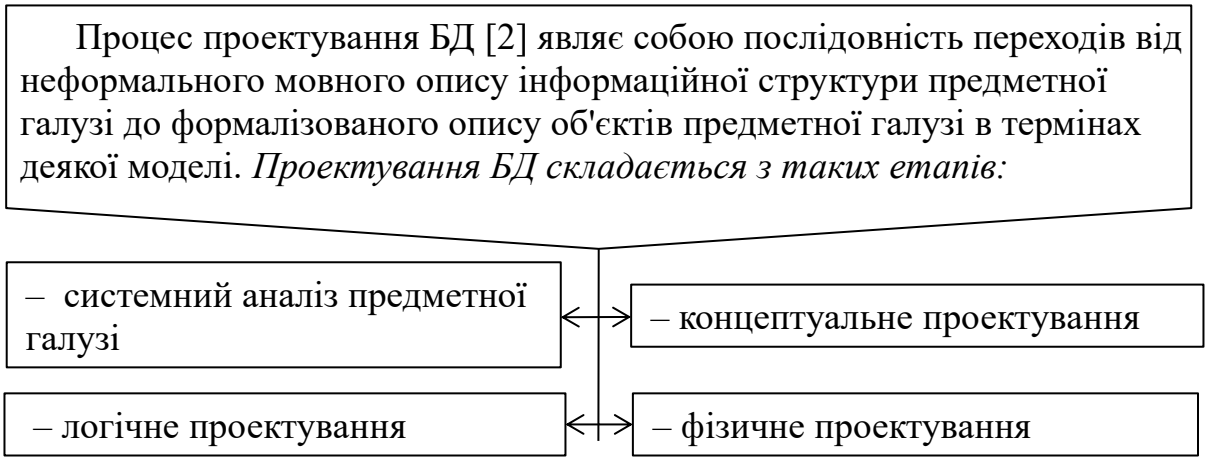


Рис. 7.6. Склад елементів процесу проєктування бази даних

Важливим елементом процесу проєктування бази даних є системний аналіз предметної залузі. Зміст завдань, вирішуваних у процесі системного аналізу, наведено на рис. 7.7.

Рис. 7.7. Завдання, вирішувані у процесі системного аналізу

Для визначення складу і структури предметної галузі застосовуються або функціональний, або предметний підходи.

Функціональний підхід застосовує рух «від завдань» і використовується у тих випадках, коли *заздалегідь відомі функції*

майбутніх користувачів БД, а також *відомі всі завдання*, для інформаційних потреб яких створюються БД.

У цьому випадку на основі виробничих документів, опитувань замовників можна *чітко визначити мінімальний набір об'єктів предметної галузі та їхній взаємозв'язок.*

Предметний підхід застосовується в тому випадку, коли інформаційні потреби майбутніх користувачів *чітко не визначені*.

До опису предметної галузі включаються об'єкти і зв'язки, що є найбільш характерними та найбільш суттєвими для неї.

БД називається *предметною* і може використовуватися для *вирішення завдань, заздалегідь не визначені.*

У загальному випадку існує два підходи до проєктування БД: низхідне проєктування і висхідне проєктування (рис. 7.8) [2].

Рис. 7.8. Схема підходів до проєктування бази даних

Низхідне проєктування починається з визначення наборів даних, потім з елементів даних для кожного з таких наборів. Цей процес включає ідентифікацію різних типів сутностей і визначення атрибутів кожної сутності.

Низхідне проєктування включає операції *декомпозиції,* що передбачає заміну вихідної множини відношень, які входять до схеми БД, іншою множиною відношень, що є проєкціями вихідних відношень.

Цей підхід рекомендується застосовувати в тих випадках, коли кількість, різноманітність і складність сутностей, зв'язків і транзакцій значна за розмірами. Найбільш поширеними

моделями для цього проєктування є моделі «сутність-зв'язок» (ER-моделі).

Висхідне проєктування починається з виявлення елементів даних, що потім групуються в набори даних. Спочатку визначаються атрибути, які потім об'єднуються в сутності.

Висхідне проєктування включає операції *синтезу,* що передбачає компонування з заданої множини функціональних залежностей між об'єктами предметної галузі вихідних відношень схеми БД.

Цей підхід рекомендується застосовувати в тому випадку, якщо розробляється невелика БД з незначною кількістю об'єктів, атрибутів і транзакцій.

Концептуальне проєктування полягає у створенні *концептуальної моделі,* яку відображує *концептуальна схема* БД. На цьому етапі визначаються об'єкти, зв'язки між об'єктами, атрибути, ключові атрибути.

Логічне проєктування полягає у створенні *логічної моделі* на основі вибраної моделі даних. На цьому етапі необхідно вже знати, яка СУБД буде застосовуватися в системі (ієрархічна, мережева, реляційна, об'єктно-орієнтована). Для перевірки правильності логічної моделі застосовується *нормалізація.* Крім того, логічна модель перевіряється на умову забезпечення всіх *транзакцій* користувачів.

Фізичне проєктування полягає в описі засобів фізичної реалізації логічного проєкту БД. *Фізичні моделі* визначають засоби розміщення даних у середовищі зберігання і засоби доступу до цих даних, підтримуваних на фізичному рівні.

7.5. Алгоритм розроблення застосувань

На рис. 7.9 наведено склад і зміст завдань, вирішуваних на етапі розроблення застосувань.

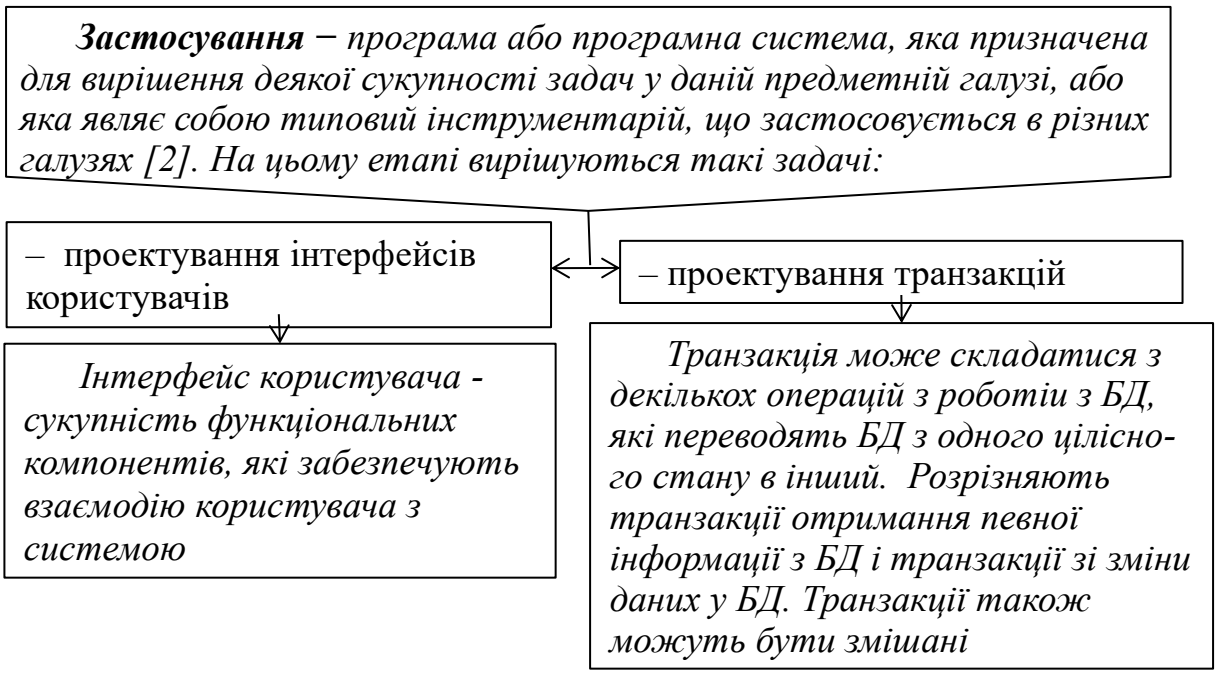


Рис. 7.9. Завдання, вирішувані на етапі розроблення застосувань

7.6. Алгоритм процесу реалізації бази даних

На рис. 7.10 наведено алгоритм процесу реалізації бази даних.

Рис. 7.10. Алгоритм процесу реалізації бази даних

7.7. Склад і зміст етапу тестування

На рис. 7.11 наведено зміст завдань, вирішуваних на етапі тестування.

Рис. 7.11. Зміст завдань, вирішуваних на етапі тестування

Важливим етапом завершення розроблення бази даних є етап покращення її роботи за підсумками тестування. На рис. 7.12 наведено зміст рекомендованих робіт.

Рис. 7.12. Зміст рекомендованих робіт з покращення бази даних

7.8. Завдання, вирішувані на етапі експлуатації

Важливим етапом життєвого циклу бази даних є етап експлуатації запровадженої бази даних. На цьому етапі вирішуються завдання, наведені на рис. 7.13.

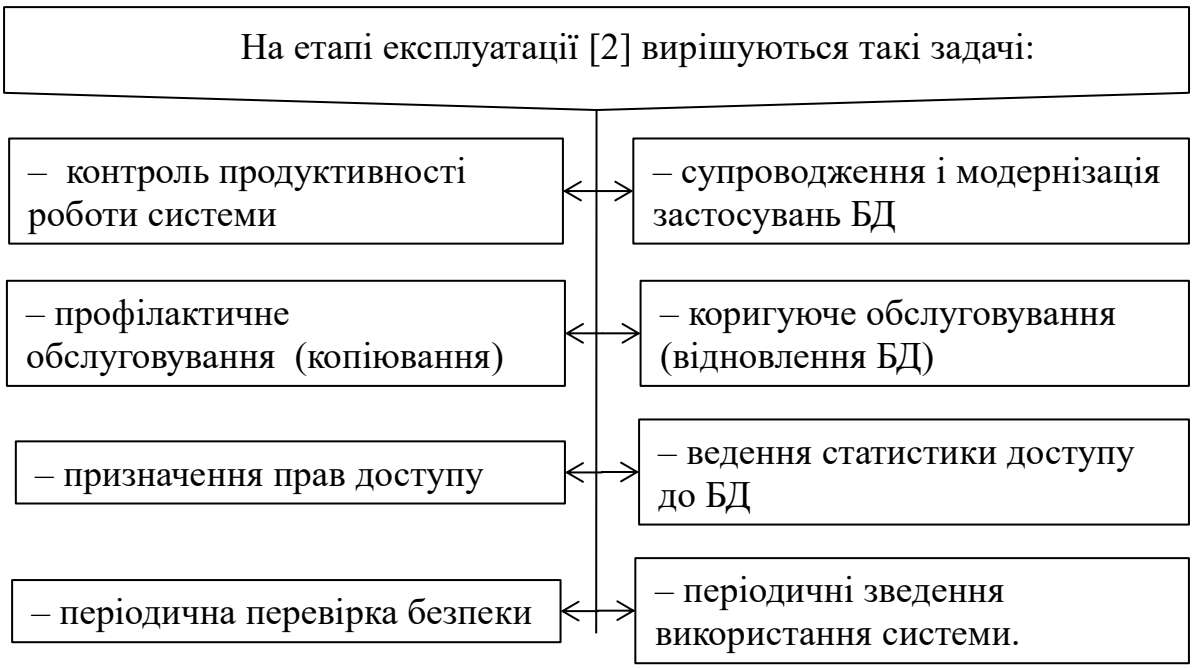


Рис. 7.13. Завдання етапу експлуатації бази даних

Контрольні запитання до розділу 7

1. Навести зміст етапів життєвого циклу бази даних.
2. Навести зміст етапу «планування бази даних».
3. Навести зміст етапу «аналіз вимог до бази даних».
4. Навести зміст етапу «проєктування бази даних».
5. Навести зміст етапу «розроблення застосувань».
6. Наведіть зміст етапу «реалізація».
7. Навести зміст етапу «тестування».
8. Навести зміст етапу «експлуатація».

Розділ 8. Розроблення концептуального проєкту бази даних

8.1. Розроблення концептуальної моделі бази даних

При розробленні концептуального проєкту бази даних [2] проєктування починається зі створення концептуальної схеми БД, в основі якої лежить *концептуальна модель даних* (дод. 1).

Концептуальна модель являє собою загальний погляд на дані. Розрізняють два головних підходи до моделювання даних при концептуальному проєктуванні:

– семантичні моделі;

– об'єктні моделі.

Семантичні моделі головну увагу приділяють структурі даних. Найбільш поширеною семантичною моделлю є модель «сутність-зв'язок» (Entity Relationship model, ER-модель).

На рис. 8.1 наведено склад елементів логічної структури даних.

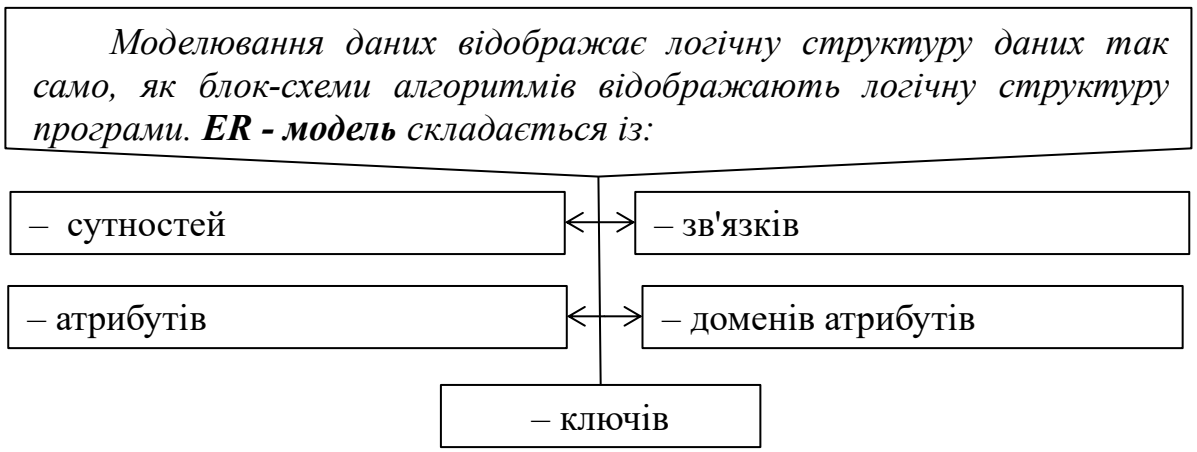


Рис. 8.1. Склад елементів логічної структури даних

Об'єктні моделі головну увагу приділяють поведінці об'єктів даних і засобам маніпуляції даними. Головне поняття таких моделей − об'єкт, тобто сутність, що має стан і поведінку. Стан об'єкта визначається сукупністю його атрибутів. Поведінка об'єкта визначається сукупністю операцій, специфікованих для нього.

8.2. Формування моделі «сутність-зв'язок»

ER-моделювання являє собою *низхідний підхід* до проєктування БД, що починається з *визначення найбільш важливих даних*, так званих *сутностей*, і *зв'язків* між даними, що мають бути подані в моделі [2].

Потім до моделі заноситься інформація про *властивості сутностей* і *зв'язків*, яка називається *атрибутами (сутностей* і *зв'язків)*, а також усі обмеження, що належать до сутностей, зв'язків і атрибутів [2].

ER-модель дає *графічне уявлення логічних об'єктів і їхніх відношень* у структурі БД. Послідовність проведення ER-моделювання показана на рис. 8.2 [2].

Рис. 8.2. Етапи побудови моделі «сутність-зв'язок»

На рис. 8.3 наведено основні типи моделей для концептуального моделювання баз даних.

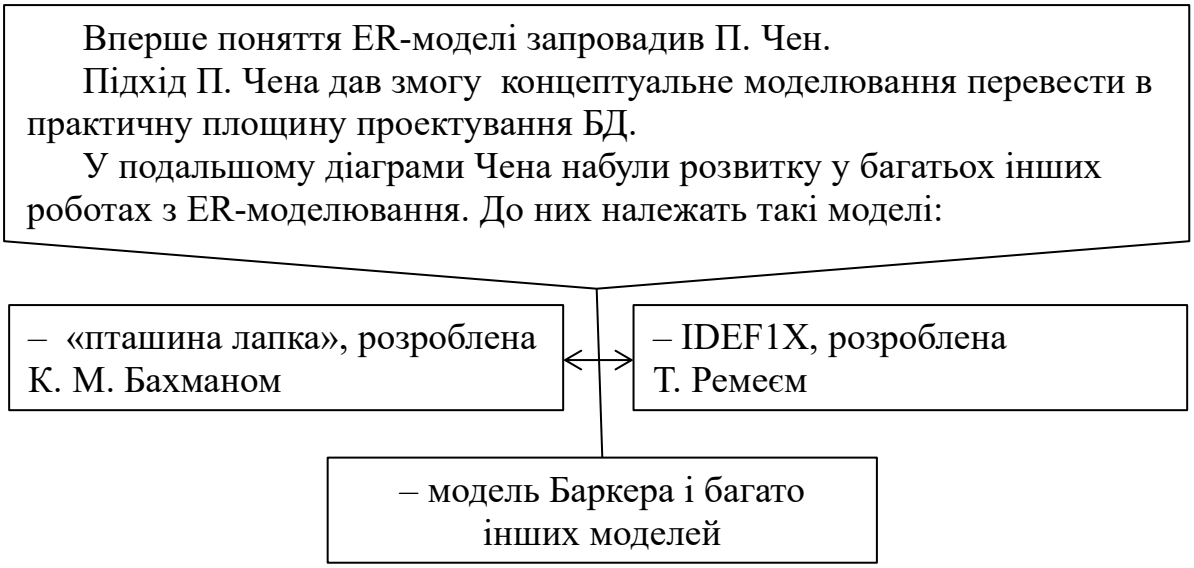


Рис. 8.3. Основні типи моделей для концептуального моделювання баз даних

8.3. Визначення складу та змісту сутності

Сутність дає змогу *моделювати клас однотипних* об'єктів [2]. Загальноприйняте позначення сутності – прямокутник (рис. 8.4) [2].

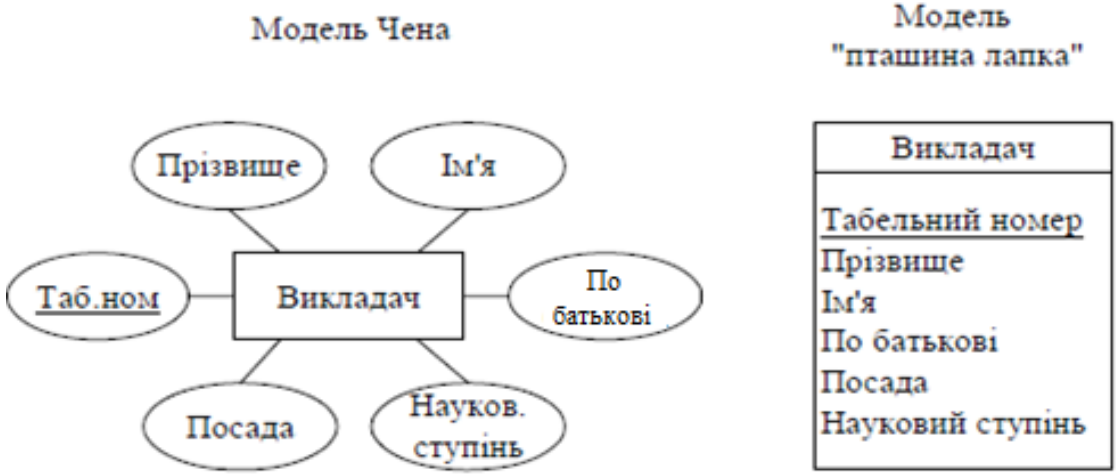


Рис. 8.4. Подання сутностей і атрибутів у ER-діаграмах П. Чена і ER-діаграмах «пташина лапка»

Сутність має *унікальне ім'я* в межах системи, що моделюється. Оскільки *сутність відповідає* деякому *класу однотипних об'єктів*, то передбачається, що в системі *існує багато екземплярів цієї сутності* [2].

Об'єкт, якому відповідає сутність, має *набір атрибутів*, що характеризують його властивості. При цьому набір атрибутів має бути таким, щоби можна було розрізняти конкретні екземпляри сутності [2].

8.4. Визначення форм зв'язків поміж сутностями

На рис. 8.5 наведно характеристики типів зв’язків поміж сутностями [2].

Рис. 8.5. Характеристики типів зв’язків поміж сутностями

На рис. 8.6, 8.7 показані відображення зв'язків у різних ER-моделях [2].

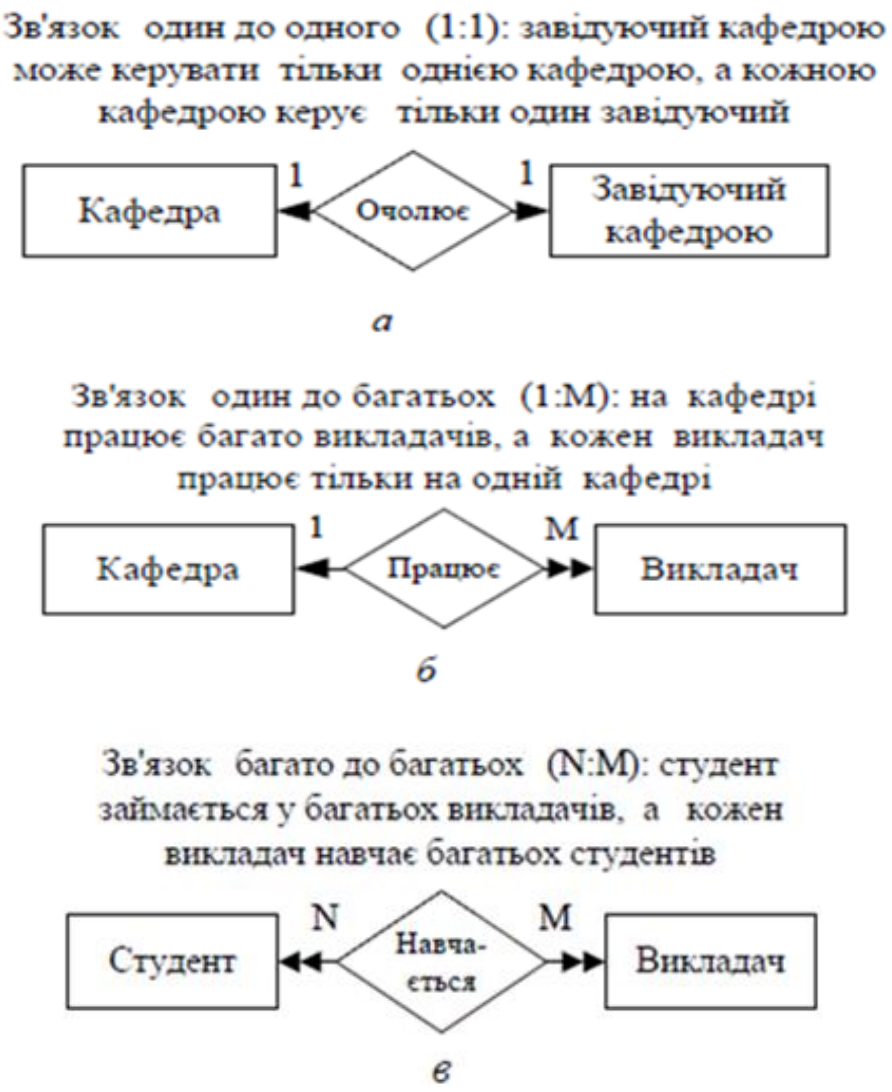


Рис. 8.7. Подання зв’язків між сутностями в ER-діаграмі П. Чена: *а* – 1:1; *б* – 1:M; в – M:N

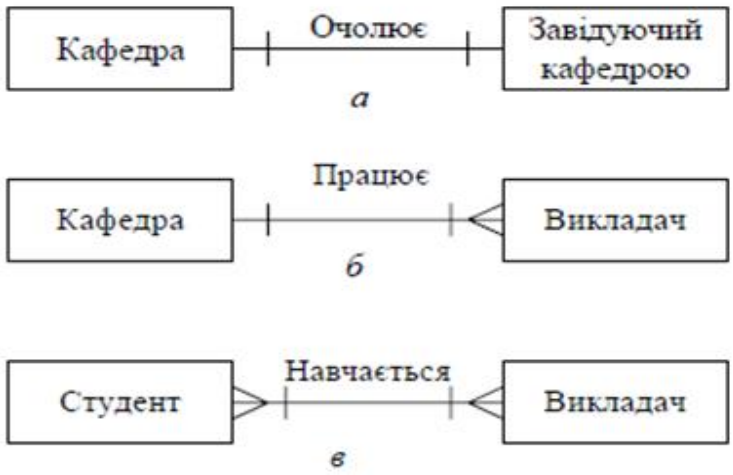


Рис. 8.8. Подання зв'язків між сутностями на діаграмі «пташина лапка»: *а* –1:1; *б* – 1:M; *в* – N:M

8.5. Визначення складу та змісту атрибутів сутності

Атрибути являють собою властивості сутності [2]. Значення кожного атрибута вибирають з відповідної множини значень, яка включає всі потенційні значення, що можуть бути присвоєні атрибуту. Ця множина значень називається *доменом.*

Атрибути залежно від складності значень, яких вони можуть набувати, поділяються на певні категорії (табл. 8.1) [2].

Таблиця 8.1

Категорії атрибутів

Тип	Властивість
Простий	Атрибут, який не може бути поділений на інші атрибути
Складовий	Атрибут, який може бути поділений на інші атрибути
Однозначний	Атрибут, який може набувати тільки одного значення
Багатознач-ний	Атрибут, який може набуватибагато значень
Похідний	Атрибут, який не зберігається в БД, а обчислюється за допомогою певного алгоритму

Приклад. Розглянемо сутність *Студент* (рис. 8.9) [2].

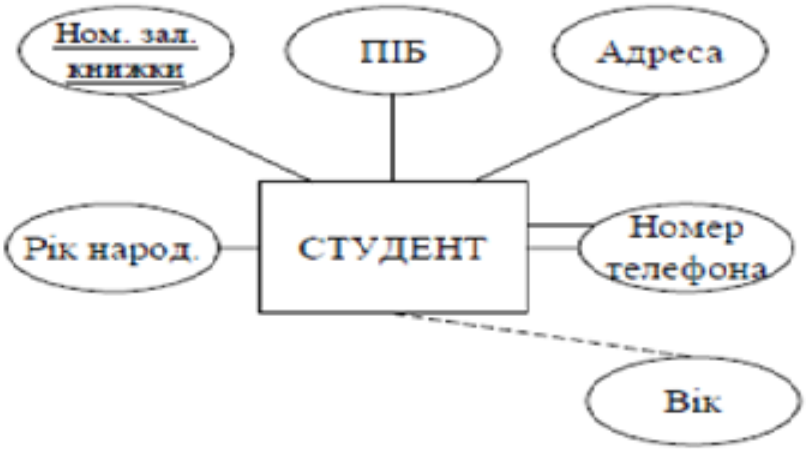


Рис. 8.9. ER-діаграма сутності *Студент*

Атрибути *Номер залікової книжки, Рік народження* є простими.

Атрибути *ПІБ* і *Адреса* є складовими. *ПІБ* може бути поділений на такі атрибути: *Прізвище, Ім'я, По батькові,* а *Адреса* – індекс, місто, вулиця, будівля, квартира.

Атрибут *Вік* є похідним: він обчислюється за значенням атрибута *Рік народження* (зображується пунктирною лінією).

Атрибут *Номер залікової книжки* є однозначним: він не може набувати двох значень для одного студента.

Атрибут *Номер телефону* є багатозначним: він може набувати декількох значень для одного студента (зображується подвійною лінією).

Атрибут або набір атрибутів сутності, застосовуваних для ідентифікації екземпляра сутності, називаються ***потенційним ключем.*** Сутність може містити декілька потенційних ключів. У прикладі як потенційні ключі можуть бути такі атрибути: *Номер залікової книжки, Прізвище*, *Ім'я, По батькові.*

Потенційний ключ, вибраний для однозначної ідентифікації кожного екземпляра сутності певного типу, називається ***первинним ключем.***

Первинний ключ вибирається за умови гарантії *унікальності його значень,* а також *мінімальної довжини атрибутів*, що входять до його складу.

У прикладі як первинний ключ служить *Номер залікової книжки.*

Контрольні запитання до розділу 8

1. Навести визначення концептуальної моделі.
2. Навести визначення семантичної моделі.
3. Навести склад ER-моделі даних.
4. Навести склад об'єктної моделі.
5. Навести склад моделі «сутність-зв'язок».
6. Навести визначення поняття «сутності».
7. Навести визначення поняття «зв'язки».
8. Навести визначення поняття «атрибути».
9. Навести визначення поняття «домени атрибутів».
10. Навести визначення поняття «ключ».

Розділ 9. Характеристики зв'язків

9.1. Визначення змісту поняття «потужність зв’язків»

Потужність зв'язку (кардинальність) відображує певну кількість екземплярів сутностей, пов'язаних з одним екземпляром зв'язаної сутності [2].

У моделі Чена потужність зв'язку відображується вказівкою відповідних чисел поруч з сутностями у форматі (x, *y).*

Потужність вказує на кількість екземплярів у зв'язаній сутності.

Приклад. Розглянемо зв'язок *Група-Студент* (рис. 9.1) [2].

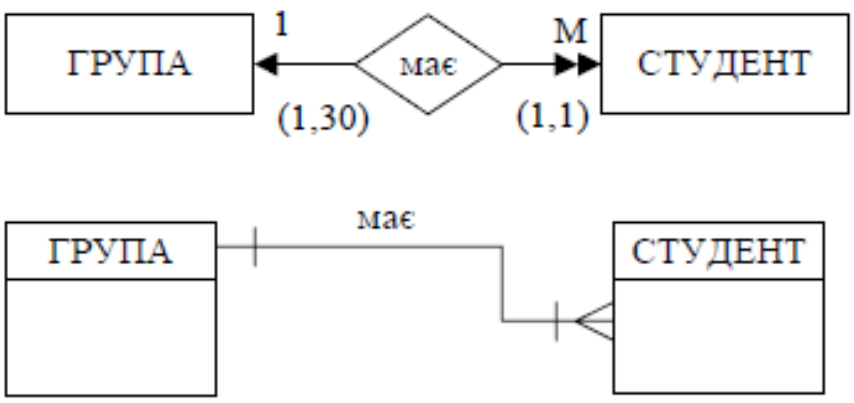


Рис. 9.1. Зв'язок і потужність в ER-діаграмах

У моделі «пташина лапка» числовий діапазон значень потужності не відображується в ER-діаграмах.

9.2. Характеристики сильних і слабких зв'язків [2]

Якщо сутність може існувати незалежно від інших сутностей, то вона є *незалежною від існування* [2]. Якщо сутність залежить від існування інших сутностей, то вона є *залежною вд існування*.

Потужність зв'язку (кардинальність) відображує певну кількість екземплярів сутностей, пов'язаних з одним екземпляром зв'язаної сутності [2].

У моделі Чена потужність зв'язку відображується вказівкою відповідних чисел поруч з сутностями у форматі (x, *y).*

Якщо одна сутність ***незалежна*** від існування іншої сутності, то зв'язок між ними називається ***неідентифікаційним зв'язком,*** або *слабким зв'язком.*

На ER-діаграмах «пташина лапка» ***слабкий*** зв'язок відображується ***штриховою*** лінією.

Ідентифікаційний зв'язок або *сильний зв'язок* існує в тому випадку, коли одна зв'язана сутність залежить від існування іншої. На ER-діаграмах «пташина лапка» ***сильний*** зв'язок відображується ***суцільною*** лінією.

Приклад. Розглянемо зв'язки *Група-Студент* і *Студент-Нагорода* (рис. 9.2) [2].

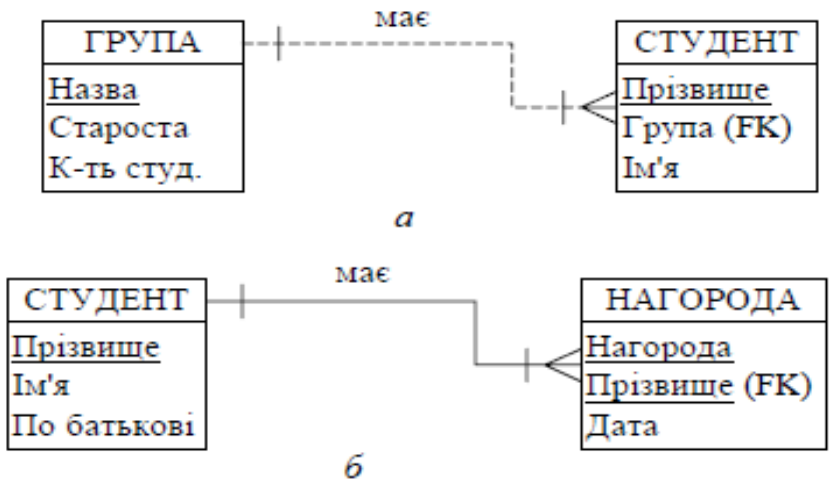


Рис. 9.2. Неідентифікаційний (*а*) та ідентифікаційний (*б*) зв’язки

Сутність *Студент* не залежить від сутності *Група,* у цьому випадку зв'язок між сутностями відображується штриховою лінією, а атрибут *Назва групи* в сутності *Студент* є зовнішнім ключем (РК).

Сутність *Нагорода* залежить від сутності *Студент,* у цьому випадку зв'язок між сутностями відображується суцільною лінією, а атрибут *Прізвище студента* в сутності *Нагорода* є частиною первинного ключа (РК) і одночасно зовнішнім ключем (РК).

9.3. Характеристики атрибутів зв'язків

Як і сутності, ***зв'язки можуть мати атрибути*** [2].

Приклад. Атрибути *День* і *Аудиторія* належать до зв'язку між сутностями *Студент* і *Дисципліна* (рис. 9.3) [2].

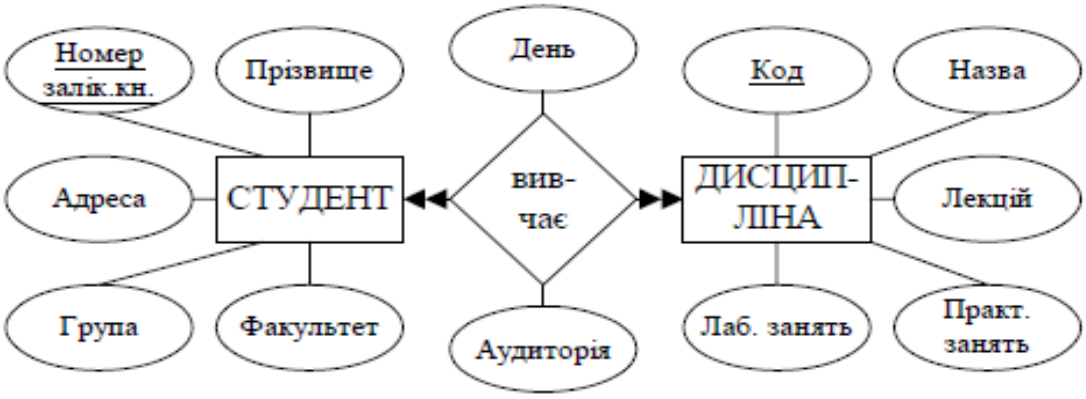


Рис. 9.3. ER-діаграма з атрибутами *Дені* і *Аудиторія*

9.4. Характеристики обов'язкових і необов'язкових зв'язків

Участь сутності може бути ***обов'язковою*** або ***необов'язковою*** [2].

Якщо один екземпляр сутності не потребує наявності відповідного екземпляра сутності в окремому зв'язку, то участь сутності у зв'язку є ***необов'язковою.***

Необов'язкова сутність позначається невеликим колом з боку необов'язкової сутності.

Існування необов'язковості вказує на те, що для необов'язкової сутності мінімальне значення потужності зв'язку дорівнює 0.

Участь сутності у зв'язку буде ***обов'язковою,*** якщо ***кожен екземпляр сутності обов'язково потребує*** відповідного екземпляра сутності в окремому зв'язку.

При обов'язковому зв'язку для обов'язкової сутності мінімальна потужність зв'язку дорівнює 1.

Приклад. Зв'язок між сутностями *Студент* і *Нагорода* є необов'язковим (рис. 9.4) [2]. Необов'язково кожен студент має нагороду, але якщо є нагорода, то вона обов'язково пов'язана з певним студентом.

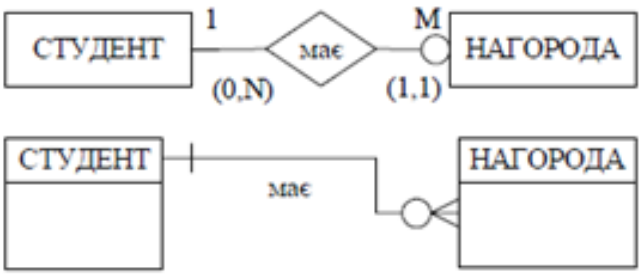


Рис. 9.4. Зв'язок між сутностями *Студент* і *Нагорода*

Контрольні запитання до розділу 9

1. Навести визначення змісту поняття «потужність зв’язків».
2. Навести визначення змісту поняття «сильні зв'язки».
3. Навести визначення змісту поняття «слабкі зв'язки».
4. Навести визначення змісту поняття «атрибути зв'язків».
5. Навести визначення змісту поняття «обов'язкові зв'язки».
6. Навести визначення змісту поняття «необов'язкові зв'язки».

Розділ 10. Характеристики слабких сутностей

10.1. Визначення поняття «слабка сутність» [2]

На рис. 10.1 наведено умови віднесення сутності до «слабкої».

Рис. 10.1. Умови віднесення сутності до «слабкої»

Приклад. Сутність *Нагорода* є слабкою відносно сутності *Студент:* вона залежить від існування цієї сутності і до її первинного ключа входить первинний ключ сутності *Студент* (рис. 10.2) [2].

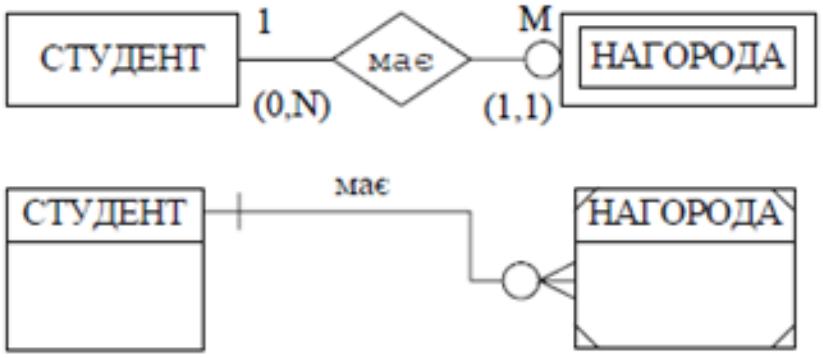


Рис. 10.2. Слабка сутність на діаграмах П. Чена і «пташина лапка»

10.2. Характеристики складних зв'язків

Використання зв'язків більш високого порядку дає можливість у багатьох випадках краще відобразити семантику проблемної галузі.

Приклад. Сутності *Викладач, Дисципліна* та *Екзамен* утворюють ***тернарний зв'язок*** (рис. 10.3) [2].

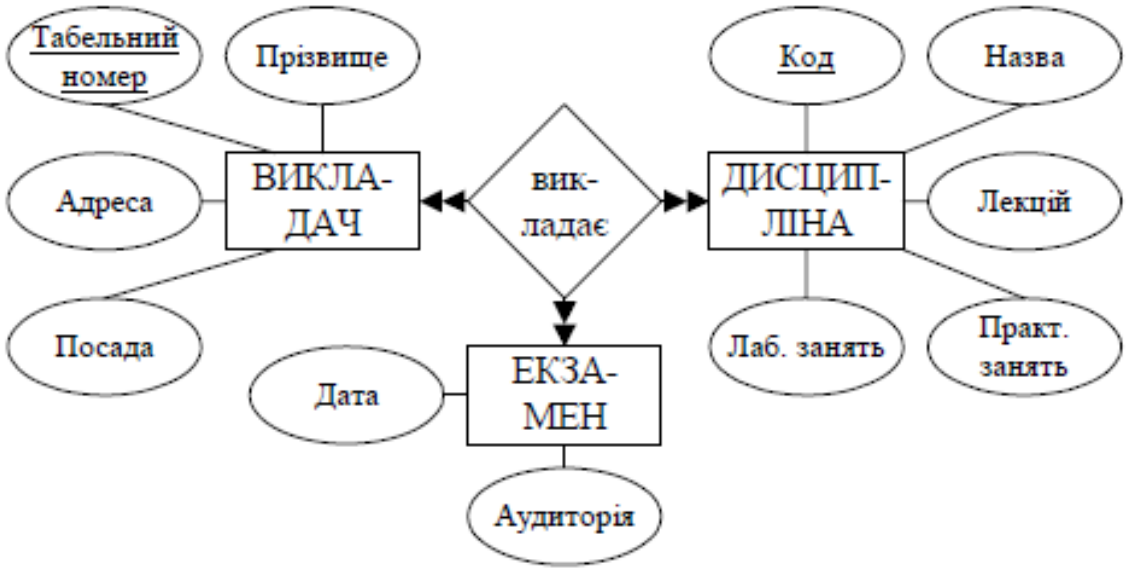


Рис. 10.3. Тернарний зв’язок між трьома сутностями на діаграмі П. Чена

10.3. Особливості розкриття змісту рекурсивних зв'язків

Рекурсивний зв'язок має місце, коли є зв'язок між екземплярами одного і того самого набору сутностей [2].

Приклад. Розглянемо можливі варіанти рекурсивних зв'язків (рис. 10.4) [2]. Зв'язок 1:1 зображує висловлювання: «викладач може бути одружений тільки з одним співробітником».

Зв'язок 1:М зображує таке висловлювання: «викладач, якщо він є завідуючим кафедрою, керує декількома викладачами, а викладачі мають тільки одного керівника – завідуючого кафедрою».

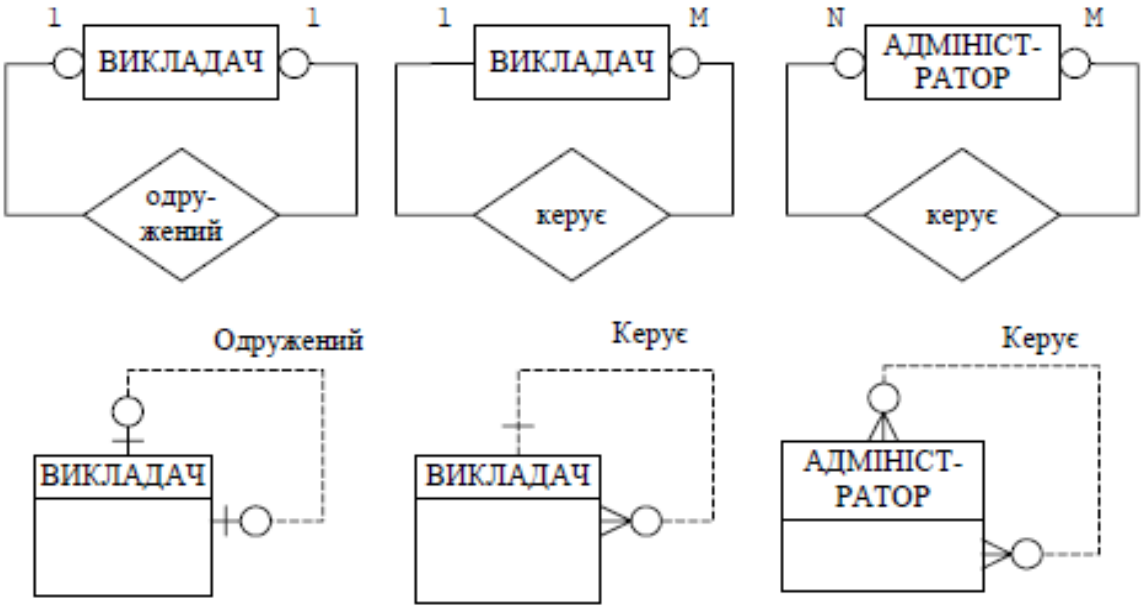


Рис. 10.4. Подання рекурсивних зв'язків на діаграмі П. Чена і моделі «пташина лапка» [2]

Зв'язок М:К являє собою висловлювання «адміністратор має декілька підлеглих-адміністраторів і у свою чергу керівників-адміністраторів».

Між двома сутностями може бути декілька зв'язків з різними змістовими навантаженнями [2].

Приклад. Між сутностями *Викладач* і *Студент* можна встановити такі змістові зв'язки: *Викладає* і *Керівництво дипломним проєктуванням.* Викладач викладає для багатьох студентів, для кожного студента викладає багато викладачів. Кожен студент обов'язково повинен мати одного керівника дипломного проєкту, але необов'язково кожен викладач керує дипломниками (рис. 10.5) [2].

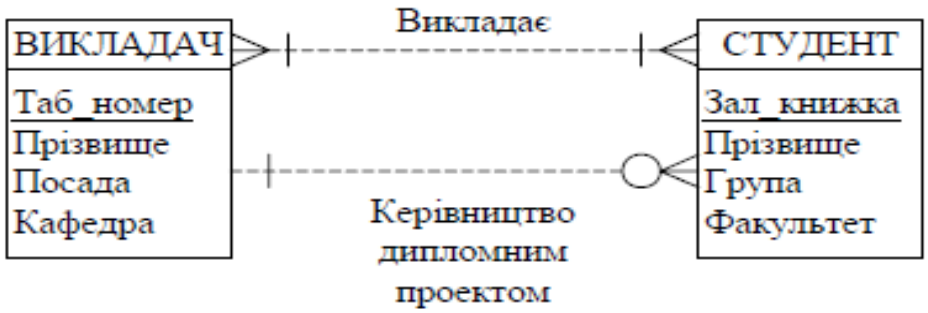


Рис. 10.5. Різні зв’язки між двома сутностями

Контрольні запитання до розділу 10

1. Визначення поняття «слабка сутність».

2. Як позначається на слабка сутність на діаграмі Чена?

3. Як позначається слабка сутність на діаграмі «пташина лапка»?

4. Що забезпечує використання зв'язків більш високого порядку?

5. Коли має місце рекурсивний зв'язок?

6. Чи може бути між двома сутностями декілька зв'язків з різними змістовими навантаженнями?

7. Навести зміст зв'язку 1:1.

8. Навести зміст зв'язку 1:М.

9. Навести зміст зв'язку М:К.

Розділ 11. Характеристика розширених моделей «сутність-зв'язок»

11.1. Опис розширеної моделі «сутність-зв'язок»

Для задоволення нових потреб, що висуваються більш складними застосуваннями, до ***семантичного*** моделювання були введені додаткові концепції, які розширюють його можливості. Така модель має назву *розширеної ER-моделі* [2].

Вона включає всі концепції ER-моделі плюс концепції:
– *уточнення;*
– *узагальнення;*
– *агрегування*;
– *композиції.*
Додаткові концепції базуються на таких поняттях:
– *суперклас*;
– *підклас.*
Суперклас може мати декілька підкласів

Використання понять *суперклас і підклас* дає змогу визначити для підкласів *власні атрибути* і атрибути, що наслідуються від суперкласу. Так, підклас *Викладач* повинен мати ті самі атрибути, що і всі *Співробітники.* Однак він має і свої власні атрибути, не визначені для інших категорій працівників університету.

Уточнення − це процес збільшення відмінності між окремими екземплярами об'єкта шляхом визначення їхніх відмінних характеристик. Цей процес є низхідним, наприклад перехід від об'єкта *Співробітник* до об'єктів *Викладач* і *Керівник.*

Узагальнення − це процес зведення відмінностей між об'єктами до мінімуму шляхом виділення їхніх спільних характеристик. Цей процес є висхідним, наприклад перехід від об'єктів *Викладач* і *Керівник* до об'єкта *Співробітник.*

У процесі проведення *уточнення* або *узагальнення* можуть застосовуватися обмеження:

– ступеня участі;

– неперетинання.

Підкласи набору сутностей можуть перетинатися і не перетинатися. Якщо підкласи суперкласу *не перетинаються,* то це означає, що кожен екземпляр сутності може бути елементом тільки одного з підкласів.

Не кожен елемент суперкласу має бути елементом одного з підкласів.

Зв'язок суперкласу з підкласом з *обов'язковою участю,* вказує на те, що кожен елемент суперкласу має бути також елементом підкласу.

Приклад. Студент обов'язково навчається або за денною, або заочною, або вечірньою формою навчання, або екстернатом (рис. 11.1).

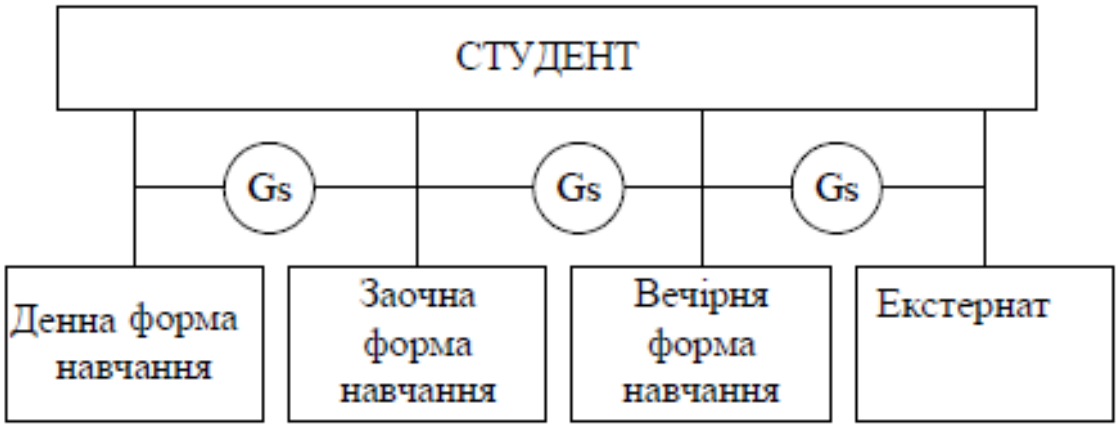


Рис. 11.1. Діаграма зв'язку суперкласу з підкласом з обов'язковою участю [2]

Зв'язок суперкласу з підкласом з *необов'язковою участю*, вказує на те, що деякі елементи суперкласу можуть не належати жодному з підкласів.

Уведення понять суперкласів і підкласів дає змогу уникнути опису різних екземплярів сутності, що можуть мати різні атрибути. Це може призвести до появи великої кількості незаповнених атрибутів. До того ж індивідуальні атрибути можуть показати зв'язки, притаманні тільки конкретним екземплярам, а не всім екземплярам сутностей.

Для моделювання інших видів зв'язків вводиться поняття агрегування і композиції.

Агрегування являє собою зв'язок типу «входить до складу» або «включає» між двома сутностями, одна з яких являє «ціле», а інша − «частину».

Композиція – це особлива форма агрегування, яка є залежністю між сутностями, що характеризуються повною приналежністю і збігом термінів існування між «цілим» і «частиною».

11.2. Задачі побудови моделей «сутність-зв'язок» [2]

На рис. 11.2 наведено характеристики дефектів з’єднання та розгалуження.

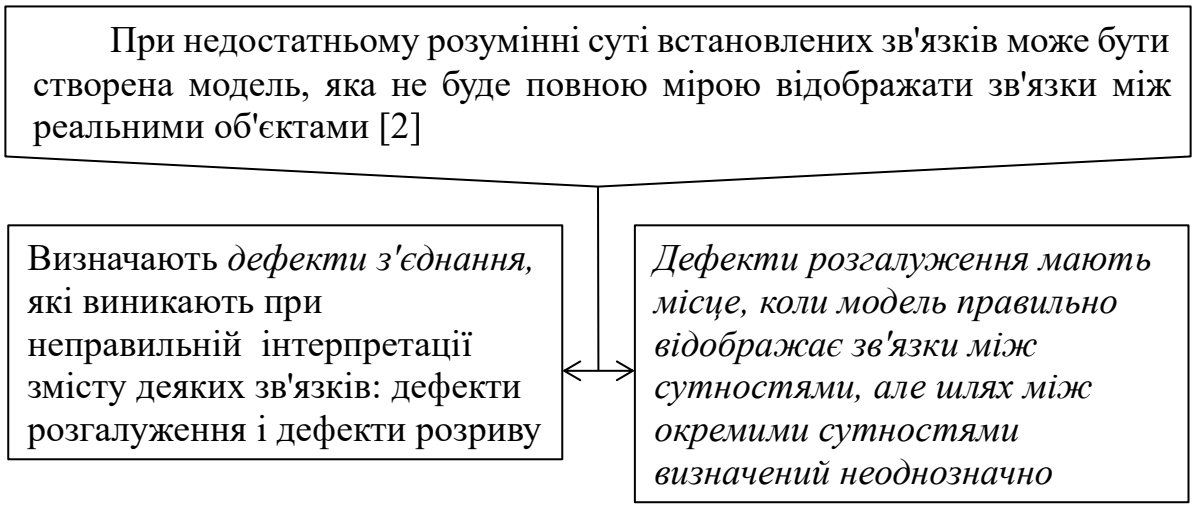


Рис. 11.2. Характеристики дефектів з’єднання та розгалуження

Приклад. Розглянемо такі зв'язки: на факультеті навчається багато студентів, до складу факультету входить багато груп (рис. 11.3) [2].

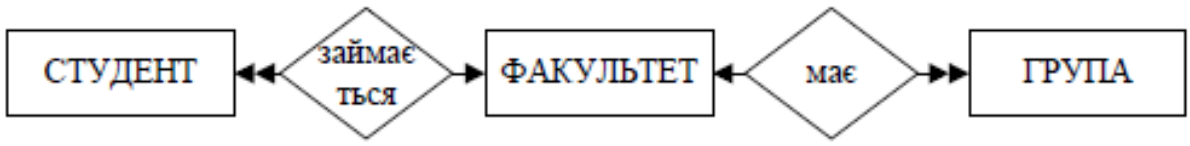


Рис. 11.3. Приклад дефекту розгалуження в ER-моделі

Усунути цю проблему можна шляхом перебудови моделі для подання правильної взаємодії цих сутностей (рис. 11.4) [2].

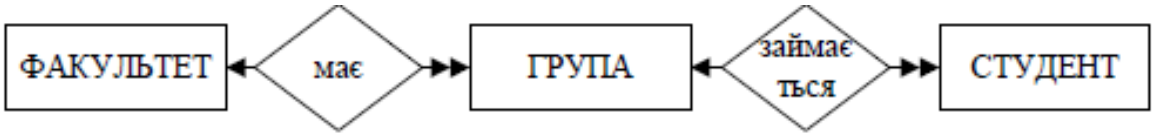


Рис. 11.4. Усунення дефекту розгалуження в прикладі ER-моделі

Отже, тепер відповідь на попереднє питання не є проблемою.

Дефекти розриву виникають у тому випадку, коли в моделі передбачається наявність зв'язку між декількома сутностями. Цей дефект виникає тоді, коли існує один або декілька зв'язків з мінімальною потужністю, рівною 0, що визначає необов'язкову участь, і ці зв'язки складають частину шляху між взаємозв'язаними сутностями.

Контрольні запитання до розділу 11

1. Навести визначення розширеної моделі «сутність-зв'язок».
2. Навести визначення концепції «*уточнення*» для ER-моделі.
3. Навести визначення концепції «*узагальнення*» ER-моделі.
4. Навести визначення концепції «*агрегування*» ER-моделі.
5. Навести визначення концепції «*композиції*» ER-моделі.
6. Навести визначення поняття «*дефекти з'єднання*».
7. Навести визначення поняття «*дефекти розгалуження*».

Розділ 12. Алгоритм побудови моделі «сутність-зв'язок»

Процес концептуального проєктування БД є ітеративним і заснований на повторюваних операціях і процедурах. Спочатку створюється базова ER-модель певної предметної галузі. При дослідженні цієї моделі, як правило, з'являться додаткові сутності, атрибути і зв'язки. Після цього ER-модель буде змінюватися. Процес змін повторюється доти, поки концептуальна модель не буде відображувати предметну галузь [2].

На рис. 12.1 наведено перелік завдань, вирішуваних при опитуванні фахівців.

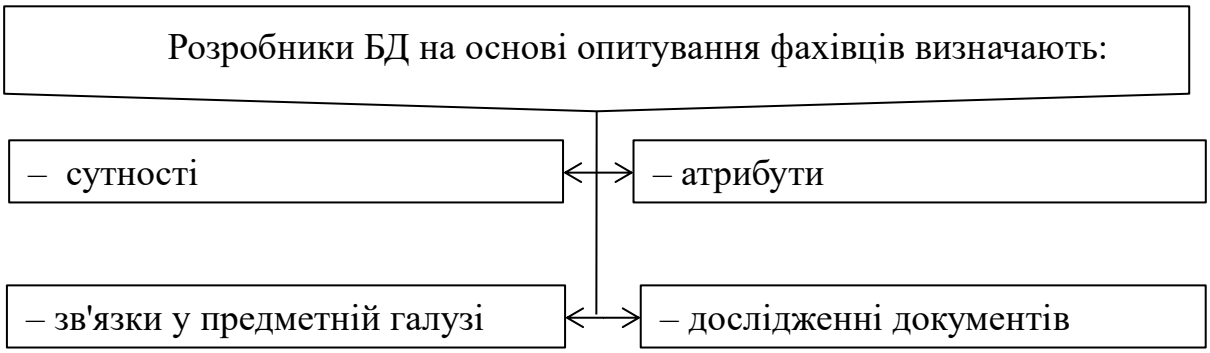


Рис. 12.1. Перелік завдань, вирішуваних при опитуванні фахівців

12.1. Визначення завдань, вирішуваних інформаційною системою

Мета створення бази даних така [2]:

– забезпечення адміністрації університету довідковою інформацією про факультети, кафедри, спеціальності, викладачів, студентів і дисципліни, що викладаються у ВНЗ;

– контроль успішності студентів.

12.2. Визначення сутностей предметної галузі

У результаті дослідження й аналізу предметної галузі для інформаційної системи «Заклад вищої освіти» були визначені *сутності*, *атрибути і первинні ключі* (табл. 12.1) [2].

Таблиця 12.1

Сутності, атрибути і первинні ключі предметної галузі ЗВО [2]

Сутність	**Атрибути**	**Первинний ключ**
ФАКУЛЬТЕТ	*Код.* *Назва* *Декан*	*Код факультету*
СПЕЦІАЛЬНІСТЬ	*Код* *Назва* *Вимоги*	*Код спеціальності*
ГРУПА	*Код. Назва* *Кількість студентів* *Староста*	*Код групи*
СТУДЕНТ	*Номер залікової книжки* *Прізвище* *Адреса* *Рік народження*	*Номер залікової книжки*
КАФЕДРА	*Код. Назва* *Завідуючий кафедрою* *Кількість викладачів*	*Код кафедри*
ВИКЛАДАЧ	*Табельний номер* *Прізвище* *Посада* *Науковий ступінь*	*Табельний номер викладача*
ДИСЦИПЛІНА	*Код. Назва* *Кількість годин* *Семестр*	*Код дисципліни*

Зв'язки сутностей визначаються на основі *бізнес-правил*, побудованих з урахуванням *організаційної структури та операцій*, що виконуються в системі [2]:

– в університеті існує декілька факультетів;

– на факультеті навчаються групи студентів за певними спеціальностями;

– до складу групи входять студенти;

– факультет об'єднує декілька кафедр;

– на кожній кафедрі працюють декілька викладачів;

– на кожній спеціальності викладається ряд дисциплін, які проводять викладачі з різних кафедр;

– з кожної дисципліни своєї спеціальності студенти складають іспит або залік;

– не кожен викладач читає дисципліни (наприклад асистент) і не з кожної дисципліни є викладач (наприклад нова дисципліна, з якої викладача ще не призначено).

Для спрощення концептуальної моделі бази даних цілий ряд об'єктів і бізнес-правил ЗВО залишився нерозглянутим.

Дослідження предметної галузі виявило, що всі сутності є сильними, а зв'язки між сутностями – неідентифікуючими.

Зв'язок між сутностями *Студент* і *Дисципліна* має атрибут *Оцінка.*

12.3. Приклад побудови ER-діаграми

На основі завдань, що були поставлені перед інформаційною системою, і на основі аналізу предметної галузі, побудована ER-діаграма ВНЗ (рис. 12.2) [2].

Перевірка на збитковість передбачає перевірку ER-моделі з метою виявлення збиткових даних і вилучення їх у тому випадку, якщо вони визначені. Збиткові зв'язки виявляються в тому, що між двома сутностями є декілька шляхів, які дублюють один одний (це не стосується зв'язків, що є різними асоціаціями) [2].

Перевірка моделі на відповідність транзакціям користувачів виконується на основі таких підходів:

– перевірка того, чи подає модель усю інформацію (сутності, атрибути, зв'язки), необхідну для кожної транзакції;

– перевірка за ER-діаграмою маршруту кожної транзакції.

Перевірка моделі на збитковість і відповідність транзакціям користувачів дозволяє зробити висновок, що концептуальний проєкт відповідає всім необхідним вимогам.

Застосування ER-діаграм дає змогу забезпечити просте і наочне уявлення про головні логічні об'єкти БД і зв'язки, що між цими об'єктами існують.

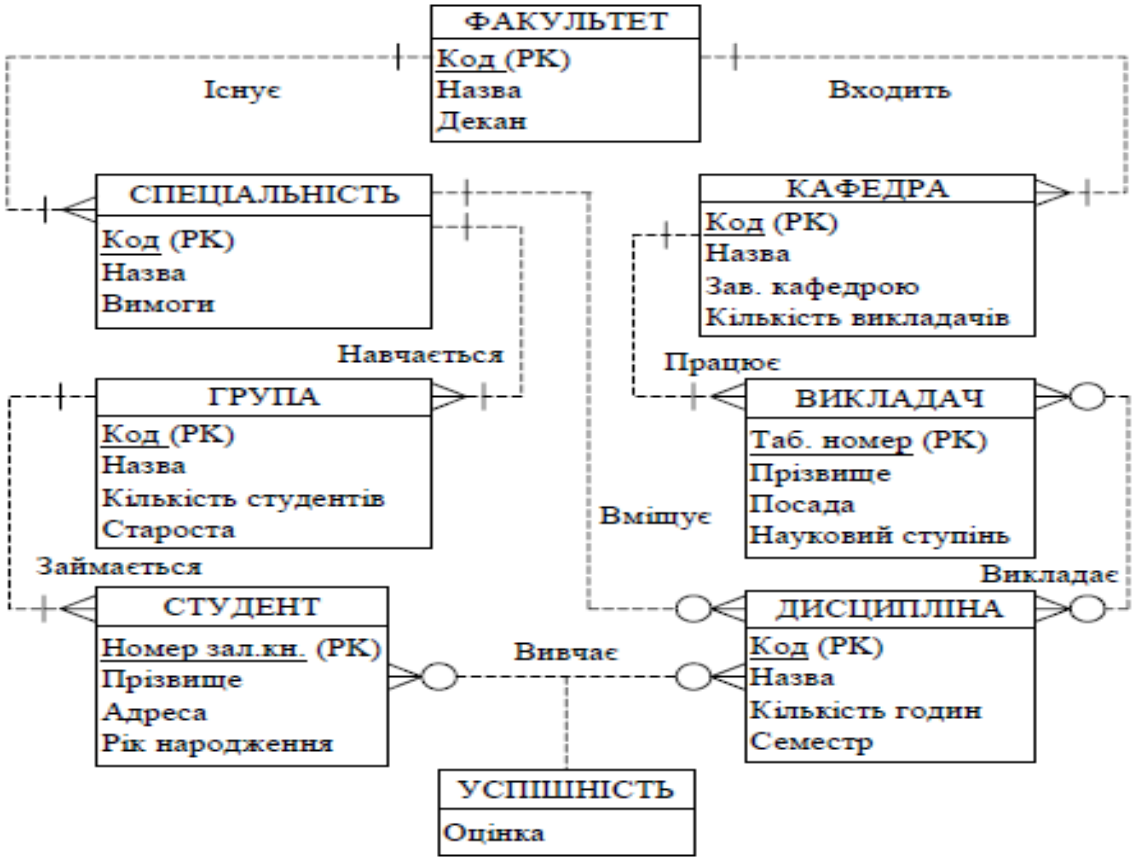


Рис. 12.2. ER-діаграма предметної галузі ЗАКЛАД ВИЩОЇ ОСВІТИ

Контрольні запитання до розділу 12

1. На чому засновано процес концептуального проєктування БД?
2. На основі чого розробники БД визначають сутності, атрибути, зв'язки у предметній галузі?
3. У чому полягає мета створення бази даних?
4. У результаті чого визначаються сутності, атрибути і первинні ключі БД?
5. На основі чого будується ER-діаграма?
6. З якою метою проводиться перевірка на збитковість?
7. З якою метою проводиться перевірка моделі на відповідність транзакціям користувачів?
8. Що є недоліком ER-моделей?

Розділ 13. Логічне проєктування баз даних

13.1. Етапи логічного проєктування

Логічне проєктування виконується для певної моделі даних.

Проєктування являє собою циклічний процес. Етапи логічного проєктування наведено на рис. 13.1 [2].

Рис. 13.1. Етапи логічного проєктування бази даних

Для *реляційної моделі даних* ***логічне проєктування*** полягає у створенні реляційної схеми, визначенні кількості і структури таблиць, формуванні запитів до БД, визначенні типів звітних документів, розробленні алгоритмів обробки інформації, створенні форм для введення і редагування даних у БД і вирішенні цілого ряду інших завдань. *Концептуальні* моделі за

певними правилами перетворюються в логічні моделі даних. Коректність логічних моделей перевіряється за допомогою *правил нормалізації,* які дають змогу переконатися в структурній узгодженості, логічній цілісності і мінімальній збитковості прийнятої моделі даних.

Модель також перевіряється з метою виявлення можливостей *виконання транзакцій,* що будуть задаватися користувачами.

13.2. Алгоритм спрощення концептуальної моделі

Першим кроком спрощення концептуальної моделі є попередні перетворення з метою усунення зв'язків, несумісних з реляційною моделлю (рис. 13.2) [2].

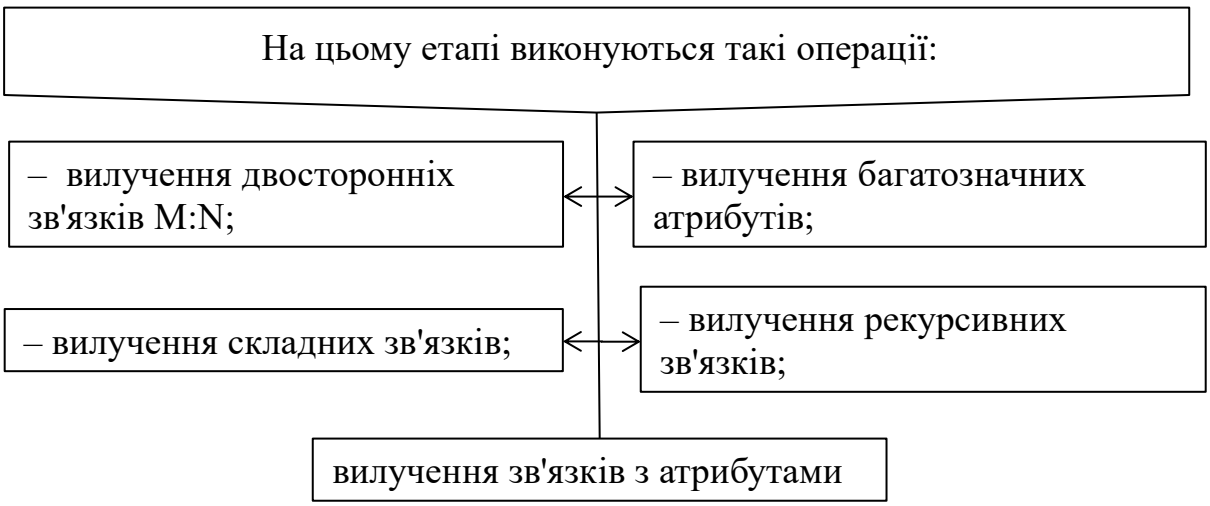


Рис. 13.2. Зміст завдань для етапу спрощення моделі

13.3. Метод вилучення зв'язків «багато до багатьох»

Перетворення зв'язку «багато до багатьох» виконується шляхом введення проміжної сутності з заміною одного зв'язку М:К двома зв'язками 1:К з новою сутністю (рис. 13.3).

Приклад. Викладач може викладати багато *Дисциплін,* одну *Дисципліну* викладає багато *Викладачів* [2].

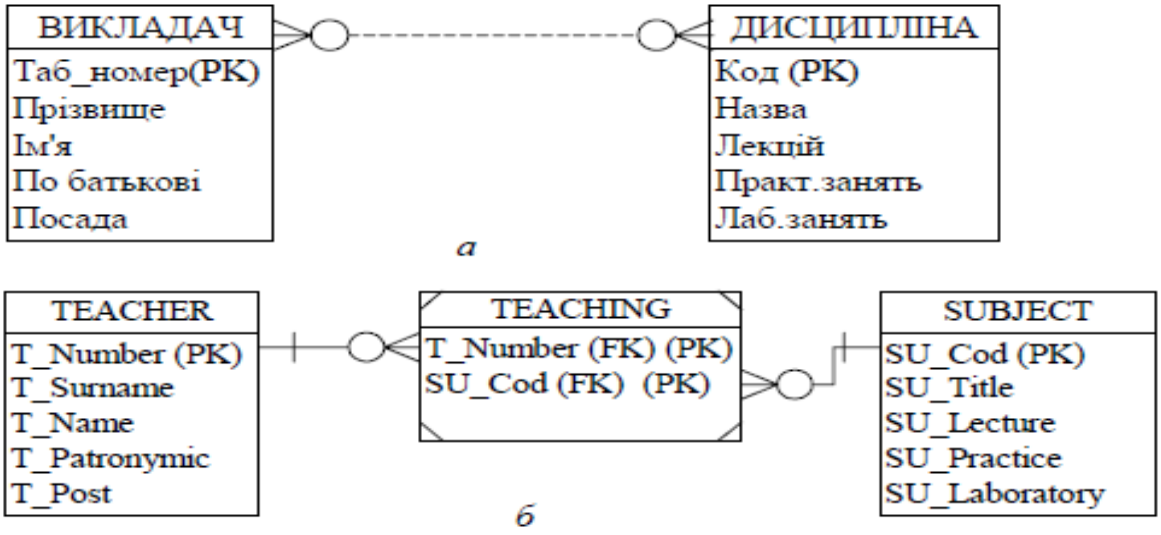


Рис. 13.3. Перетворення зв'язку «багато до багатьох»:
а – зв'язок M:N; *б* – результат перетворення – два зв'язки 1:N

13.4. Алгоритм вилучення складних зв'язків [2]

Для вилучення складних зв'язків [2] виконуються такі операції:

– до моделі вводиться нова сутність;

– складний зв'язок заміняється бінарними зв'язками «один до багатьох» зі знов створеною сутністю;

– кількість бінарних зв'язків дорівнює ступеню складності зв'язку.

Приклад. Викладач може викладати багато *Дисциплін,* одну *Дисципліну* викладає багато *Викладачів.* З *Дисципліни Викладач* проводить *Іспит* (рис. 13.4) [2].

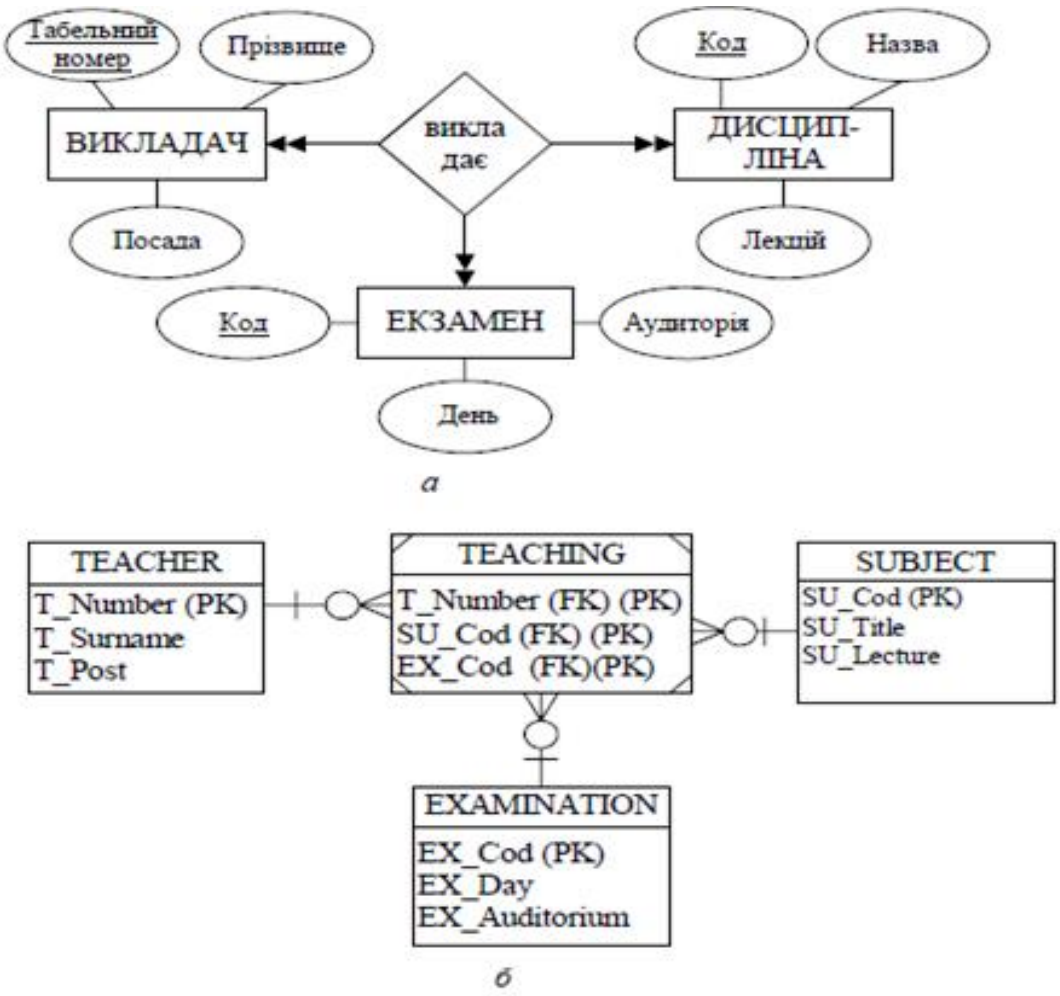


Рис. 13.4. Подання тернарного зв'язку бінарними зв'язками: *а* – складний зв'язок *Викладає; б* – декомпозиція складного зв'язку

Контрольні запитання до розділу 13

1. Навести етапи логічного проєктування.
2. Для чого виконується спрощення концептуальної моделі?
3. Як здійснюється вилучення двосторонніх зв'язків «багато до багатьох»?
4. Як здійснюється вилучення складних зв'язків?
5. Як здійснюється вилучення багатозначних атрибутів?
6. Як здійснюється вилучення рекурсивних зв'язків?
7. Як здійснюється вилучення зв'язків з атрибутами?

Розділ 14. Алгоритм перетворення ER-діаграм у реляційні структури

Для ER-моделі існує алгоритм однозначного перетворення її в реляційну модель даних. Розглянемо правила перетворення ER-моделі в реляційну модель.

14.1. Формування відношень та атрибутів для сутностей

Для кожної сутності створюється відношення, кожен атрибут сутності стає атрибутом відповідного відношення [2].

Для *сильних сутностей* первинний ключ сутності стає PRIMARI KEY (PK) відповідного відношення.

Приклад. Розглянемо сутність *Студент* (рис. 14.1) [2].

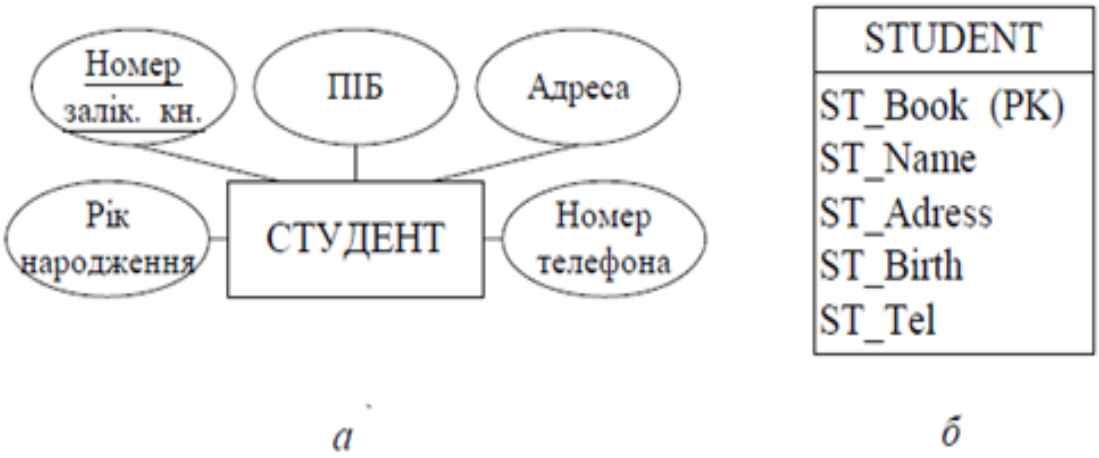


Рис. 14.1. Перетворення сутності *Студент* (*а*)
у відношення *Student* (*б*)

Для слабких сутностей первинний ключ частково або повністю залежить від ключа сутності володаря (декількох володарів), тобто РК визначається тільки тоді, коли визначено всі РК сутностей володарів.

Приклад. Розглянемо перетворення сильної сутності *Студент* і слабкої сутності *Нагорода* (рис. 14.2) [2].

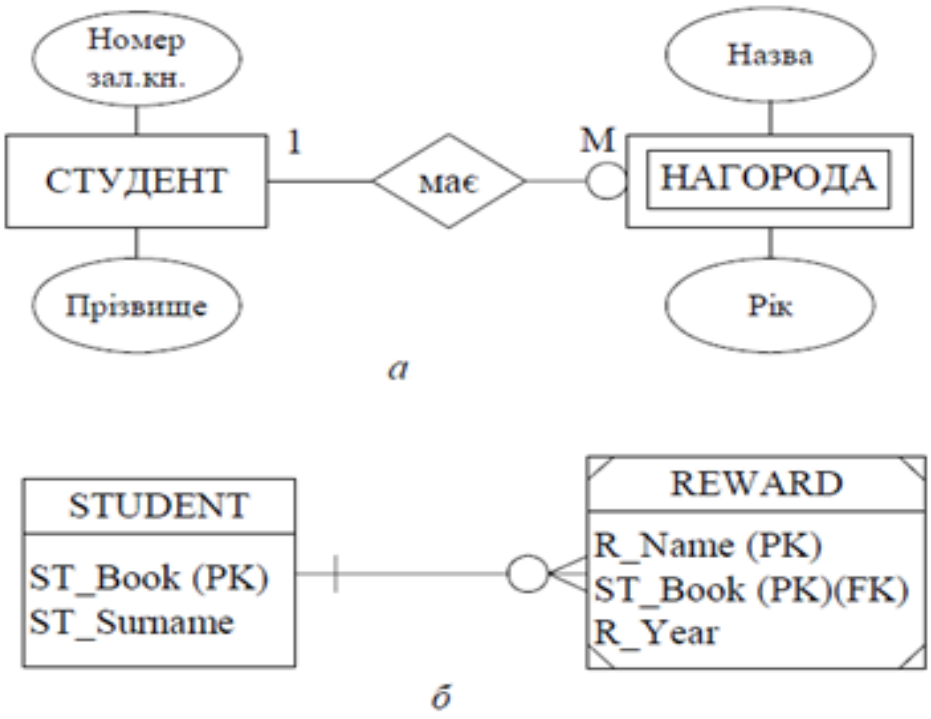


Рис. 14.2. Перетворення сильної сутності *Студент* і слабкої сутності *Нагорода* з ідентифікуючим зв'язком між ними (*а*) у пов'язані відношення *Student* і *Reward* (*б*)

14.2. Склад і зміст зв'язків атрибутів

Після перетворення концептуальної моделі залишаються такі типи зв'язків [2]:

– «один до одного»;
– «один до багатьох»;
– рекурсивні зв'язки;
– суперклас-підклас.

Для кожного типу зв'язку залежно від умов зв'язування існують свої різновиди. Зв'язки між відношеннями в реляційній моделі реалізуються шляхом використання первинних і зовнішніх ключів.

14.3. Склад і зміст зв'язків «один до одного»

У концептуальних моделях даних визначають такі обмеження ступеня участі сутностей [2]:

– обов'язкова участь для обох сутностей;

– обов'язкова участь для однієї сутності;

– необов'язкова участь для обох сутностей.

Залежно від обмежень перетворення на реляційну модель будуть різні.

Приклад. Розглянемо можливі варіанти перетворення зв'язку між сутностями *Викладач* і *Дисципліна* (рис. 14.3) [2].

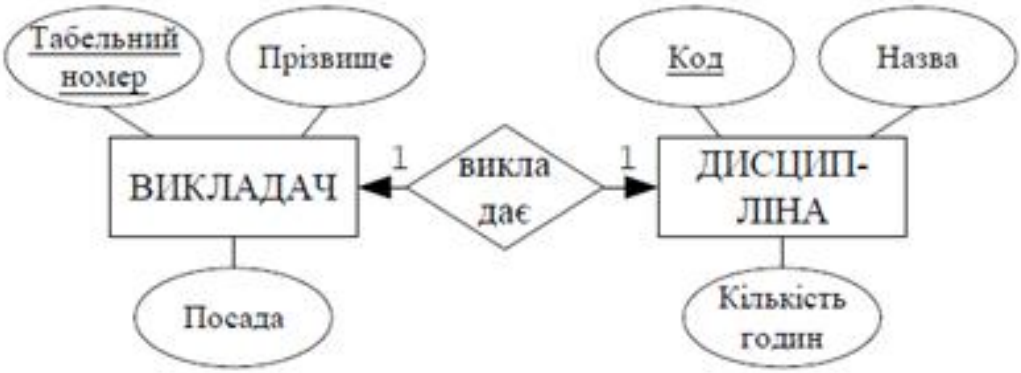
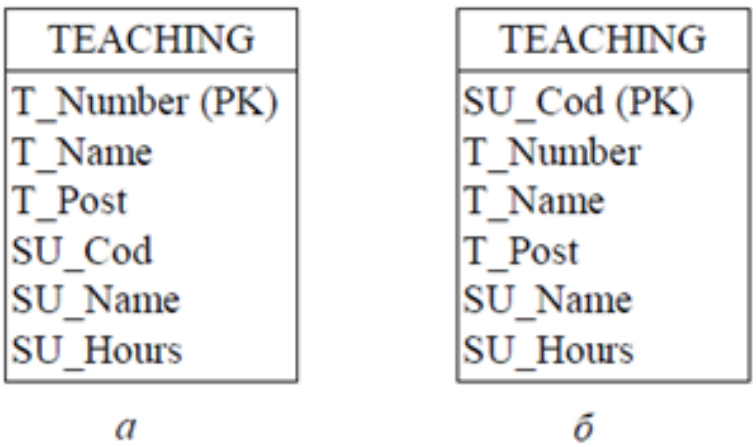


Рис. 14.3. Зв’язок 1:1 між сутностями *Викладач* і *Дисципліна*

1. *Обов'язкова участь для обох сутностей*. Припустимо, що кожен викладач обов'язково викладає одну дисципліну і кожну дисципліну обов'язково викладає один викладач. У цьому випадку реляційна структура буде складатися з одного відношення і мати один з варіантів, наведених на рис. 14.4 [2].



а *б*

Рис. 14.4. Варіанти відношень для перетворення зв'язку 1:1 з обов'язковою участю для обох сутностей

2. *Обов'язкова участь для однієї сутності*. Припустимо, що кожен викладач обов'язково викладає одну дисципліну, а за кожною дисципліною необов'язково закріплений викладач. У цьому випадку сутність, що є необов'язковою *(Дисципліна)*, виступає як батьківська сутність, а обов'язкова сутність визначається як дочірня *(Викладач)*. Реляційна структура показана на рис. 14.5 [2].

Зовнішній атрибут SU_Cod (PK) також може бути ключем для *Викладача.*

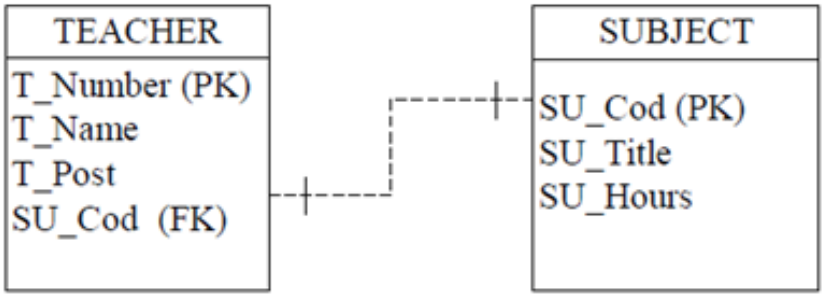


Рис. 14.5. Перетворення зв'язку 1:1 з обов'язковою участю сутності *Викладач* і необов'язковою участю сутності *Дисципліна*

3. *Необов'язкова участь для обох сутностей*. Припустимо, що необов'язково кожен викладач викладає дисципліни і за кожною дисципліною закріплений викладач. У цьому випадку можливі три реляційні структури (рис. 14.6) [2].

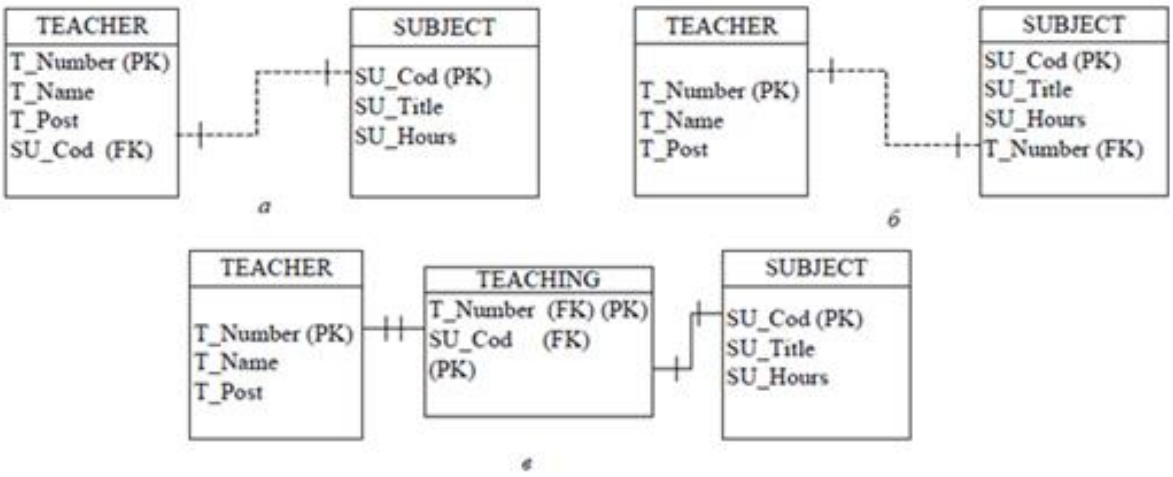


Рис.14.6. Варіанти реляційних схем відношень

14.4. Склад і зміст зв'язків «один до багатьох»

До кожного відношення, що відповідає підлеглій (дочірній) сутності, додається набір атрибутів основної (батьківської) сутності, який складає первинний ключ основної сутності.

У відношенні, що відповідає підлеглій сутності, цей набір атрибутів стає зовнішнім ключем.

Приклад. Розглянемо можливі варіанти перетворення зв'язку між сутностями *Викладач* і *Дисципліна* (рис. 14.7) [2].

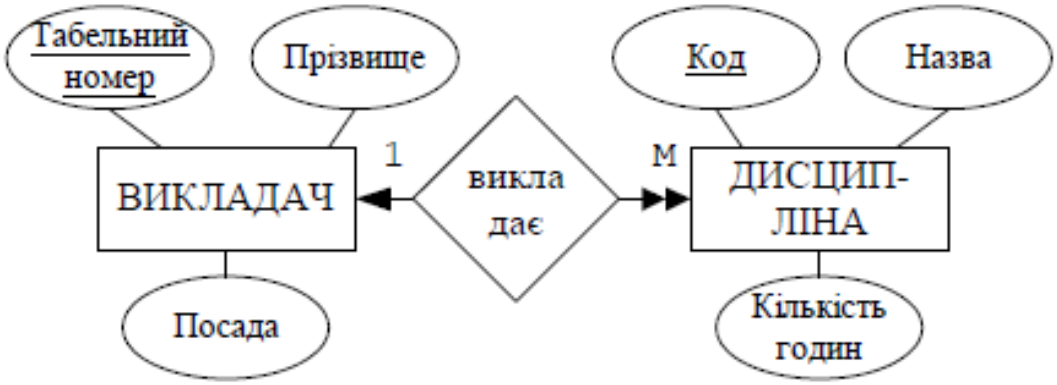


Рис. 14.7. Зв’язок 1:М між сутностями *Викладач* і *Дисципліна*

1. Необов'язкова участь сутності *Викладач* і обов'язкова участь сутності *Дисципліна* (рис. 14.8) [2].

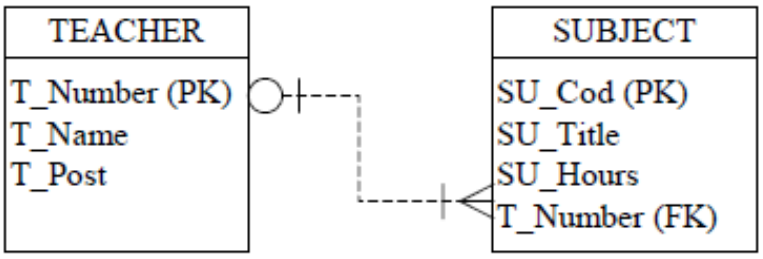


Рис. 14.8. Перетворення зв'язку 1:М

2. Необов'язкова участь сутності *Викладач* і необов'язкова участь сутності *Дисципліна* (рис. 14.9).

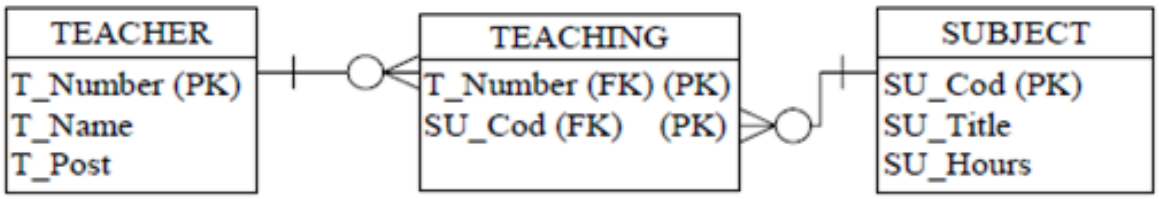


Рис. 14.9. Перетворення зв'язку 1:M з необов'язковою участю сутності *Викладач* і необов'язковою участю сутності *Дисципліна*

14.5. Перевірка відношень за допомогою правил нормалізації

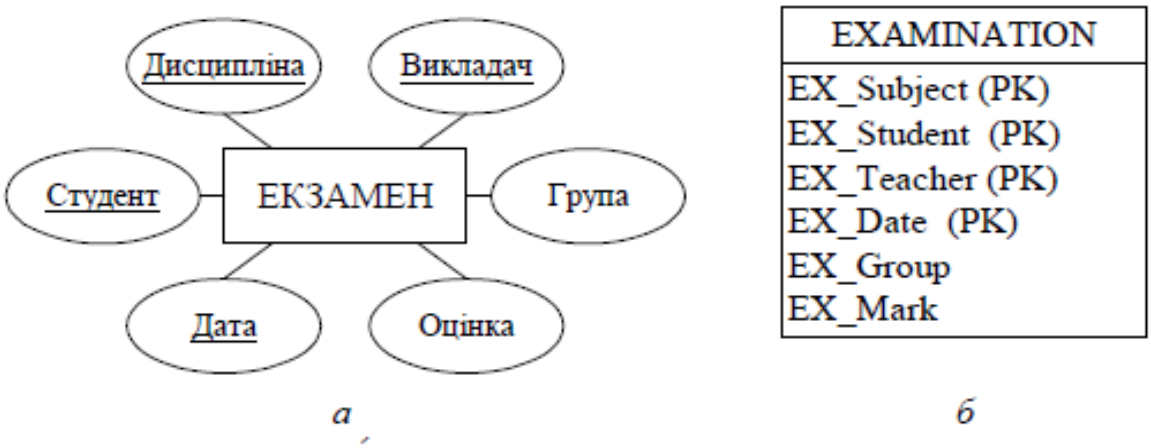
14.5.1. Необхідність перевірки коректності відношень

Створений на попередніх етапах набір відношень логічної моделі БД має бути перевірений на коректність об'єднання атрибутів у кожному відношенні [2]. Перевірка виконується шляхом застосування до кожного відношення процедури послідовної нормалізації. Нормалізація гарантує, що отримана модель не буде мати протиріч і буде мати мінімальну збитковість.

Атрибути в результаті нормалізації будуть згруповані відповідно до існуючих між ними логічних зв'язків.

Для забезпечення коретності логічної моделі при виявленні відношень, що не відповідають вимогам нормалізації, необхідно повернутися на попередні етапи проєктування і перебудувати помилково створені елементи моделі.

Приклад. У результаті проєктування отримано відношення, показане на рис. 14.10 [2].



а *б*

Рис. 14.10. Перетворення сутності *Іспит* (*а*) у відношення *Examination* (*б*)

При дослідженні цього відношення були виявлені такі функціональні залежності:

Дисципліна, Викладач, Студент, Дата – Оцінка, *Студент – Група*. У наведеній схемі існують аномалії і необхідно продовжити нормалізацію. У результаті декомпозиції вихідного відношення буде отримана схема, показана на рис. 14.11 [2].

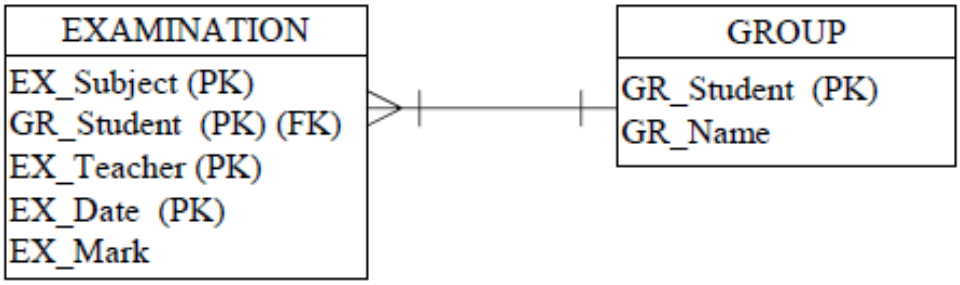


Рис. 14.11. Реляційна схема, що відповідає сутності *Іспит*

14.5.2. Перевірка відповідності відношень вимогам транзакцій користувачів

Перевірка полягає в нанесенні безпосередньо на ER-діаграму всіх шляхів, потрібних для виконання кожної з транзакцій. Якщо так вдається виконати всі транзакції, то перевірка на цьому завершується [2].

Якщо в результаті перевірки будуть виявлені області, що не беруть безпосередньої участі в роботі транзакцій, то можливо їх вилучення з моделі.

14.5.3. Перевірка підтримки цілісності

Обмеження цілісності запобігають появі в БД суперечливих даних [2]. Вирішення цієї проблеми на стадії проєктування полягає в такому:

– наявність обов'язкових і необов'язкових значень даних для атрибутів;

– наявність обмежень для доменів атрибутів (визначення області значень або діапазону значень);

– цілісність сутностей (обов'язкова наявність Primary Kej в кожному відношенні);

– посилкова;

– обмеження предметної галузі (бізнес-правила), що реалізуються як засобами БД, так і на рівні застосувань.

14.5.4. Приклад створення логічної моделі бази даних

У прикладі розд. 12 розроблено концептуальний проєкт бази даних для предметної галузі ЗАКЛАД ВИЩОЇ ОСВІТИ (ЗВО). ER-діаграма відображує всі бізнес-правила, які у свою чергу визначають сутності, атрибути, зв'язки і т. д.

Наступним етапом проєктування бази даних є створення логічної моделі бази даних на основі створеної ER-моделі. Створення логічної моделі бази даних виконується шляхом застосування правил перетворення ER-діаграми в логічну модель (рис. 14.12) [2].

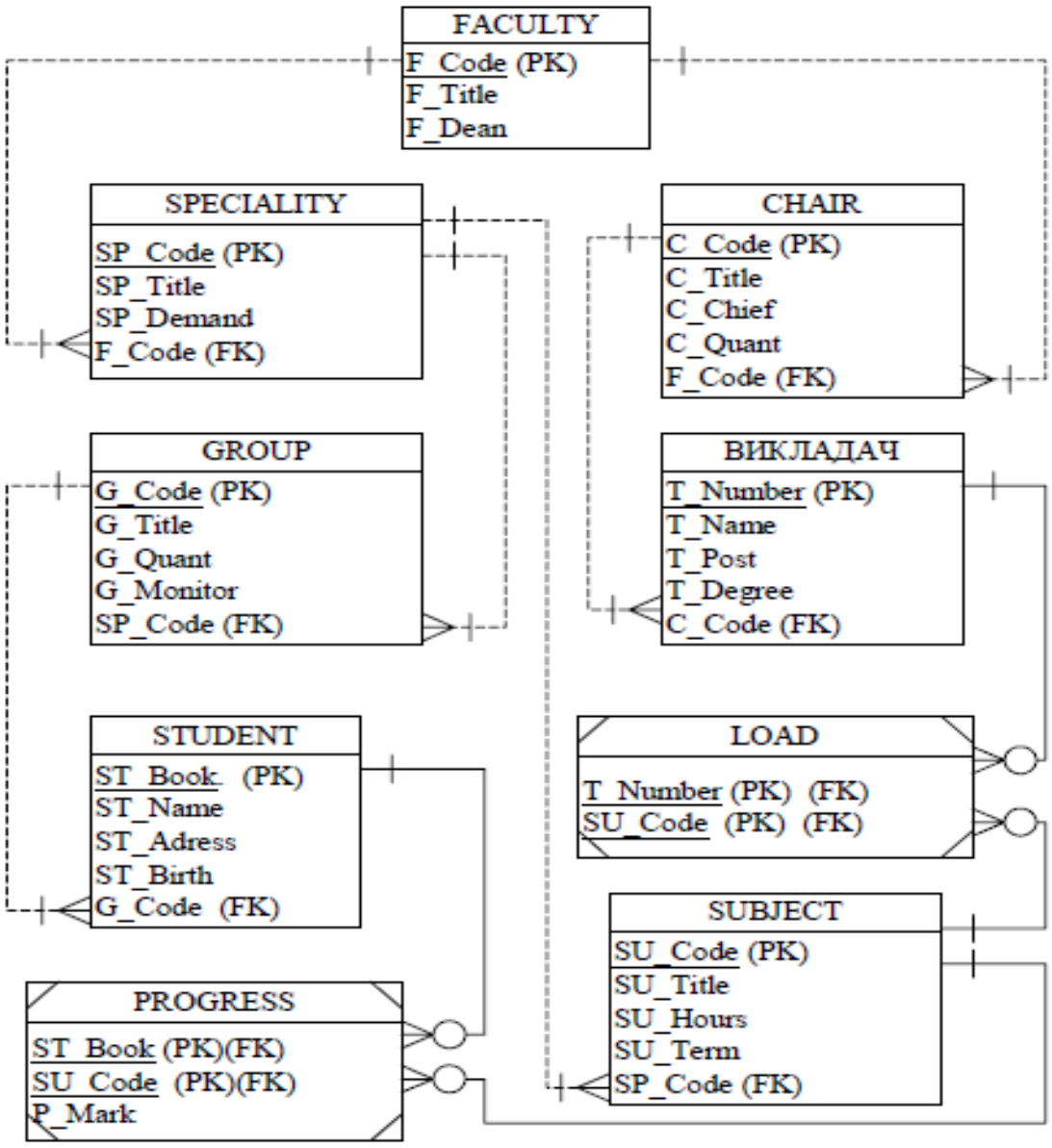


Рис. 14.12. Логічна модель бази даних для предметної галузі ЗАКЛАД ВИЩОЇ ОСВІТИ

Після створення логічної моделі даних реляційна схема аналізується на коректність об'єднання атрибутів в одному відношенні.

Перевірка коректності виконується шляхом застосування послідовної нормалізації до кожного з відношень.

Метою цієї перевірки є отримання гарантій того, що схема бази даних щонайменше знаходиться в третій нормальній формі або нормальній формі Бойса-Кодда. Якщо ця умова не виконується, то необхідно повернутися на попередні етапи проєктування і перебудувати помилково створені фрагменти моделі. Перевірка логічної моделі бази даних ЗВО показує, що реляційна схема знаходиться в четвертій нормальній формі й корегування моделі не потрібно [2].

Після перевірки логічної моделі за допомогою правил нормалізації система аналізується щодо виконання транзакцій користувачів, що задаються на початкових етапах проєктування.

У разі неможливості виконання певних транзакцій необхідне корегування моделі бази даних.

Подальша перевірка моделі вимагає перевірки підтримки цілісності даних. На основі матеріалів прикладу можна тільки визначити, що підтримується посилкова цілісність. Усі інші перевірки, включаючи і перевірку транзакцій користувачів, вимагають більш детально опрацьованого проєкту бази даних.

Контрольні запитання до розділу 14

1. Що створюється для кожної сутності відношення?
2. Навести приклади зв'язків для сутностей.
3. Навести зміст зв'язку «один до одного».
4. Навести зміст зв'язку «один до багатьох».
5. Навести зміст зв'язку «багато до багатьох».
6. Навести зміст інших видів зв'язків.
7. Навести зміст зв'язку «суперклас-підклас».
8. Перевірка відношень за допомогою правил нормалізації.
9. Чим обумовлена необхідність перевірки коректності відношень?
10. У чому полягає перевірка відповідності відношень вимогам транзакцій користувачів?
11. У чому полягає перевірка підтримки цілісності?
12. Навести приклад створення логічної моделі бази даних.

Розділ 15. Елементи мови SQL

У цьому розділі розглядаються елементи мови SQL (Structured Query Language). Поточна версія стандарту мови SQL прийнята в 1992 році (офіційна назва стандарту – Міжнародний стандарт мови баз даних SQL (1992) (International Standart Database Language SQL), неофіційна назва – SQL / 92, або SQL-92, або SQL2). Документ, що описує стандарт, містить понад 600 сторінок. Ми дамо тільки деякі поняття мови [3–6].

Мова SQL стала фактично стандартною мовою доступу до баз даних. Усі СУБД, що претендують на назву «реляційні», реалізують той чи інший діалект SQL. Багато нереляційних систем також мають у наш час засоби доступу до реляційних даних. Метою стандартизації є переносимість додатків між різними СУБД.

Слід зауважити, що зараз жодна система не реалізує стандарт SQL у повному обсязі. Крім того, у всіх діалектах мови є можливості, які не є стандартними. Отже, можна сказати, що кожен діалект – це надмножина деякої підмножини стандарту SQL. Це ускладнює переносимість додатків, розроблених для одних СУБД, в інші СУБД [3–6].

На рис. 15.1 наведено співставлення понять для реляційних баз даних з поняттями, застосовуваними в мові SQL.

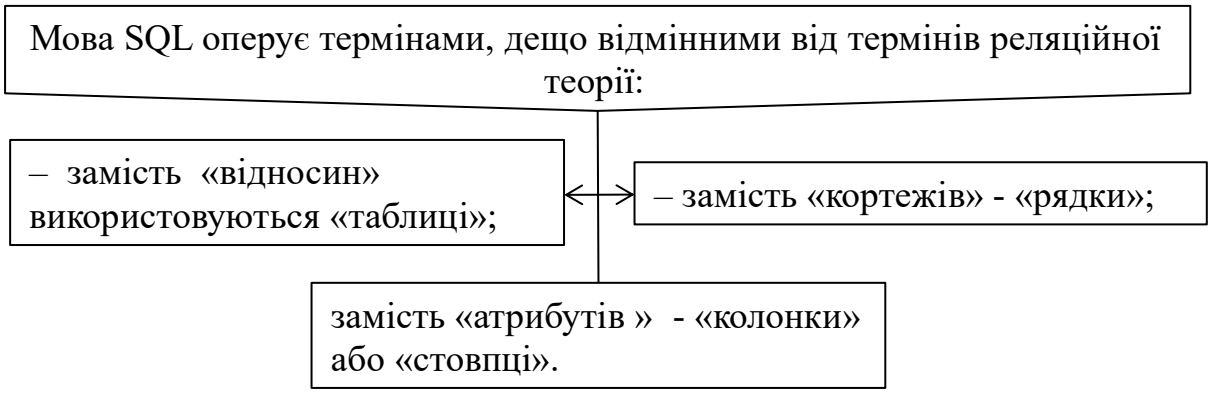


Рис. 15.1. Співставлення понять для реляційних баз даних з поняттями, застосовуваними в мові SQL

Стандарт мови SQL, хоча і заснований на реляційній теорії, у багатьох місцях відходить він неї. Наприклад, відношення в

реляційній моделі даних не допускає наявності однакових кортежів, а таблиці в термінології SQL можуть мати однакові рядки. Є й інші відмінності.

Мова SQL є реляційно повною. Це означає, що будь-який оператор реляційної алгебри може бути виражений відповідним оператором SQL.

15.1. Оператори SQL

Основу мови SQL складають оператори, умовно розбиті на кілька груп за виконуваними функціями [3–6].

Можна виділити такі групи операторів (перераховані не всі оператори SQL):

– оператори DDL (Data Definition Language) – оператори визначення об'єктів бази даних:

• CREATE SCHEMA – створити схему бази даних;
• DROP SHEMA – видалити схему бази даних;
• CREATE TABLE – створити таблицю;
• ALTER TABLE – змінити таблицю;
• DROP TABLE – видалити таблицю;
• CREATE DOMAIN – створити домен;
• ALTER DOMAIN – змінити домен;
• DROP DOMAIN – видалити домен;
• CREATE COLLATION – створити послідовність;
• DROP COLLATION – видалити послідовність;
• CREATE VIEW – створити подання;
• DROP VIEW – видалити подання;

– оператори DML (Data Manipulation Language) – оператори маніпулювання даними:

• SELECT – відібрати рядки з таблиць;
• INSERT – додати рядки до таблиці;
• UPDATE – змінити рядки в таблиці;
• DELETE – видалити рядки в таблиці;
• COMMIT – зафіксувати внесені зміни;
• ROLLBACK – скасувати внесені зміни;

– оператори захисту і керування даними:

• CREATE ASSERTION – створити обмеження;
• DROP ASSERTION – видалити обмеження;

• GRANT – надати привілеї користувачу або додатку на маніпулювання об'єктами;

• REVOKE – скасувати привілеї користувача або додатки.

Крім того, є групи операторів установлення параметрів сеансу, отримання інформації про базу даних, оператори статичного SQL, оператори динамічного SQL.

Найбільш важливими для користувача є оператори маніпулювання даними (DML).

15.2. Приклади використання операторів маніпулювання даними

INSERT – вставлення рядків до таблиці [3–6].

Приклад 1. Вставлення одного рядка до таблиці:

```
INSERT INTO
P (PNUM, PNAME)
VALUES (4, «Іванов»).
```

Приклад 2. Вставлення до таблиці кількох рядків, обраних з іншої таблиці (до таблиці TMP_TABLE вставляються дані про постачальників з таблиці P, мають номери, великі 2):

```
INSERT INTO
TMP_TABLE (PNUM, PNAME)
SELECT PNUM, PNAME
FROM P
WHERE P.PNUM> 2;
```

UPDATE – оновлення рядків у таблиці.

Приклад 3. Оновлення декількох рядків у таблиці:

```
UPDATE P
SET PNAME = «Пушник»
WHERE P.PNUM = 1;
```

DELETE – видалення рядків у таблиці.

Приклад 4. Видалення кількох рядків у таблиці:

```
DELETE FROM P
 WHERE P.PNUM = 1.
```

Приклад 5. Видалення всіх рядків у таблиці:

```
DELETE FROM P.
```

15.3. Приклади використання оператора SELECT

Оператор SELECT є фактично найважливішим для користувача і найскладнішим оператором SQL [3-6]. Він призначений для вибірки даних з таблиць, тобто він, власне, і реалізує одне з основних призначень бази даних – надавати інформацію користувачеві.

Зауваження. Насправді в базах даних можуть бути не тільки постійно збережені таблиці, а також тимчасові таблиці і так звані подання. Подання – це просто збережені в базі даних SELECT-вирази. З точки зору користувачів, подання – це таблиця, яка не зберігається постійно в базі даних, а «виникає» в момент звернення до неї. З точки зору оператора SELECT, і постійно збережені таблиці, і тимчасові таблиці та подання виглядають абсолютно однаково. Звичайно при реальному виконанні оператора SELECT системою враховуються відмінності між збереженими таблицями і поданнями, але ці відмінності приховані від користувача [3–6].

Результатом виконання оператора SELECT завжди є таблиця. Отже, за результатами дій оператор SELECT схожий на операторів реляційної алгебри. Будь-який оператор реляційної алгебри може бути виражений відповідним чином сформульованим оператором SELECT. Складність оператора SELECT визначається тим, що він містить усі можливості реляційної алгебри, а також додаткові можливості, яких у реляційній алгебрі нема.

Відбір даних з однієї таблиці [3–6]. **Приклад 6.** Вибрати всі дані з таблиці постачальників (ключові слова SELECT FROM) [3–6]:

```
SELECT *
FROM P.
```

Зауваження. У результаті отримаємо нову таблицю, що містить повну копію даних з вихідної таблиці P.

Приклад 7. Вибрати всі рядки з таблиці постачальників, що задовольняють деяку умову (ключове слово WHERE):

```
SELECT *
FROM P
WHERE P.PNUM> 2.
```

Зауваження. Як умову в розділі WHERE можна застосовувати складні логічні вирази, що використовують поля таблиць, константи, відношення (>, <, = і т. д.), дужки, сполучники AND і OR, заперечення NOT.

Приклад 8. Вибрати деякі колонки з вихідної таблиці:

```
SELECT P.NAME
FROM P.
```

Зауваження 1. У результаті отримаємо таблицю з однією колонкою, що містить усі найменування постачальників.

Зауваження 2. Якщо у вихідній таблиці були присутні кілька постачальників з різними номерами, але однаковими найменуваннями, то в результуючій таблиці будуть рядки з повтореннями – дублікати рядків автоматично не відкидаються.

Приклад 9. Вибрати деякі колонки з вихідної таблиці, видаливши з результату повторювані рядки (ключове слово DISTINCT):

```
SELECT DISTINCT P.NAME
FROM P.
```

Зауваження. Використання ключового слова DISTINCT призводить до того, що в результуючій таблиці будуть видалені всі повторювані рядки.

Контрольні запитання до розділу 15

1. Що забезпечує оператор маніпулювання даними SELECT?
2. Що забезпечує оператор маніпулювання даними INSERT?
3. Що забезпечує оператор маніпулювання даними UPDATE?
4. Що забезпечує оператор маніпулювання даними DELETE?
5. Що забезпечує оператор маніпулювання даними COMMIT?
6. Що забезпечує оператор маніпулювання даними ROLLBACK?

7. Що забезпечує оператор захисту і керування даними CREATE ASSERTION?

8. Що забезпечує оператор захисту і керування даними DROP ASSERTION?

9. Що забезпечує оператор захисту і керування даними GRANT?

10. Що забезпечує оператор захисту і керування даними REVOKE?

Бібліографічний список

1. Історія розвитку баз даних. URL: https: helpiks.org/2-9486.html.

2. Гайна Г. А. Основи проектування баз даних: навч. посіб. Київ: КНУБА, 2005. 204 с.

3. Класифікація моделей даних. URL: https://helpiks.org/2-9488.html.

4. Реляційна алгебра. URL: https://rdb.dp.ua/uk/chapter_05.

5. Реляційна алгебра. URL: http://elenagavrile.narod.ru/dm/Lekccya_6.pdf.

6. Реляційна алгебра. URL: https://stud.com.ua/93791/informatika/relyatsiyna_algebra#43.

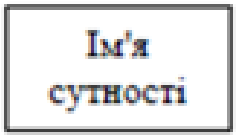
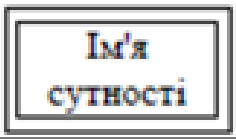
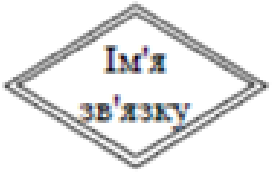
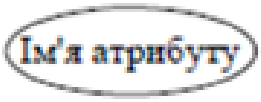
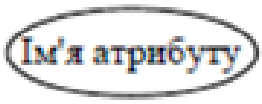
7. Пасічник В. В., Реаніченко В. А. Організація баз даних та знань. Київ: Видавнича група ВНУ, 2006. 384 с.

Додаток 1

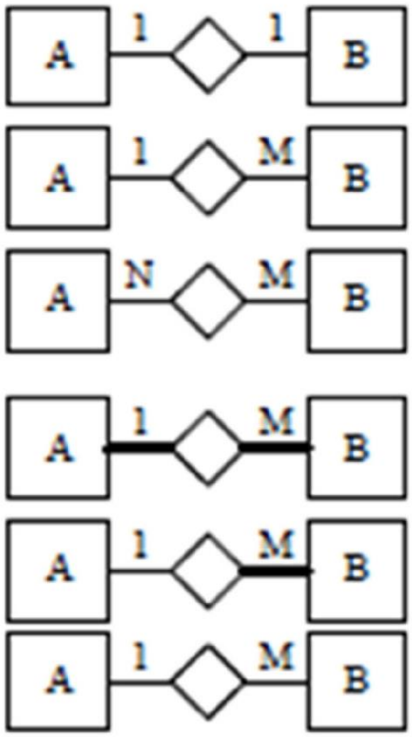
СИСТЕМИ ПОЗНАЧЕНЬ В ER-МОДЕЛЯХ [2]

Таблиця Д.1.1

Модель Чена

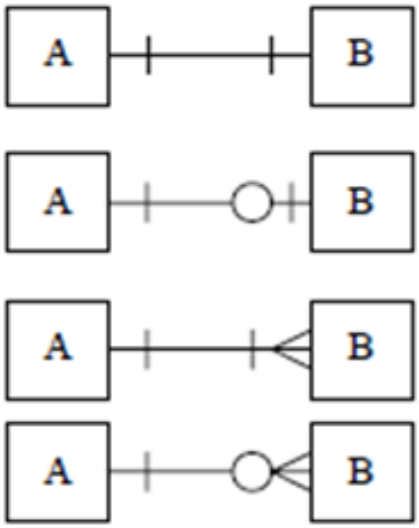
Графічне подання	Зміст
1	2
Сутності	
	Сильна сутність
	Слабка сутність
Зв'язки	
	Зв'язок
	Зв'язок, який пов'язаний зі слабкою сутністю
Атрибути	
	Атрибут
	Атрибут первинного ключа
	Багатозначний атрибут
	Довільний атрибут

Продовження табл. Д.1.1

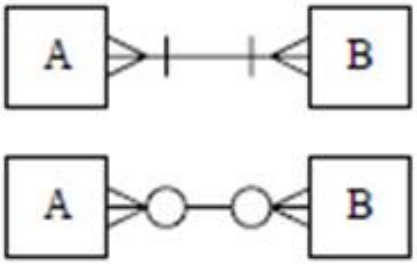
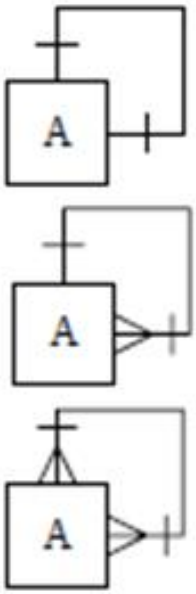
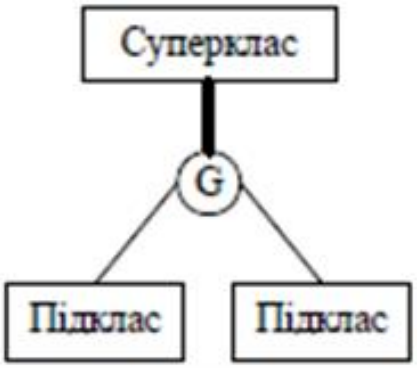
1	2
Види зв'язків 	Зв'язок один до одного , 1:1 Зв'язок один до багатьох , 1:M Зв'язок багато до багатьох , N:M Зв'язок 1:M, з обов'язковою участю сутностей A і B Зв'язок 1:M, з обов'язковою участю сутності B і необов'язковою участю сутності A Зв'язок 1:M, з необов'язковою участю сутностей A і B
	Рекурсивний зв'язок
	Уточнення/узагальнення Літера d означає, що зв'язок не перетинається , літера o означає, що зв'язок може перетинатися. Подвійна лінія означає обов'язкову участь, а одинарна лінія означає необов'язкову участь

Таблиця Д.1.2

Модель «пташина лапка»

Графічне представлення	Зміст
Сутності Ім'я сутності Ім'я атрибуту 1 Ім'я атрибуту 2 ..... {Ім'я атрибут N} Ім'я сутності Ім'я атрибуту 1 Ім'я атрибуту 2 ..... Ім'я атрибут N	Сильна сутність Ім'я атрибуту - атрибут Ім'я атрибуту - атрибут первинного ключа {Ім'я атрибута} - багатозначний атрибут Слабка сутність
Зв'язок Ім'я зв'язку Ім'я зв'язку 	Сильний (ідентифікуючий) зв'язок - зв'язок сильної і слабкої сутностей Слабкий (неідентифікуючий) зв'язок - зв'язок сильних сутностей Зв'язок один до одного з обов'язковою участю сутностей A і B Зв'язок один до одного з необов'язковою участю сутності B і обов'язковою участю сутності A Зв'язок один до багатьох з обов'язковою участю сутностей A і B Зв'язок один до багатьох з необов'язковою участю сутності B і обов'язковою участю сутності A

Продовження табл. Д.1.2

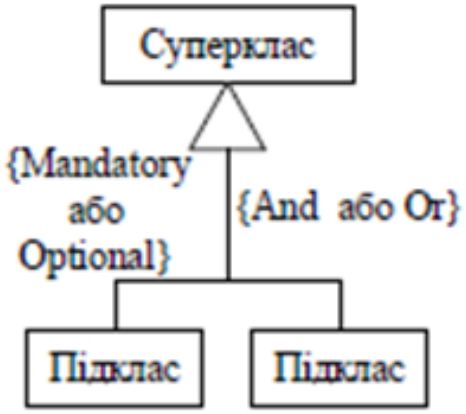
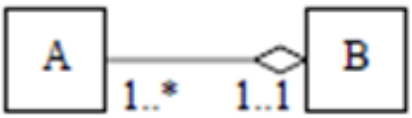
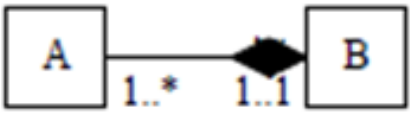
1	2
Зв'язок 	Зв'язок багато до багатьох з обов'язковою участю сутностей A і B Зв'язок багато до багатьох з необов'язковою участю сутностей A і B
Рекурсивний зв'язок 	Рекурсивний зв'язок один до одного Рекурсивний зв'язок один до багатьох Рекурсивний зв'язок багато до багатьох
	Уточнення/узагальнення Літера G означає, що зв'язок не перетинається , літера Gs означає, що зв'язок може перетинатися. Подвійна лінія означає обов'язкову участь, а одинарна лінія означає необов'язкову участь

Таблиця Д.1.3

Модель UML

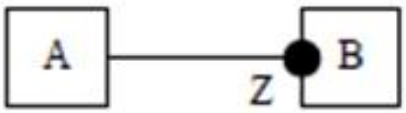
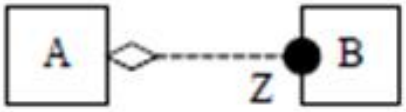
Графічне подання	Зміст
1	2
Сутності Ім'я сутності *Ім'я атрибута 1 (PK)* *Ім'я атрибута 2* *Ім'я атрибута 21* *Ім'я атрибута 22* *Ім'я атрибут 3[1..M]* */Ім'я атрибута 4* *Ім'я атрибута 5* ..... *Ім'я атрибута N*	*Сутність* *Ім'я атрибута* - атрибут *Ім'я атрибута (PK)*- атрибут первинного ключа *Ім'я атрибута [1..M]* - багатозначний атрибут */Ім'я атрибута* - атрибут, що обчислюється *Ім'я атрибута 1* *Ім'я атрибута 11* *Ім'я атрибута 12* ..... *Ім'я атрибута 1N* - складний атрибут
Зв'язок Ім'я зв'язку ▶ A — 1..1 0..1 — B A — 0..1 0..* — B A — 0..* 1..* — B	Ім'я зв'язку Зв'язок один до одного з обов'язковою участю сутності A і необов'язковою участю сутності B Зв'язок один до багатьох з необов'язковою участю сутностей A і B Зв'язок багато до багатьох з обов'язковою участю сутності B і необов'язковою участю сутності A

Продовження табл. Д.1.3

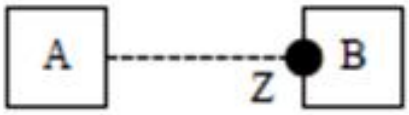
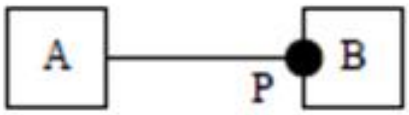
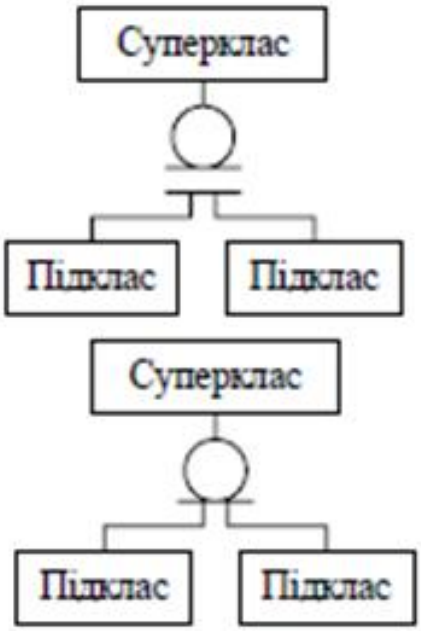
1	2
Розширена модель сутність зв'язок Суперклас/підклас  Агрегування  Композиція  	Уточнення/узагальнення. *Or* означає, що зв'язок не перетинається, а *And* означає, що зв'язок може перетинатися. *Mandatory* означає обов'язкову участь, а *Optional* означає необов'язкову участь Агрегування. На сутність ціле вказує пустий ромб Композиція. На сутність ціле вказує заповнений ромб

Таблиця Д.1.4

Модель IDEF1X

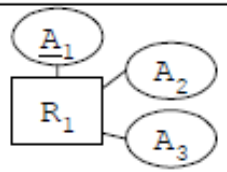
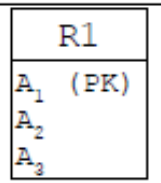
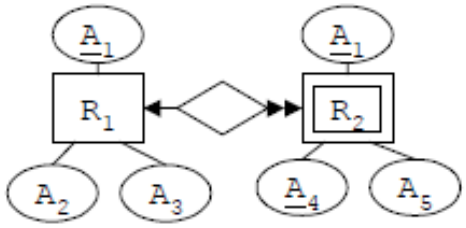
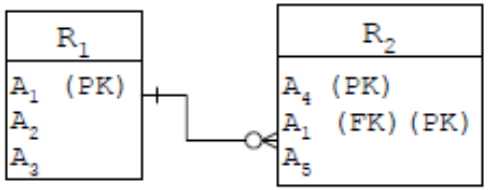
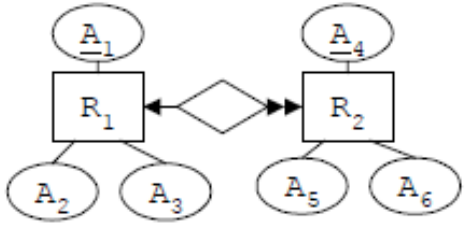
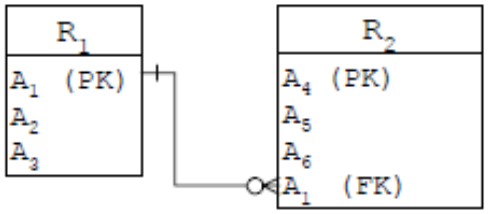
Графічне подання	Зміст
1	2
Сутності Ім'я сутності Ім'я атрибута Ім'я атрибута 1 Ім'я атрибута 2 ..... Ім'я атрибута N Ім'я сутності Ім'я атрибута Ім'я атрибута 1 Ім'я атрибута 2 ..... Ім'я атрибута N	Сильна сутність. Ім'я атрибута - атрибут Ім'я атрибута - атрибут первинного ключа Слабка сутність
Зв'язки Ім'я зв'язку Ім'я зв'язку  	Сильний (ідентифікуючий) зв'язок - зв'язок сильної і слабкої сутностей Слабкий (неідентифікуючий) зв'язок - зв'язок сильних сутностей Зв'язок один до одного . Кожен запис з B повинен бути зв'язаний тільки з одним записом A. Запис з A може бути зв'язаний тільки з одним записом B (Z - 0..1) Зв'язок один до одного . Запис з A може бути зв'язаний тільки з одним записом з B. Запис з B може бути зв'язаний тільки з одним записом з A, записи в A можуть приймати значення NULL

Продовження табл. Д.1.4

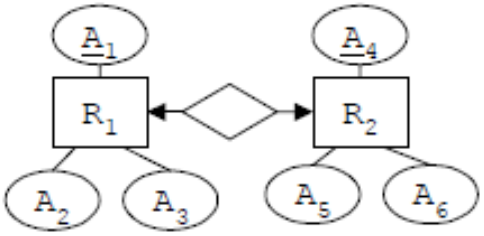
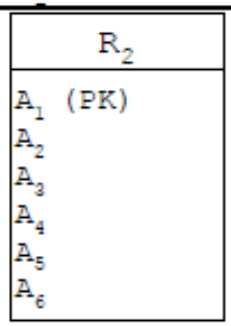
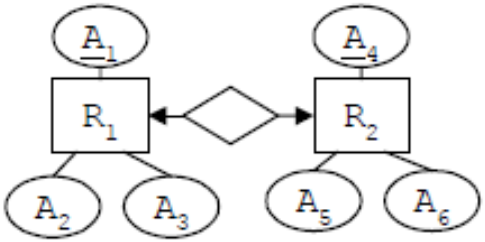
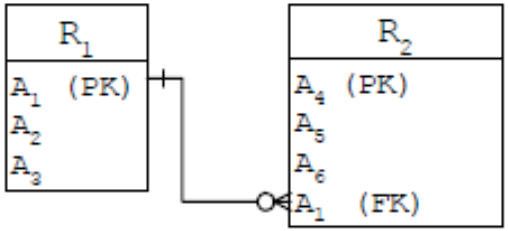
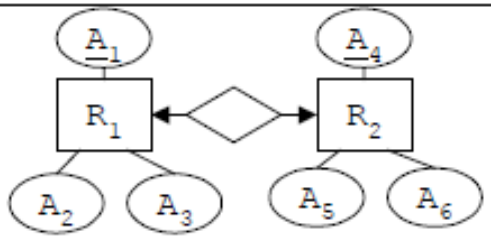
1	2
Зв'язки 	Зв'язок один до одного . Запис з А може бути зв'язаний тільки з одним записом з В. Запис з В може бути зв'язаний тільки з одним записом з А, записи в А набувають значення NOT NULL
	Зв'язок один до багатьох . Запис з А повинен бути зв'язаним з одним або декількома записами з В. Кожен запис В повинен бути зв'язаний тільки з одним записом з А (P - 1..*)
	Зв'язок багато до багатьох з обов'язковою участю сутностей А і В
Розширена модель сутність-зв'язок Суперклас/підклас 	Уточнення/узагальнення. Обов'язкова участь Уточнення/узагальнення. Необов'язкова участь

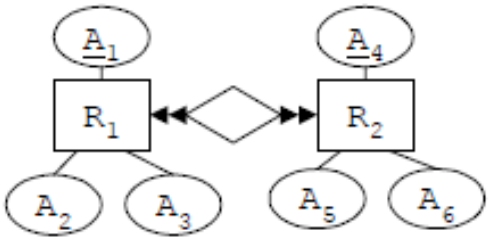
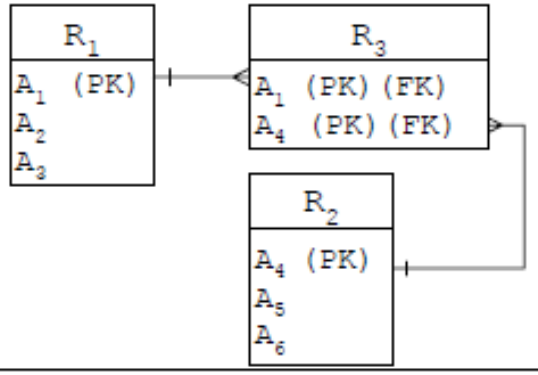
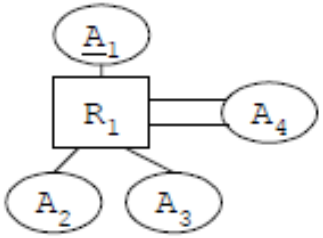
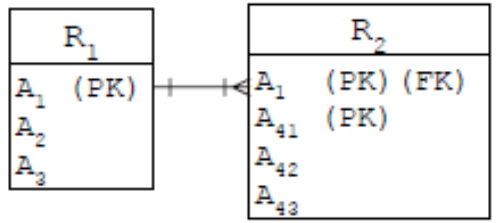
Таблиця Д.1.5

Правила перетворення сутностей, атрибутів і зв'язків у відношення

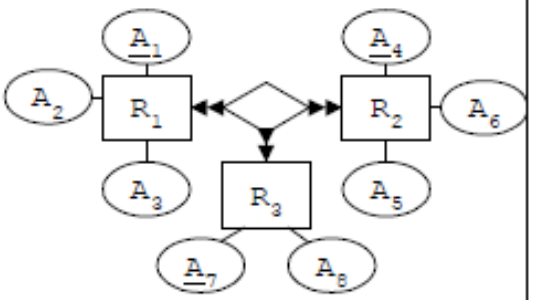
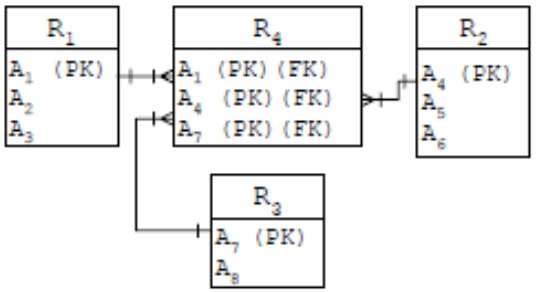
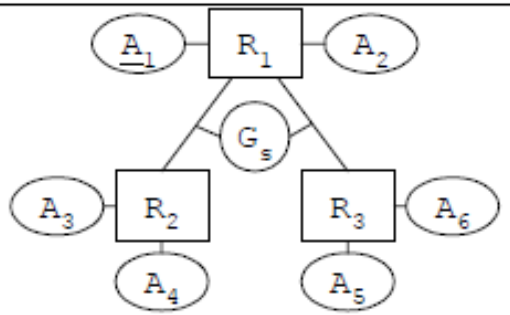
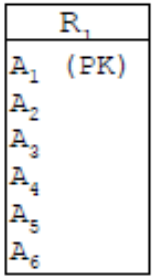
Сутність /зв'язок	Правило перетворення	Приклад	
		ER-модель	Логічна модель
Сильна сутність	Створення відношення, яке включає всі прості атрибути		
Слабка сутність	Створення відношення, яке включає всі прості атрибути. Після перетворення зв'язку з сутністю-володарем необхідно визначити первинний ключ		
Зв'язок 1:M	Передача первинного ключа сутності один на відношення багато як зовнішнього ключа. На бік багато передаються також всі атрибути зв'язку		

Продовження табл. Д.1.5

Сутність /зв'язок	Правило перетворення	Приклад	
		ER-модель	Логічна модель
Зв'язок 1:1	Обов'язкова участь обох боків зв'язку. Сутності об'єднуються в одне відношення		
	Обов'язкова участь одного боку зв'язку. Передача первинного ключа на обов'язковий бік для використання як зовнішнього ключа у відношенні, що являє не обов'язковий бік		
	Необов'язкова участь обох боків зв'язку. У разі відсутності додаткової інформації вибір буде довільним		Можливі три варіанти логчної моделі (з двома відношеннями і з трьома відношеннями)

Сутність /зв'язок	Правило перетворення	Приклад	
		ER-модель	Логічна модель
Зв'язок N:M	Створення відношення, яке являє зв'язок, і включення всіх атрибутів зв'язку. Передача в нове відношення копії первинного ключа з кожної сутності-володаря для використання як зовнішнього ключа		
Багато-значний атрибут	Створення відношення, яке являє багатозначний атрибут, і передає в нове відношення копії первинного ключа із сутності-володаря для використання як зовнішнього ключа		

Продовження табл. Д.1.5

Сутність /зв'язок	Правило перетворення	Приклад	
		ER-модель	Логічна модель
Склад-ний зв'язок	Створення відношення, яке являє зв'язок, і включення всіх атрибутів зв'язку. Передача в нове відношення копії первинного ключа з кожної сутності-володаря для використання як зовнішнього ключа		
Зв'язок клас–підклас	Обов'язкова участь зв'язку суперклас/підклас (Mandatory), дозволяється перетин підкласів (And)		

Навчальний посібник

Доценко Сергій Ілліч

ОРГАНІЗАЦІЯ ТА СИСТЕМИ КЕРУВАННЯ БАЗАМИ ДАНИХ

Відповідальний за випуск Доценко С. І.

Редактор Ібрагімова Н. В.

Підписано до друку 29.03.2021 р.
Умовн. друк. арк. 8,75. Тираж . Замовлення № .

Видавець та виготовлювач Український державний університет
залізничного транспорту,
61050, Харків-50, майдан Фейєрбаха,7.
Свідоцтво суб'єкта видавничої справи ДК № 6100 від 21.03.2018 р.