МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БУДІВНИЦТВА І АРХІТЕКТУРИ

Г.А.Гайна

ОСНОВИ ПРОЕКТУВАННЯ БАЗ ДАНИХ

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, що навчаються за напрямом підготовки 0804 "Комп'ютерні науки"

Київ 2005

УДК 681.3.07
ББК 32.973.26-018.2.75
Г12

Рецензенти: *О.С. Городецький,* доктор технічних наук, професор, заступник директора НДІАСБ
С.М. Гриша, доктор технічних наук, професор кафедри автоматизованих систем обробки інформації та управління НТУУ "КПІ"

Рекомендовано Міністерством освіти і науки України як навчальний посібник для студентів вищих навчальних закладів, що навчаються за напрямом підготовки "Комп'ютерні науки" (лист Міністерства освіти і науки України № 14/18.2–2622 від 23 листопада 2005 р.).

Гайна Г.А.
Г12 Основи проектування баз даних: Навчальний посібник. – К.: КНУБА, 2005. – 204 с.

ISBN 966-627-117-6

Розглянуто головні концепції проектування і побудови баз даних, основні моделі баз даних, архітектури СУБД, сучасні напрямки розвитку технологій баз даних, питання, які пов'язані з експлуатацією баз даних. Всі головні положення книги розглядаються з використанням навчальних прикладів і ілюструються графічними матеріалами.

Зміст навчального посібника відповідає курсу лекцій, які автор викладає в КНУБА.

Призначений для студентів усіх спеціальностей напряму підготовки "Комп'ютерні науки".

УДК 681.3.07
ББК 32.973.26-018.2.75

ISBN 966-627-117-6

ВСТУП

Впровадження інформаційних технологій зробило можливим значно підвищити ефективність роботи підприємств, організацій, проведення наукових досліджень, підвищити якість роботи в різних галузях освіти, науки, культури. Широке практичне використання технологій баз даних обумовлено значними досягненнями у цій галузі провідними комп'ютерними компаніями світу, потребою суспільства в ефективних засобах зберігання і обробки інформації. Отже, зростає потреба у фахівцях, здатних розробляти і застосовувати системи баз даних.

Даний навчальний посібник підготовлено за матеріалами лекційних курсів щодо баз даних, які автор протягом багатьох років викладає в Київському національному університеті будівництва і архітектури студентам, що навчаються за напрямом підготовки "комп'ютерні науки". Навчальний посібник відповідає вимогам стандартів до дисципліни "Організація баз даних і знань" для всіх комп'ютерних спеціальностей.

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

Матеріал книги поділено на чотири частини і додатки:

- *Частина 1.* Основні відомості.
- *Частина 2.* Проектування баз даних.
- *Частина 3*. Сучасні технології баз даних.
- *Частина 4.* Експлуатація баз даних.
- *Додатки*.

Кожна з частин містить декілька глав, які мають велику кількість прикладів, а також контрольні запитання для перевірки засвоєння матеріалу.

У *першій частині* (глави 1-3) розглядаються основні поняття баз даних, аналізується сучасний стан технологій баз даних, визначається трирівнева архітектура баз даних. Наведено загальні характеристики моделей представлення даних, розглянуто реляційну модель даних, яка є основою переважної більшості сучасних систем керування базами даних. Розглянуто механізми маніпулювання даними.

У *другій частині* (глави 4-9) розглядаються сучасні методології проектування баз даних. Наводяться етапи життєвого циклу інформаційної системи, аналізується зміст концептуального, логічного і фізичного проектування бази даних. Детально розглядаються етапи побудови діаграми "сутність-зв'язок", розширеної діаграми "сутність-зв'язок". Значна частина матеріалу присвячена основам побудови логічної моделі бази даних, нормалізації відношень методом нормальних форм. Визначаються принципи автоматизації проектування баз даних на основі сучасних інструментальних засобів.

У *третій частині* (глави 10-12) розглядаються сучасні напрямки застосування технологій баз даних. Описуються особливості побудови систем розподіленої обробки даних, розглядаються моделі клієнт-сервер, розподілені бази даних, організація багатокористувацької роботи з базою даних. Визначаються принципи побудови об'єктно-орієнтованих баз даних. Наведено склад, структуру, моделі інформаційних сховищ.

У *четвертій частині* (глави 13-14) розглядаються питання експлуатації баз даних. Описуються задачі адміністрування базами даних, організація відновлення баз даних. Наводяться принципи організації захисту інформації в базах даних.

Додатки мають довідковий характер і в них у стислій формі розглядаються особливості позначень діаграм "сутність-зв'язок" в різних інформаційних системах і правила побудови логічної моделі бази даних.

У *заключенні* стисло розглядаються перспективи розвитку сучасних систем баз даних.

У *словнику* наводяться визначення основних термінів з технологій баз даних.

Навчальний посібник написаний на основі аналізу, систематизації і узагальнення великої кількості різноманітних матеріалів і власних робіт автора. Також на структуру і зміст книги вплинув значний викладацький досвід автора та його багаторічне спілкування зі студентами під час проведення занять і бажання у стислій та доступній формі розкрити питання, що стосуються баз даних, показати шляхи вирішення проблем зберігання інформації.

Оскільки у викладанні матеріалу автор був обмежений вимогами щодо обсягу навчального посібника, певна частина матеріалу залишилась поза розглядом. У кінці книги наведено список рекомендованої літератури (в тому числі й автора), де всебічно розглядаються питання, що пов'язані з базами даних і які не увійшли в книгу.

Книга орієнтована на використання в навчальному процесі, тому автор намагався до дохідливого викладання матеріалу, строгому визначенню понять, чіткій класифікації сучасних технологій баз даних з зазначенням їх переваг і недоліків. В книзі широко застосовується ілюстративний матеріал, приводиться багато прикладів, що на думку автора повинно сприяти підвищенню рівня вивчення дисципліни. По кожному розділу приводяться контрольні запитання.

В першу чергу книга адресована студентам вищих навчальних закладів, які вивчають дисципліни: "Організація баз даних і знань", "Інформаційне забезпечення", "Інформаційні системи", а також може бути корисною розробникам баз даних, прикладним програмістам, фахівцям у галузі інформаційних технологій, всім бажаючим підвищити свої знання в галузі інформаційних технологій.

Частина 1. ОСНОВНІ ВІДОМОСТІ

Глава 1. ВСТУП ДО БАЗ ДАНИХ

1.1. Технології баз даних

Виникнення технологій баз даних припадає на початок 60-х років. Їх швидкому розвитку сприяли потреби в обробці інформації, досягнення в суміжних областях інформаційних технологій таких, як операційні системи, мови програмування, технічне забезпечення. Спочатку зароджувалися певні ідеї щодо управління 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.

Інформаційні задачі на відміну від обчислювальних мають такі особливості:

- збереження даних складної структури;
- відносно прості алгоритми обробки;
- великі обсяги оброблюваної інформації.

Інформаційна система виконує функції збирання, зберігання, розповсюдження і обробки інформації. Під *інформацією* розуміють будь-які відомості про будь-яку подію, сутність, процес і т.ін., які є об'єктом певних операцій: передачі, перетворення, зберігання або використання. *Дані* можна визначити, як інформацію зафіксовану у певній формі, яка придатна для подальшої обробки, зберігання і передачі (рис. 1.1). *Предметна область* – область застосування конкретної бази даних.

Інформації відповідає *інфологічне представлення*, яке розглядає питання пов'язані зі змістом інформації, незалежно від представлення її в пам'яті комп'ютера. *Даталогічне представлення* розглядає питання представлення даних в пам'яті комп'ютера.

Функції управління інформацією в інформаційних системах виконують системи управління базами даних.

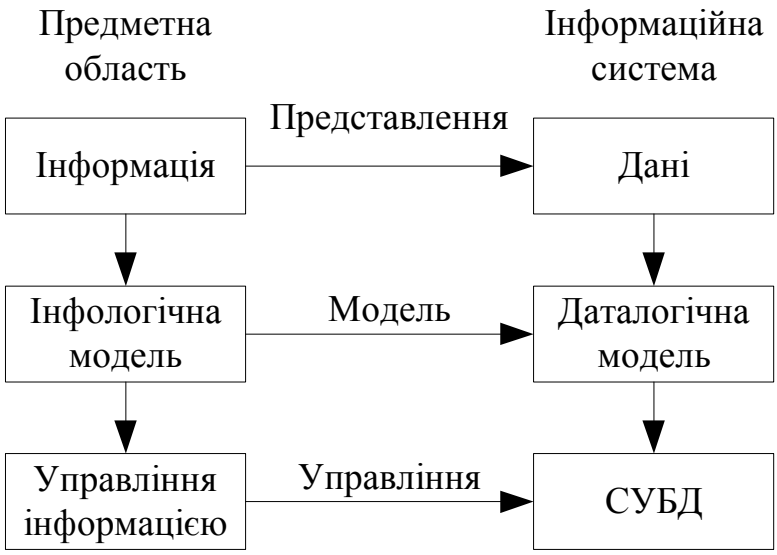


Рис. 1.1. Зв'язок між інформацією і даними

Спочатку для збереження даних застосовувалися *файлові системи* (рис. 1.2). Для цих систем були характерні такі особливості:

- структура запису файла даних була відома тільки прикладній програмі, яка з ним працювала, а система управління файлами її не знала; кожна програма, яка працювала з файлом даних, повинна була мати у себе структуру даних, яка відповідала цьому файлу (така ситуація характеризувалась, як *залежність програм від даних*);
- система управління файлами повинна була забезпечити авторизацію доступу до файлів для різних користувачів, але були відсутні централізовані методи управління доступом до інформації;
- для організації паралельної роботи користувачів з даними необхідне узгоджене управління, а система управління файлами при цьому працювала надто повільно.

Для подолання цих недоліків практично паралельно почалися роботи над *ієрархічними* і *мережними* базами даних (БД) і над відповідними системами управління цими БД – СУБД. В основі цих БД лежали відповідні моделі даних: ієрархічні і мережні (рис. 1.2).

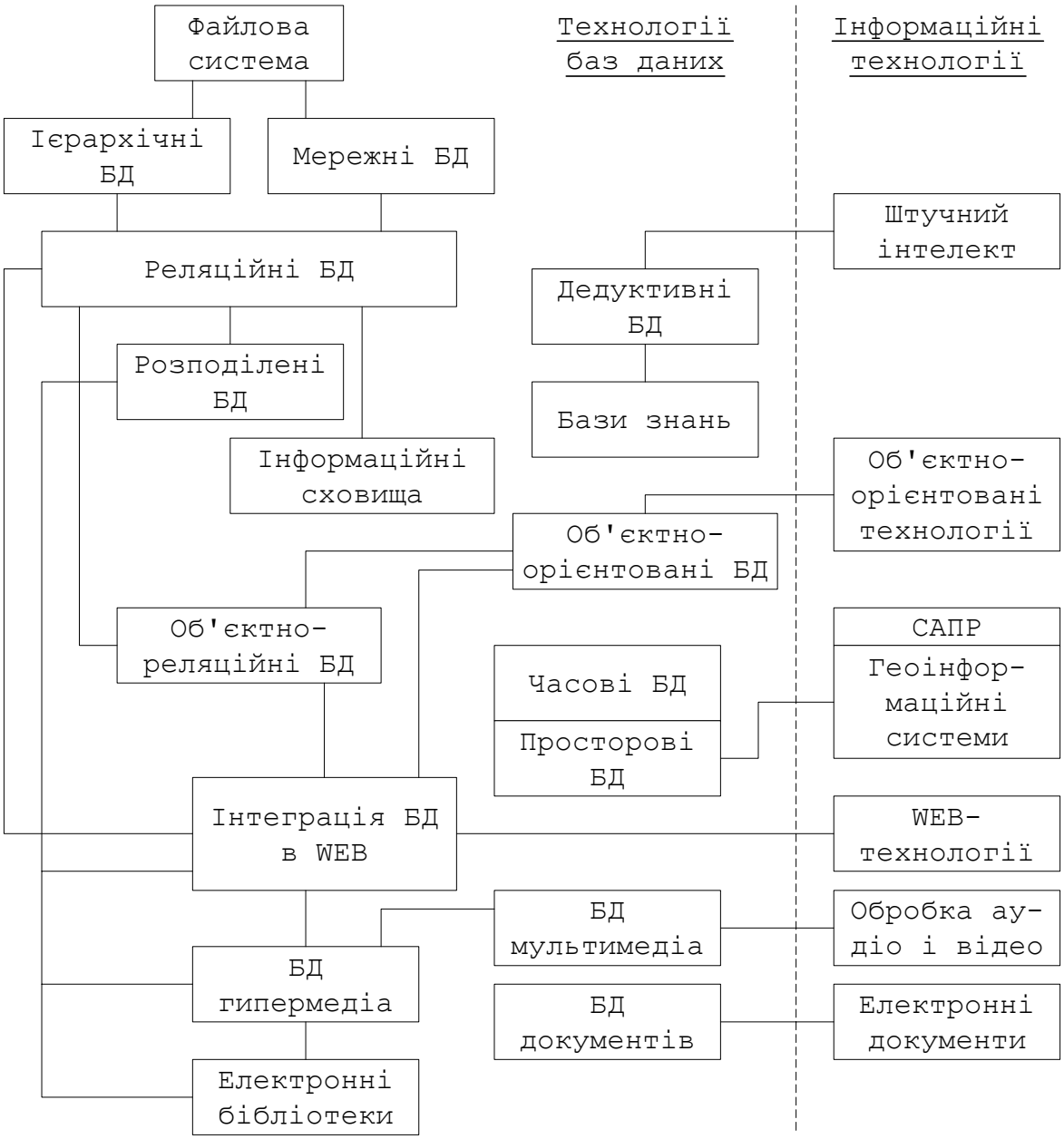


Рис. 1.2. Етапи розвитку баз даних

Ці моделі належать до теоретико-графових моделей. До переваг цих БД можна віднести ефективне використання пам'яті

комп'ютера, швидке виконання основних операцій над даними. Недоліками цих БД є таке:

- труднощі при обробці інформації з достатньо складними зв'язками;
- складність розуміння змісту звичайним користувачем, залежність від фізичної реалізації.

На початку 70-х років були сформовані теоретичні основи сучасних технологій БД і остаточно сформувався самостійний напрям інформаційних технологій – наука о базах даних. Головною подією цього періоду стало виникнення *реляційних* баз даних. В основі реляційної моделі лежить поняття відношення. Реляційна модель є простою і зрозумілою, а її фізичне представлення добре реалізується на комп'ютері. До недоліків реляційних моделей можна віднести складність реалізації ієрархічних і мережних зв'язків.

У 70-х роках почали формуватися підходи в БД, які пов'язані з використанням апарата логіки в якості моделі даних. Ці роботи привели до створення *дедуктивних* БД. Розвиток цього напряму дозволяє створювати *бази знань*.

У 80-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 застосувань, систем, які оперують просторовими даними і т.ін. В середині 80-х років успіхів досягло об'єктно-орієнтоване програмування. Під його впливом і в зв'язку з необхідністю реалізації нетрадиційних застосувань, вимоги яких погано узгоджуються з можливостями реляційних систем, почались роботи з практичної реалізації *об'єктно-орієнтованих* БД. Логічна структура об'єктно-орієнтованої БД зовні схожа на структуру ієрархічної БД, але вона доповнена об'єктно-орієнтованими механізмами. Об'єктно-орієнтовані БД у порівнянні з реляційними БД мають можливість відображати інформацію о складних взаємозв'язках об'єктів, визначати функції обробки окремих записів. До недоліків об'єктно-орієнтованих моделей

належить висока понятійна складність, низька швидкість виконання запитів, незручність обробки даних.

Бажання розробників реляційних БД зберегти лідируючі позиції і підвищити ефективність реляційної моделі, привели до розробки об'єктно-реляційної моделі, на основі якої почали розроблятися *об'єктно-реляційні* БД. В цих моделях припускається застосування багатозначних полів – полів, які складаються з підзначень. Ці БД дозволяють забезпечити високу наглядність представлення інформації і підвищити ефективність її обробки. До недоліків об'єктно-реляційних моделей можна віднести складність рішення забезпечення цілісності і несуперечливості даних.

На початку 90-х років почало збільшуватися значення інформаційного забезпечення систем підтримки прийняття рішень. Це привело до необхідності створення багатомірних СУБД і на їх основі розробки *інформаційних сховищ*. Ці системи використовують історичну інформацію, яка представляється в агрегованому вигляді. На основі цієї інформації виконується аналітична обробка, прогнозування даних, а також інтелектуальний аналіз даних.

В цей час також велика увага приділялася розвитку розподілених систем. Успіхи в розробці комп'ютерних мереж стимулювали дослідження в технологіях *розподілених* БД, були розроблені архітектурні концепції *клієнт – сервер*.

Одним з найбільших досягнень 90-х років в області інформаційних технологій стало створення відкритої глобальної розподіленої неоднорідної гіпермедійної інформаційної системи, яка використовує комунікаційну мережу Internet. Ця система отримала назву WWW (Web). З самого початку виконувались спроби інтегрувати системи БД у Web. Одним з напрямів роботи є *інтеграція структурованих даних БД і слабкоструктурованих даних Web*, проводяться роботи зі створення БД на мові XML.

У даний час в багатьох комп'ютерних компаніях здійснюються роботи зі створення *цифрових бібліотек* –

інформаційних систем, які призначені для зберігання, пошуку, обробки, аналізу і розповсюдження інформаційних ресурсів різної природи – структурованих і слабкоструктурованих даних, мультимедійної інформації, повнотекстових документів. Для створення таких систем використовуються Web-технології, розробляються методи інтеграції неоднорідних ресурсів, розпізнавання і пошуку інформаційних ресурсів.

1.2. Компоненти банків даних

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

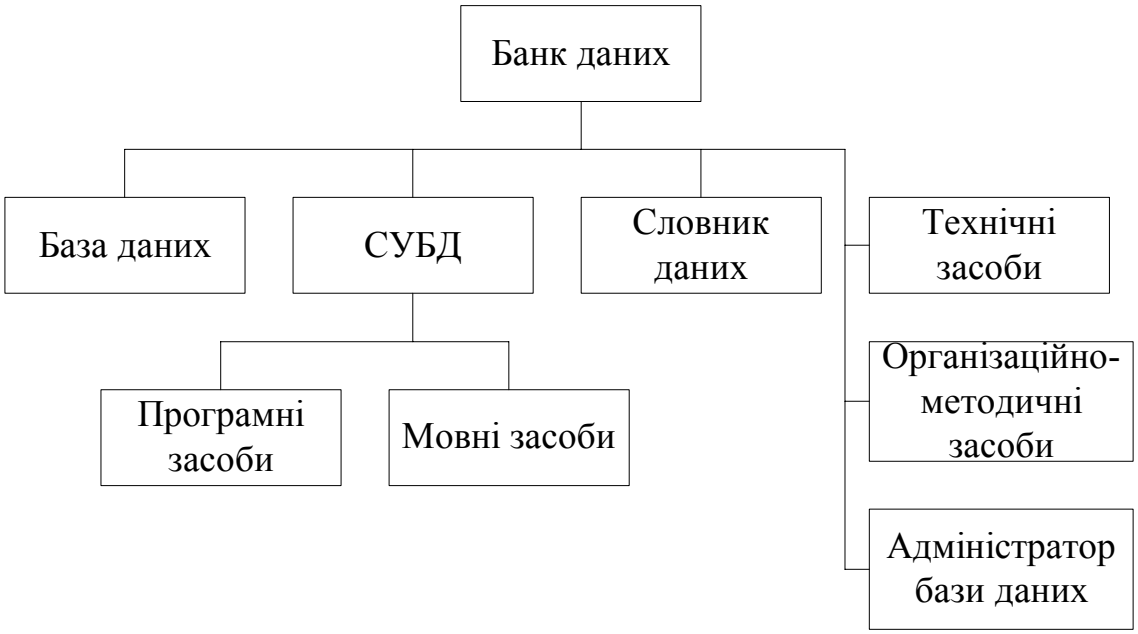


Рис. 1.3. Банк даних

База даних – пойменована сукупність взаємозв'язаних даних, які знаходяться під управлінням СУБД. В 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 повинні бути несуперечливі;
– *безпека* означає, що виконується захист даних від санкціонованого і несанкціонованого доступу;
– *відновленість* означає можливість відновлення БД після збоїв роботи системи.

Система управління базами даних (СУБД) – сукупність мовних і програмних засобів, які призначені для створення, ведення і сумісного використання БД багатьма користувачами.

До головних функцій СУБД належать такі:
– управління даними у зовнішній пам'яті і буферами оперативної пам'яті;
– управління транзакціми і паралельним доступом;
– відновлення БД;
– підтримка мов БД;
– контроль доступу до даних;
– підтримка цілісності даних;
– підтримка незалежності даних;
– підтримка обміну даними.

Склад БД містить не тільки дані, що зберігаються, але і опис БД. Опис БД належить до *метаінформації*, тобто інформації про інформацію. Опис БД часто називають *схемою*. Централізоване сховище метаінформації називається словником даних.

Словник даних (*каталог даних*) – використовується для централізованого накопичення і опису ресурсів даних. Словник даних відповідає за визначення всіх елементів даних:

– імена, типи і розміри елементів даних;
– імена зв'язків;
– обмеження даних по підтримці цілісності;
– схеми БД (зовнішня, концептуальна і внутрішня), а також відображення між ними;
– імена користувачів і їх права доступу до даних;
– статистична інформація.

Програмні засоби БД включають в свій склад ядро СУБД, транслятори, утиліти, прикладні програми.

Мовні засоби поділяються на мови опису даних (МОД) і мови маніпулювання даними (ММД). МОД призначені для опису схеми БД або її частини. З її допомогою виконується опис типів даних, їх структур і зв'язків між собою. Відповідно до отриманого опису СУБД знаходить в програмі необхідні дані, перетворює їх і передає в прикладну програму. ММД виконує функції вибірки з БД даних за певними умовами, зміну даних, додавання даних, вилучення даних і т.ін.

Адміністратор даних – людина, яка відповідає за управління даними (планування БД, розробку і супроводження стандартів, прикладних алгоритмів і ділових процедур), а також за концептуальне і логічне проектування БД.

Адміністратор БД – людина, яка відповідає за фізичну реалізацію БД (фізичне проектування і втілення проекту), за забезпечення безпеки і цілісності даних, за супроводження операційної системи, а також за забезпечення максимальної продуктивності застосувань і користувачів.

Адміністратор даних і адміністратор БД виконують функції: управління структурою БД, управління паралельною обробкою, розподіл прав і обов'язків при обробці, забезпечення безпеки БД, відновлення БД, управління СУБД, підтримка репозиторія даних.

Адміністрування даними і БД передбачає управління інформаційними ресурсами, проектування БД, управління реалізацією застосувань, підтримку цілісності даних, захист даних, спостереження за поточною продуктивністю системи, а також реорганізацію БД при необхідності.

Переваги і недоліки застосування СУБД наведені в табл. 1.1.

Таблиця 1.1

Переваги і недоліки застосування СУБД

Переваги СУБД	Недоліки СУБД
Мінімізація збитковості даних	Використання значної частини ресурсів на потреби СУБД, а не на прикладну задачу
Несуперечливість даних і контроль їх 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 робітників
Багатокористувацький режим роботи	Наслідки збоїв

1.3. Компоненти системи баз даних

Компонентами системи баз даних є БД, СУБД і прикладні програми, з якими працюють як розробники, так і користувачі.

В СУБД входять такі компоненти (рис. 1.4): ядро СУБД, підсистема засобів проектування і підсистема засобів обробки.

Ядро СУБД – містить сукупність базових механізмів СУБД, які використовуються при будь-яких варіантах конфігурації системи. Ядро СУБД виконує функцію посередника між підсистемами засобів проектування і обробки і даними. Сучасні БД у більшості представляють користувачу дані у вигляді таблиць. Ядро СУБД отримує запити від інших компонентів в термінах таблиць, стовпців, рядків і перетворює

ці запити в команди операційної системи, які виконують запис і читання з фізичних носіїв інформації.

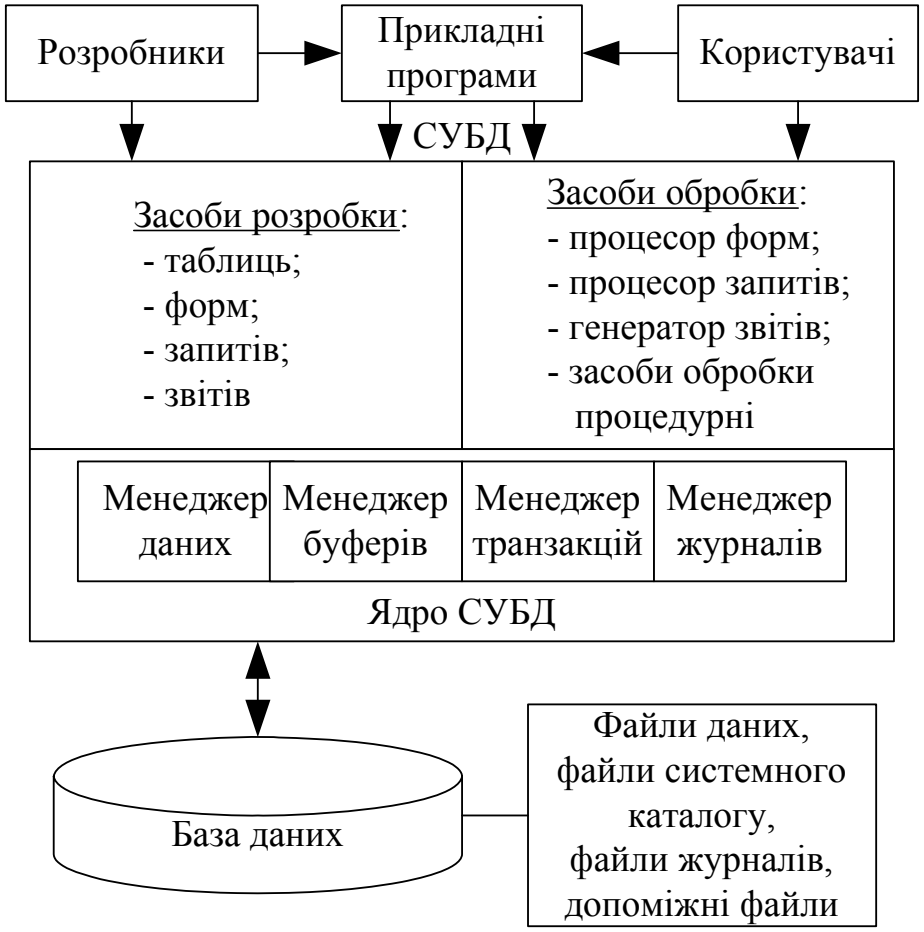


Рис. 1.4. Компоненти системи бази даних

Крім того, ядро СУБД задіяне в управлінні транзакціями, блокуваннях, резервному копіюванні і відновленні. В ядро СУБД входять менеджери буферів, даних, транзакцій, журналів.

Менеджер буферів – призначений для рішення задач ефективної буферізації оперативної пам'яті.

Менеджер даних – призначений для управління зовнішньою пам'ятью, забезпечення створення структур для даних, що зберігаються і допоміжних структур (індекси і т.ін.).

Менеджер транзакцій – підтримує механізми фіксації і відкату транзакцій, пов’язаний з менеджером буферів

оперативної пам'яті і забезпечує зберігання всієї інформації, яка потрібна після збоїв системи.

Менеджер журналів – забезпечує реєстрацію відомостей про виконання транзакцій, про працюючих користувачів, про виконання застосування, про доступи до різних структур даних і т.ін.

Підсистема засобів проектування являє собою набір інструментів, які спрощують проектування і реалізацію баз даних і їх застосувань. Як правило, цей набір містить засоби для створення таблиць, форм, запитів й звітів. В СУБД є також мови програмування і інтерфейси до них.

Підсистема обробки здійснює обробку компонентів застосування, які створені за допомогою засобів проектування.

Застосування БД складається з форм, запитів, звітів, меню і прикладних програм. Форми, запити і звіти можна створювати за допомогою засобів, що постачаються у комплекті з СУБД. Прикладні програми повинні бути написані або на вхідній мові СУБД, або на одній зі стандартних мов, а потім за допомогою СУБД з'єднані з БД.

Контрольні запитання

1. Дати визначення таких термінів: *інформація*, *інформаційна система*, *дані*, *предметна область*.
2. Дати визначення *бази даних* і *СУБД*.
3. Пояснити, чому база даних є моделлю. Яка існує різниця між реальністю і моделлю реальності?
4. Перелічити основні функції СУБД.
5. Перелічити переваги і недоліки СУБД.
6. Дати визначення словника даних.
7. Назвати основні компоненти системи бази даних і пояснити функцію кожної з них.
8. Перелічити основні етапи розвитку технологій баз даних.

9. Що таке банк даних?
10. Які складові частини містить банк даних?
11. Кого називають адміністратором бази даних?
12. Що таке ядро СУБД?
13. Які підсистеми входять в СУБД?

Глава 2. СЕРЕДОВИЩЕ БАЗИ ДАНИХ

2.1. Архітектура бази даних

Для організації роботи з БД необхідно забезпечити *незалежність* прикладних програм від даних. Це обумовлено тим, що при зміні системи, а також з метою забезпечення ефективного обслуговування користувачів необхідно виконувати роботи щодо зміни методів зберігання даних в БД, шляхів доступу до даних, змінювати структури і формати даних та зв'язки між ними. Якщо не застосовувати спеціальні підходи і при написанні застосувань вводити програмний опис методів доступу, засобів зберігання даних, формати даних, то при будь-якій зміні в БД для перелічених випадків буде необхідно корегувати текст програми користувача, що потребує значних витрат.

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

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

Опис структури даних на будь-якому рівні називається *схемою*. Існує три різних типи схем БД, які визначаються згідно з рівнями абстракції архітектури СУБД. На самому верхньому рівні є декілька *зовнішніх схем*, які відповідають різним

представленням даних. Цей рівень визначає точку зору на БД окремих застосувань. Кожне застосування бачить і обробляє тільки ті дані, які необхідні цьому застосуванню.

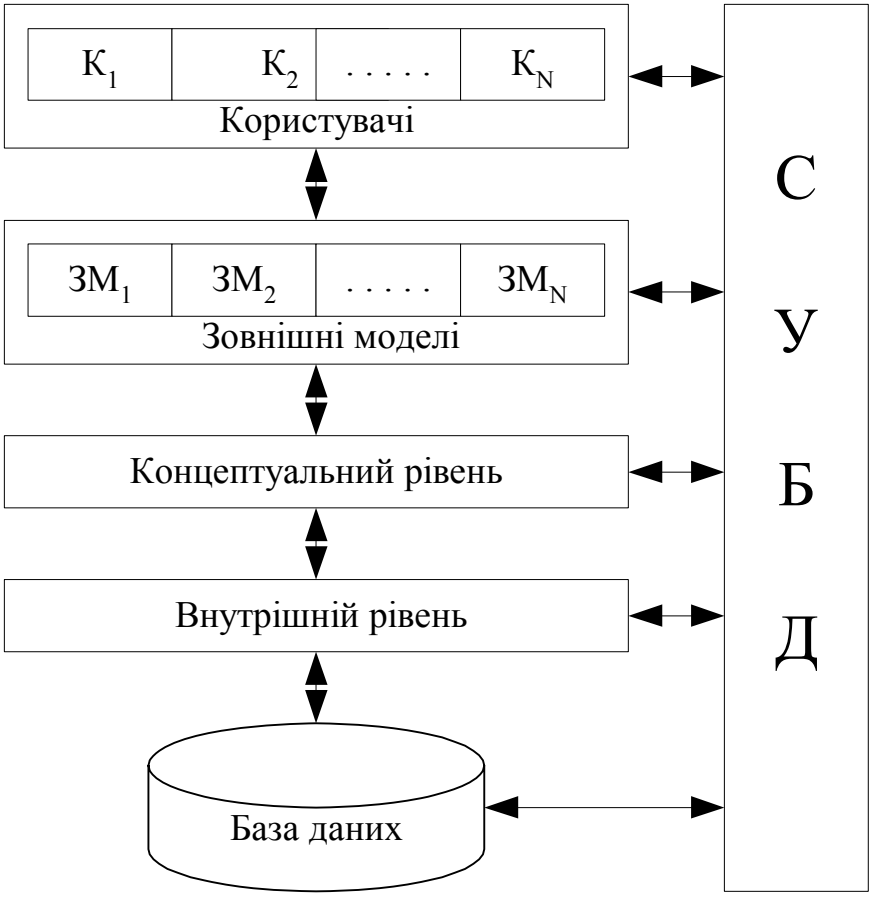


Рис. 2.1. Трирівнева архітектура СУБД

На концептуальному рівні опис БД називається *концептуальною схемою*. Тут БД представлена в найбільш загальному вигляді, який об'єднує дані, що використовуються всіма застосуваннями, які працюють з БД. Фактично концептуальний рівень відображає модель предметної області, для якої створювалася БД.

На внутрішньому рівні опис БД називається *внутрішньою схемою*. Тут БД представлена у вигляді безпосередньо даних, що розташовані в файлах, які відповідають *фізичній організації* БД.

Трирівнева архітектура СУБД дозволяє 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 вносяться в концептуальну схему. Зміни концептуальної схеми БД не викликають необхідності в корегуванні існуючих зовнішніх схем для користувачів, і відповідно не викликають змін в застосуваннях, що працюють з цими схемами. *Фізична незалежність від даних* означає захищеність концептуальної і зовнішніх схем від змін, що вносяться у внутрішню схему. До змін внутрішньої схеми належать використання різних файлових систем або структур даних, різних пристроїв зберігання, модифікація пошукових структур тощо.

Крім трьох названих рівнів абстрагування в БД існує ще один рівень, що передує їм. Цей рівень відображає інформацію про предметну область, а модель цього рівня називається *інфологічною моделлю* предметної області. Таким чином головними рівнями абстрагування в БД є рівні:

- інфологічний;
- зовнішній;
- концептуальний;
- внутрішній.

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

2.2. Моделі даних

Модель даних – це деяка абстракція, в якій знаходять своє відображення найбільш важливі аспекти функціонування визначеної предметної області, а другорядні – ігноруються. Модель даних являє собою деяку цільову модель предметної області. У моделі даних розрізняють три головні складові:

- *структурна частина*, яка визначає правила породження допустимих для даної СУБД видів структур даних;

– *управляюча частина*, яка визначає можливі операції над такими структурами ;
– *класи обмежень цілісності даних*, які можуть бути реалізовані засобами цієї системи.

Моделювання даних – це процес створення логічного представлення структури бази даних.

На рис. 2.2 показана класифікація моделей даних.

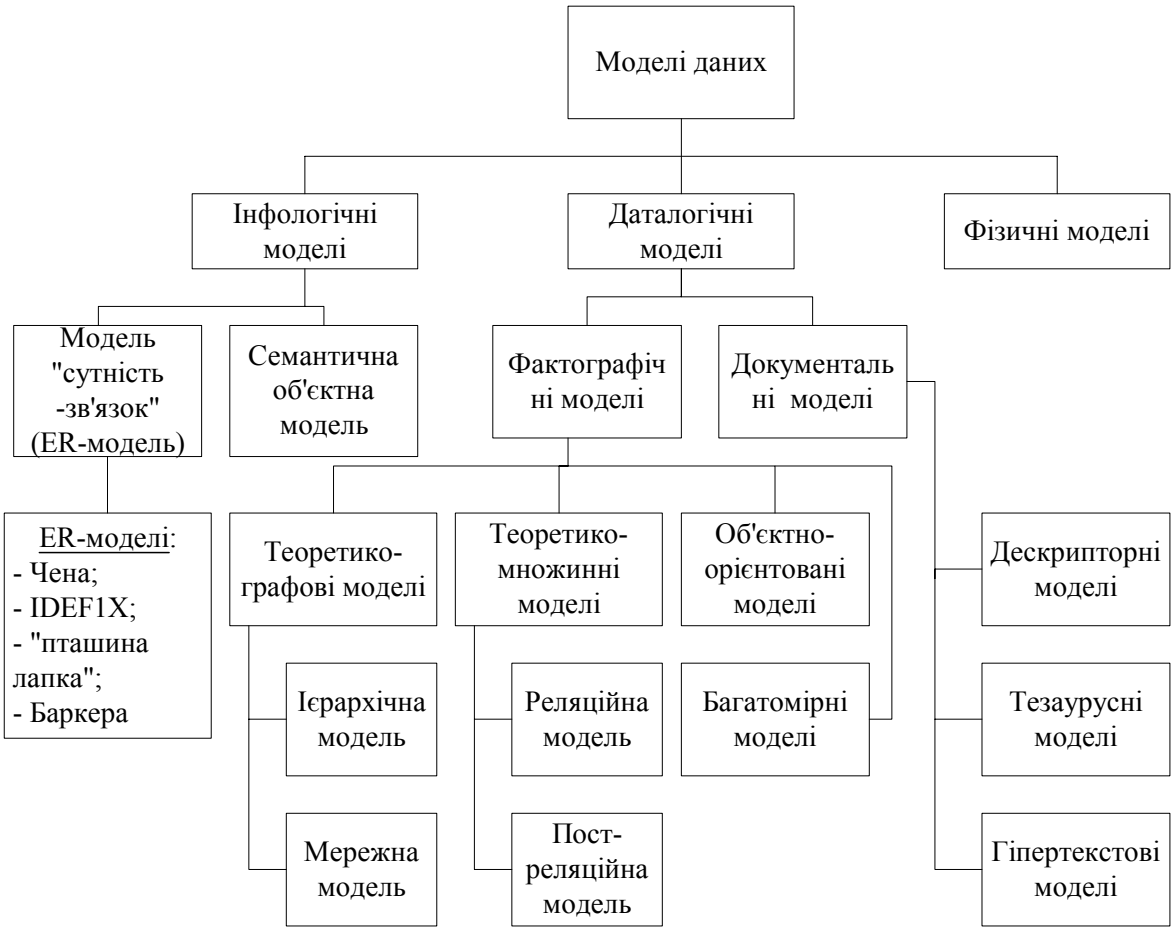


Рис. 2.2. Класифікація моделей даних

Кожному рівню представлення інформації відповідає певна модель.

Інфологічна модель – відображає інформацію про предметну область у вигляді незалежному від СУБД, що використовується. Ця модель відображає інформаційно-логічний рівень абстрагування, який пов'язаний з описом об'єктів предметної області, їх властивостей і взаємозв'язків.

Часто ці моделі ототожнюють з *концептуальними моделями* предметної області і називають *концептуальними інфологічними моделями* (внутрішня і зовнішня концептуальні інфологічні моделі).

Даталогічна модель – модель логічного рівня, яка відображає логічні зв'язки між елементами даних безвідносно до їх змісту і середовища збереження. Часто ці моделі ототожнюють з *логічними моделями*.

Фізична модель – описує те, як дані зберігаються в комп'ютері, представляючи інформацію про структуру записів, їх впорядкованість і про існуючі шляхи доступу до даних.

Модель "*сутність-зв'язок*" (ER-модель) – описує модель предметної області і складається з множини сутностей, множини звязків між сутностями, а також з атрибутів сутностей і зв'язків. В модель входить обмеження цілісності даних, що пов'язано з двома множинами сутностей і називається залежністю по існуванню. ER-моделі дозволяють графічно представляти моделі предметних областей. Вони є складовою частиною багатьох CASE-продуктів.

Семантична об'єктна модель – описує модель предметної області і являє собою модель даних. Ця модель складається з семантичних об'єктів, що містять сукупність атрибутів. Атрибути групуються у класи. Модель даних володіє більш розвиненими засобами відображення семантики у порівнянні з теоретико-множинними і теоретико-графовими моделями.

Теоретико-графова модель – модель даних, в якій дозволені структури даних можуть бути представлені у вигляді графа загального або спеціального виду, наприклад дерева. Необхідну групу операцій на мові маніпулювання даними, що засновані на цій моделі, представляють *навігаційні* операції. Операції над даними мають позаописовий характер.

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

Фактографічні моделі – містять відомості, які представлені у вигляді спеціальним чином організованих сукупностей формалізованих записів даних.

Документальні моделі – передбачають, що в якості одиничного елемента інформації виступає неподільний на менші складові частини документ, а інформація про документ, як правило, не структурується, або структурується в обмеженому вигляді. В цих моделях в основному розглядаються тексти на природній мові, формати документів є вільними.

Ієрархічна модель – модель даних в основі якої використовується ієрархічна, деревоподібна структура даних. Вершинами цієї структури є записи, які складаються з простих елементів даних різних типів. Батьківському запису відповідає довільне число екземплярів підлеглих записів кожного типу.

Мережна модель – модель даних, в якій дозволені структури даних можуть бути представлені 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 в цій моделі може мати довільне число батьків.

Реляційна модель – модель даних, яка заснована на математичному понятті відношення і представленні 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, яка вбудована в основну таблицю. Часто ці моделі ототожнюють з *об'єктно-реляційними моделями*.

Об'єктно-орієнтована модель – модель даних, яка базується на понятті об'єкта, тобто сутності, що володіє станом і поведінкою. Стан об'єкта визначається його атрибутами, а поведінка визначається сукупністю операцій, що визначені для

цього об'єкта. Також передбачається можливість підтримки зв'язків між типами об'єктів.

Багатомірна модель – модель даних, яка оперує багатомірним представленням даних (у вигляді *гіперкубу*) і орієнтована на підтримку аналіза даних. Передбачається конструювання різноманітних агрегацій даних у межах гіперкубу, побудова різних його проекцій – підмножин гіперкубу, деталізація і обертання даних, а також цілий ряд інших операцій.

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

Тезаурусна модель – описує кожен документ за допомогою дескрипторів, а також змістовних відношень між лексичними одиницями (ціле-частина, род-вид, клас-підклас і т.ін.). Ці моделі дозволяють підвищити ефективність дескрипторних моделей за рахунок більш ефективного відображення предметної області.

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

2.3. Програмні і мовні засоби баз даних

Основу програмних засобів банка даних складає СУБД. В СУБД можна виділити *ядро* СУБД, яке підтримує сукупність

базових механізмів роботи з БД, а також інші компоненти, які забезпечують засоби тестування, налагодження системи, утиліти, які забезпечують виконання таких додаткових функцій, як відновлення БД, збір статистики і т.ін. Важливою компонентою СУБД є транслятори і компілятори для мов, що використовуються. Для роботи з БД розробляються застосування.

Застосування – програма, яка призначена для рішення деякої сукупності задач в даній предметній області, або яка являє собою типовий інструментарій, що застосовується в різних предметних областях. Застосування може використовувати різні джерела даних (фактографічні, документальні, WEB і т.ін.), мати різну архітектуру (дволанкову, триланкову, розподілену).

Застосування бази даних – застосування, яке використовує ресурси деякої системи баз даних. Для доступа до БД використовується інтерфейс прикладного програмування СУБД, в середовищі якої він підтримується. Застосування можуть бути написані на стандартній алгоритмічній мові програмування (Pascal, C, Basic тощо) з вбудованими операторами на мові SQL.

Мова даних – мова, яка призначена для визначення даних, маніпулювання даними, а також інших функцій в термінах понять і рамках можливостей, які передбачені в моделі даних, що підтримується розглядуваною СУБД.

Мова запитів – мова доступу до БД, яка орієнтована на користувача. Мова запитів належить до декларативних мов, описує властивості і взаємозв'язки сутностей, але не описує алгоритм рішення задачі. Як правило мова запитів використовується в інтерактивному режимі, а також може вбудовуватися в програмний код застосувань.

Мова маніпулювання даними (*Data Manipulation Language – DML*) – мова, яка реалізує операційні можливості моделі

даних, що використовується. Ця мова визначає операції, які допустимі над даними, що знаходяться в БД.

Мова визначення даних (*Data Definition Language – DDL*) – мова, яка служить для опису структури БД, обмежень цілісності, а також, можливо, для специфікації процедур, що зберігаються, тригерів, обмежень управління доступом і т.ін. Функціональні можливості мов визначення і маніпулювання можуть інтегруватися в єдину мову даних.

Мова програмування баз даних – мова, яка забезпечує концептуально єдине інтегроване середовище, яке засновано на єдиній моделі даних, для програмування застосувань і управління даними в БД. Такі мови об'єднують функції традиційних мов програмування із засобами опису і маніпулювання даними в БД.

Мова програмування базова – традиційна мова програмування, для якої дана СУБД забезпечує *інтерфейс прикладного програмування* (API). Прикладна програма, яка написана на цій мові, має доступ до деяких функціональних можливостей СУБД і може виконувати з її допомогою доступ до БД.

Мови, які належать до мов четвертого покоління (Fourth-Generation Language – 4GL), мають такі функціональні можливості:

- генератори екранних форм для створення шаблонів вводу і відображення даних;
- генератори звітів на основі інформації, що зберігається в БД;
- генератори застосувань для створення програм обробки даних;
- генератори запитів;
- генератори для представлення даних у вигляді різного роду діаграм.

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 різниця між цима 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 даний час існують такі міжнародні стандарти на мову SQL: SQL1, SQL2, SQL3.

Мова SQL не володіє функціями повноцінної мови розробки і орієнтована на доступ до БД. Використання мови SQL може бути самостійним і вона може включатися в склад засобів розробки програм. В цьому випадку її називають *вбудованим* SQL. Розрізняють два головних методи використання вбудованого SQL: статичний і динамічний.

Статичне використання передбачає застосування в програмі функцій викликів мови SQL, які включаються в програмний модуль і виконуються після компіляції програми.

Динамічне використання передбачає динамічну побудову викликів функцій мови SQL та інтерпретацію цих викликів у ході виконання програми. Динамічний метод застосовується тоді, коли вид SQL запиту заздалегідь невідомий і будується у діалозі з користувачем.

Будь-яке SQL-застосування реляційної БД складається з трьох частин: інтерфейса користувача, набору таблиць в БД і SQL-машини.

2.4. Архітектура інформаційної системи

Ефективність функціонування інформаційної системи багато в чому залежить від її архітектури. Функціональні

частини інформаційної системи можуть розміщуватися на одному або на декількох комп'ютерах. У разі, якщо інформаційна система розміщується на одному комп'ютері, можливі такі варіанти використання програмних засобів: 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. 2.3).

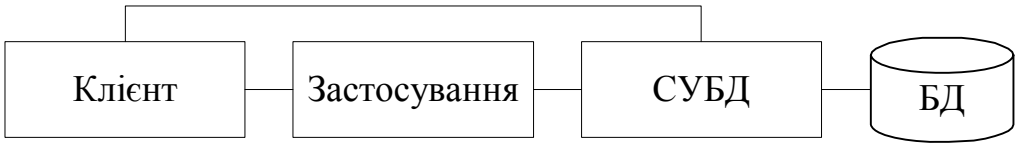


Рис. 2.3. Використання застосування і СУБД

У другому випадку взаємодія користувача і СУБД виконується за допомогою застосування (рис. 2.4). Такий підхід дозволяє підвищити швидкість роботи застосування, зменшити об'єм необхідної пам'яті.

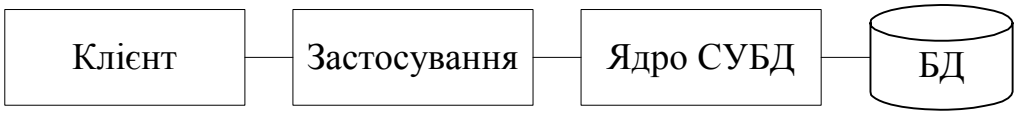


Рис. 2.4. Використання застосування і ядра СУБД

Створення незалежних застосувань дозволяє звертатися до БД без СУБД (рис. 2.5). Такий підхід дозволяє ще більше підвищити швидкість роботи застосування, зменшити об'єм необхідної пам'яті. Недоліки такого підходу пов'язані з трудомісткістю доробки застосувань, відсутністю стандартних засобів СУБД по обслуговуванню БД.

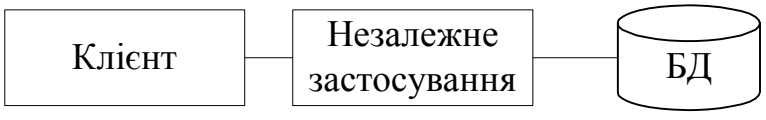


Рис. 2.5. Використання незалежного застосування

При інтеграції комп'ютерів в мережі виникає можливість розподілу застосувань, що працюють з єдиною БД, а також самої БД по мережі. Найбільш поширеною є схема, при якій кожен користувач маю свою персональну БД (КБД) і звертається до серверної БД (СБД) за інформацією, що спільно використовується багатьма користувачами (рис. 2.6).

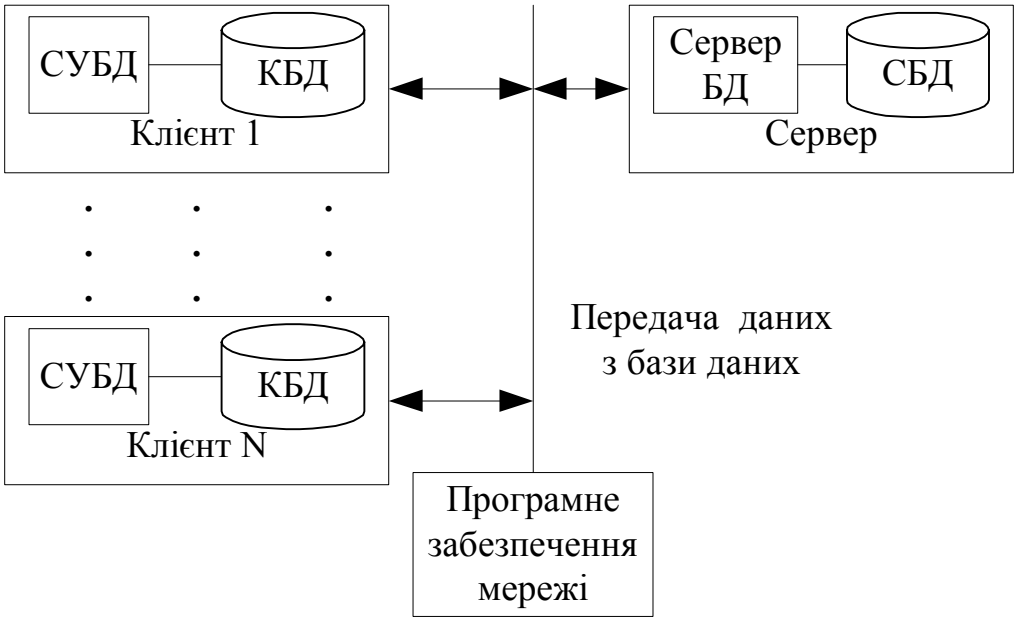


Рис. 2.6. Використання сервера БД

Під *сервером* розуміється комп'ютер або програма, які керують певними ресурсами. *Клієнт* – це теж комп'ютер або програма, які використовують цей ресурс.

Такий підхід дозволяє поєднувати переваги централізованого зберігання з індивідуальною роботою користувачів.

Контрольні запитання

1. Представити та пояснити архітектуру бази даних.
2. Які головні рівні абстрагування в базах даних?
3. Назвати основні *моделі даних* і дати їх характеристику.

4. В чому різниця між реляційною і об'єктно-реляційною моделями?
5. Які переваги і недоліки об'єктно-орієнтованої моделі?
6. Дати класифікацію програмних і мовних засобів по роботі з БД.
7. Вказати основні архітектурні рішення інформаційної системи з базами даних.
8. Що таке схема БД? Перелічити її компоненти.
9. Що розуміється під незалежністю даних?

Глава 3. РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ

3.1. Базові поняття

Реляційна модель даних заснована на математичному понятті відношення і представленні відношень у вигляді таблиць. Запропонована на початку 70-х років американським вченим Е.Коддом. В будь-якій реляційній СУБД припускається, що користувач сприймає БД як набір таблиць. Це стосується тільки логічної структури БД, тобто відноситься до концептуального і зовнішнього представлень. На фізичному рівні БД реалізується за допомогою різних структур зберігання. В табл. 3.1 наведені елементи реляційної моделі.

Для однозначної ідентифікації рядків, для зв'язування таблиць між собою, для прискорення операцій над даними застосовують ключі. В табл. 3.2 наведені можливі види реляційних ключів. Зовнішній і відповідний йому потенційний ключі повинні бути визначені на одному домені.

Таблиця 3.1

Елементи реляційної моделі

Елементи реляційної моделі	Форма представлення
Відношення	Таблиця
Кортеж	Рядок таблиці
Атрибут	Заголовок стовпця таблиці
Ключ	Сукупність атрибутів, які унікально визначають кожен рядок таблиці, або виконують функції зв'язування таблиць, або дозволяють прискорити операції над таблицями
Домен	Множина значень атрибута
Схема відношення	Рядок заголовків стовпців таблиці

Таблиця 3.2

Реляційні ключі

Назва	Пояснення
Потенційний ключ (Candidate Key)	Мінімальна підмножина атрибутів відношення, які єдиним чином ідентифікують кортеж даного відношення
Первинний ключ (Primary Key)	Потенційний ключ, який обрано для унікальної ідентифікації кортежів відношення
Вторинний ключ (Secondary Key)	Ключ, кожному значенню якого може відповідати більш ніж один екземпляр індексованих даних
Зовнішній ключ (Foreign Key)	Сукупність атрибутів відношення, значення яких є одночасно і значеннями первинного або потенційного ключа іншого відношення

Приклад. Розглянемо відношення *Студент* (рис. 3.1).

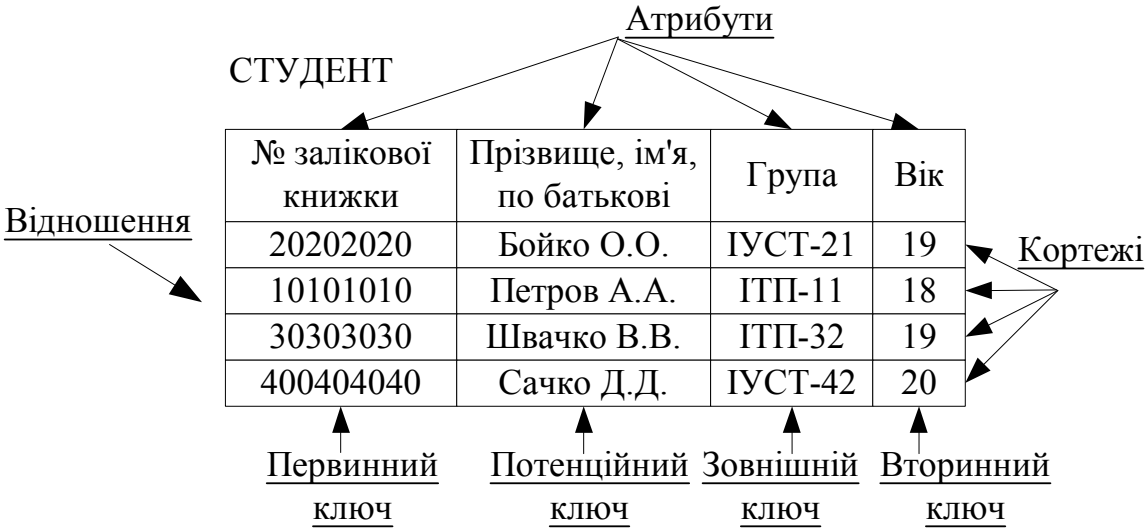


Рис. 3.1. Відношення *Студент*

Порядок кортежів у відношенні не визначений. В реляційних СУБД для зручності кортежі впорядковують за допомогою ключів (первинних або вторинних). В якості первинного ключа виступає атрибут *№ залікової книжки*, який дозволяє унікально ідентифікувати кожен кортеж. Атрибут *Вік* обирається в якості вторинного ключа (не є обов'язковим) для виконання операцій сортування і групування студентів за віком. Атрибут *Група* обирається в якості зовнішнього ключа для зв'язування з таблицею *Група* (на рис.3.1 не представлена). Домени показують множину всіх можливих значень певного атрибута відношення. Наприклад, для атрибута *Вік* значення домену відноситься до типу цілих чисел.

Реляційна модель складається з таких частин:

- *структурна* (тут фіксується відношення як єдине ціле);
- *маніпуляційна* (тут використовуються два базових механізми маніпулювання реляційною БД – реляційна алгебра і реляційні обчислення);
- *цілісність* (тут використовується механізм, який запобігає руйнуванню даних).

Реляційній моделі даних властиві простота і природність використовуваних структур даних і операцій маніпуляції даними, повна незалежність від середовища зберігання даних, *підтримка віртуальних, а не фізичних зв'язків між даними* (на основі значень даних, а не покажчиків).

Реляційна БД включає в себе такі складові:

– *інформаційні масиви* (таблиці, індекси);
– *системна інформація* (структура БД, обмеження цілісності);
– *прикладні програми* (процедури, тригери).

Операційні можливості відношення мають дві еквівалентні форми – *реляційна алгебра* і *реляційне обчислення*. У свою чергу реляційне обчислення поділяється на реляційне обчислення зі змінними кортежами, яке називається *обчислення кортежів*, і на реляційне обчислення зі змінними доменами, яке називається *обчислення доменів*. Для виконання запитів до БД Е.Кодд запропонував відповідні принципи побудови трьох мов.

Мови запитів реляційної алгебри – це алгебраїчні мови, які дозволяють висловлювати запити засобами спеціалізованих операторів, що застосовуються до відношень.

Мови реляційного обчислення дозволяють висловлювати запити шляхом специфікації предиката, якому повинні відповідати потрібні кортежі (домени).

Реальні мови запитів (SQL, QBE і т.ін.) забезпечують не тільки функції відповідної теоретичної мови, але і реалізують деякі додаткові операції (арифметичні, друку і т.ін.).

3.2. Цілісність баз даних

Цілісність баз даних – властивість даних, що визначає повноту і правильність інформації, яка вміщується в БД.

Підтримка цілісності даних включає такі складові:

– структурна цілісність;

– обмеження реальних значень даних;
– посилкова цілісність.

Структурна цілісність передбачає виконання таких умов:

– наявність тільки однорідних структур даних типу "реляційне відношення";
– відсутність дублікатів кортежів;
– обов'язкова наявність у кожному відношенні первинного ключа;
– обмеження доменів, яке передбачає визначення кожного атрибуту на своєму домені;
– можливість застосування невизначених значень NULL (позначає відсутність будь-якого значення атрибуту).

Невизначене NULL значення розглядається, як значення невідоме на даний момент часу. Це значення при появі додаткової інформації може бути замінено на деяке конкретне значення. Введення NULL викликало необхідність застосування замість двозначної логіки тризначної логіки. У цьому випадку передбачаються реляційні операції з невизначеними значеннями.

Обмеження реальних значень даних вимагають, щоби значення поля належали деякому діапазону значень, або задовольняли певному арифметичному співвідношенню між значеннями різних полів. Обмеження значень можуть включати також визначення певних форматів для полів, задоволення значень полів певним статистичним умовам, бізнес правилам предметної області і т.ін.

Посилкова цілісність означає, що зміни в таблицях повинні виконуватися синхронно, а зміст двох пов'язаних таблиць має відповідати таким правилам:

– кожному запису основної таблиці відповідає нуль або більше записів підлеглої таблиці;
– в підлеглій таблиці немає записів, які не мають батьківських записів в основній таблиці;

– кожний запис підлеглої таблиці має тільки один батьківський запис основної таблиці.

Умови цілісності даних визначають, які дані можуть бути записані в БД у результаті додавання або оновлення даних. При маніпулюванні даними в таблицях виконується контроль дій відповідно до табл. 3.3.

Таблиця 3.3

Правила вилучення і оновлення

Операція	**Правило**	**Пояснення**
Вилучення (DELETE)	RESTRICT	Заборона вилученні рядка з батьківської таблиці, якщо в підлеглій таблиці цей рядок має нащадків
	CASCADE	При вилученні рядка з батьківської таблиці в підлеглій таблиці всі рядки-нащадки автоматично вилучаються
	SET NULL	При вилученні рядка з батьківської таблиці в підлеглій таблиці всім зовнішнім ключам рядків-нащадків автоматично присвоюється значення NULL
	SET DEFAULT	При вилученні рядка з батьківської таблиці в підлеглій таблиці всім зовнішнім ключам рядків-нащадків автоматично присвоюється певне значення встановлене за замовчуванням
Оновлення (UPDATE)	RESTRICT	Заборона зміни первинного ключа в рядку батьківської таблиці, якщо в підлеглій таблиці цей рядок має нащадків

Закінчення табл. 3.3

Операція	Правило	Пояснення
Оновлення (UPDATE)	CASCADE	При зміні первинного ключа в рядку батьківської таблиці в підлеглій таблиці відповідні значення зовнішнього ключа також автоматично змінюються у всіх рядках-нащадках для того, щоби відповідати новому значенню первинного ключа
	SET NULL	При зміні первинного ключа в рядку батьківської таблиці в підлеглій таблиці відповідні значення зовнішнього ключа також автоматично змінюються у всіх рядках-нащадках і їм присвоюється значення NULL
	SET DEFAULT	При зміні первинного ключа в рядку батьківської таблиці в підлеглій таблиці відповідні значення зовнішнього ключа також автоматично змінюються у всіх рядках-нащадках і їм присвоюється певне значення встановлене за замовчуванням

Також можливо виконання правила NONE – не виконуються ніякі дії і правила NULL ALLOWED – дозволяються невизначені значення.

При введенні нових рядків (INSERT) необхідно дотримуватися такої послідовності введення: спочатку дані вводяться в батьківську таблицю, а потім − в підлеглу.

3.3. Реляційна алгебра

Алгеброю називається множина об'єктів із заданою на ній сукупністю операцій, які замкнені відносно цієї множини.

Основною множиною в реляційній алгебрі є множина відношень. Варіант реляційної алгебри, запропонований Коддом, містить такі основні операції: об'єднання, різниця, перетин, декартовий добуток, проекція, селекція, з'єднання, ділення. На рис. 3.2 показані основні операції реляційної алгебри.

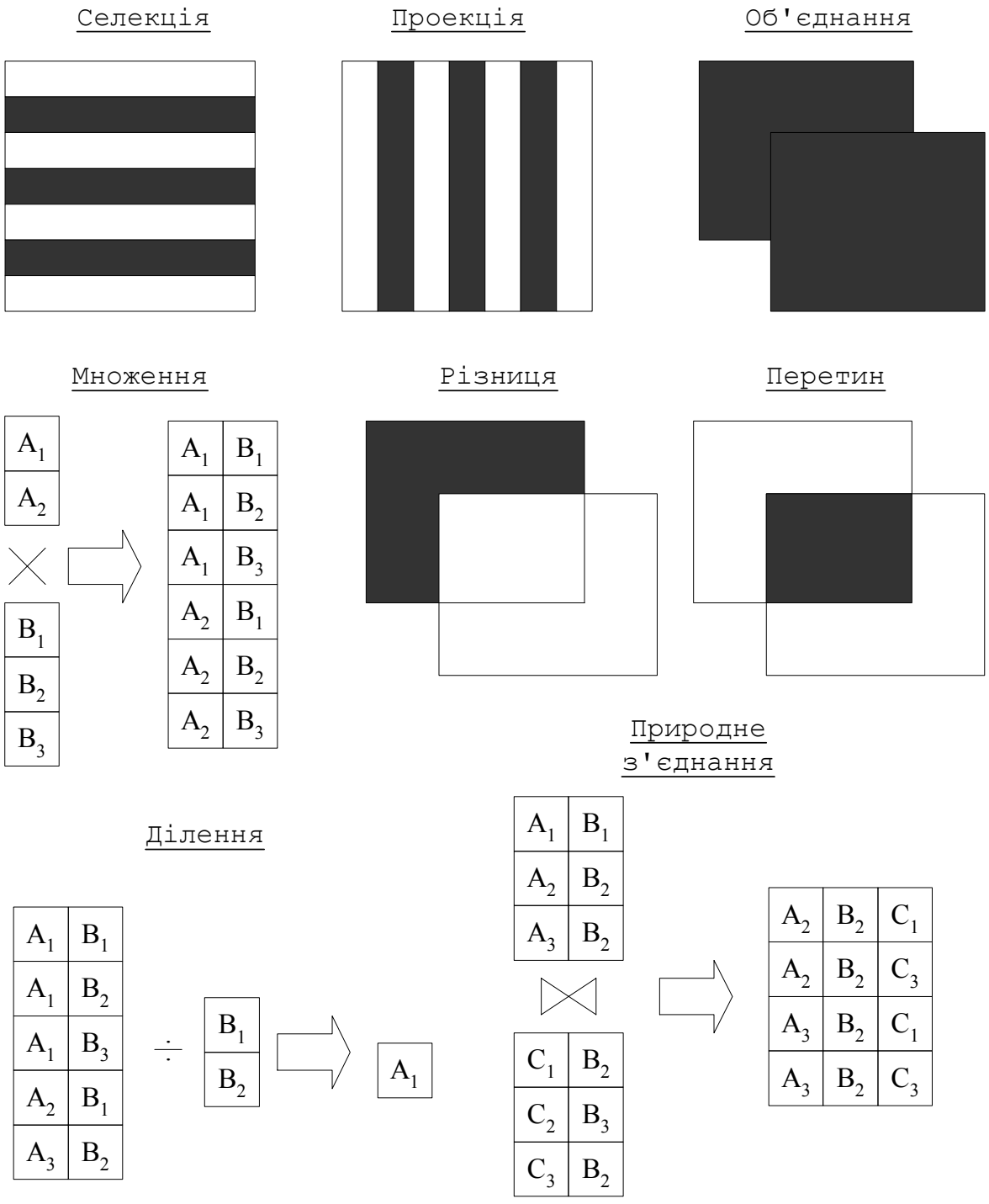


Рис. 3.2. Основні операції реляційної алгебри

В табл. 3.4 подані основні і додаткові операції реляційної алгебри.

Таблиця 3.4

Реляційні операції

Операція	Позначення	Зміст
Вибірка	$\sigma_{\text{предикат}}(R)$	Визначення відношення, яке вміщує тільки ті кортежі з відношення *R*, які задовольняють заданому предикату
Проекція	$\pi_{a1\ldots an}(R)$	Визначення відношення, яке вміщує вертикальну підмножину відношення *R*, що утворюється шляхом отримання значень вказаних атрибутів і виключення з результату рядків-дублікатів
Об'єднання	$R \cup S$	Визначення відношення, яке вміщує всі кортежі, що належать *R* або *S*, при цьому виключаються з результату всі дублікати кортежів. Відношення *R* і *S* повинні бути сумісні за об'єднанням
Перетин	$R \cap S$	Визначення відношення, яке вміщує всі кортежі, що належать *R* і *S*. Відношення *R* і *S* повинні бути сумісні за об'єднанням
Різниця	$R - S$	Визначення відношення, яке вміщує всі кортежі, що належать *R* і відсутні в *S*. Відношення *R* і *S* повинні бути сумісні за об'єднанням
Декартовий добуток	$R*S$	Визначення відношення, яке є результатом конкатенації кожного кортежа з відношення *R* з кожним кортежем з відношення *S*
Тета-з'єднання	$R \bowtie_F S$	Визначення відношення, яке вміщує кортежі з декартового добутку відношень *R* і *S*, що задовольняє предикату *F*

Закінчення табл. 3.4

Операція	Позначення	Зміст
З'єднання по еквівалентності	$R \bowtie_F S$	Визначення відношення, яке вміщує кортежі з декартового добутку відношень *R* і *S*, що задовольняє предикату *F* (предикат виконує порівняння тільки на рівність)
Природне з'єднання	$R \bowtie S$	Визначення відношення, яке отримано з'єднанням по еквівалентності двох відношень *R* і *S*, що виконано по всім спільним атрибутам *x*, з результатів якого вилучається по одному екземпляру кожного спільного атрибута
Ліве зовнішнє з'єднання	$R \supset \lhd S$	Визначення відношення, для якого кортежи відношення *R*, які не мають співпадаючих значень в спільних стовпцях відношення *S*, також включаються в результуюче відношення
Напівз'єднання	$R \rhd_F S$	Визначення відношення, яке вміщує ті кортежі відношення *R*, які входять в з'єднання відношень *R* і *S*
Ділення	$R \div S$	Визначення відношення, яке вміщує ті кортежі відношення *R*, які визначені на атрибуті *C*, що відповідає комбінації всіх кортежів відношення *S*, де *C* – множина атрибутів, які є у відношенні *R*, але відсутні у відношенні S

Приклад. Задані два відношення *Студент* і *Дисципліна*.

Студент

Прізвище	Курс	Група	Спеціальність
Бойко	2	1	ІТП
Левченко	3	2	ІУСТ

Дисципліна

Назва	Курс	Спеціальність	Викладач	Семестр
Бази даних	3	ІТП	Петренко	5
Системний аналіз	4	ІУСТ	Гавриш	7

1. Визначити всіх студентів спеціальності ІУСТ.

$\Pi_{\text{прізвище}}(\sigma_{\text{спеціальність="ІУСТ"}}\ (\textit{Студент}))$

2. Визначити всіх студентів, для яких у 7 семестрі викладач Гавриш проводить заняття.

$\Pi_{\text{прізвище}}(\sigma_{\text{семестр=7}\wedge\text{викладач="Гавриш"}}\ (\textit{Студент} \bowtie \text{Дисципліна}))$

$\Pi_{\text{прізвище}}(\textit{Студент} \bowtie (\sigma_{\text{семестр=7}\wedge\text{викладач="Гавриш"}}\ \text{Дисципліна}))$

3.4. Обчислення кортежів

Будь-який вираз обчислення кортежів може бути представлений у такому вигляді:

$$\{\ t\ /\ f(t)\ \}$$

де t – єдина вільна змінна – кортеж, яка позначає кортеж фіксованої довжини; $f(t)$ – деякий предикат над змінною t.

Формули в реляційному обчисленні кортежів будують з атомів і сукупності операторів (арифметичних і логічних). Вираз в реляційному обчисленні кортежів будують над множиною відношень. Типи можливих атомів подані в табл. 3.5.

Для побудови формули (запиту) використовуються логічні зв'язки ($\wedge,\vee,\neg$), а також квантори загальності $\forall x$ і квантори існування $\exists x$. Квантори зв'язують певні змінні. Зв'язана змінна відповідає локальній змінній у мові програмування, а вільна змінна (змінна, яка не зв'язана кванторами) відповідає глобальній змінній.

Таблиця 3.5

Правила побудови атомів формул

Номер типу атому	Правила побудови атомів
1	$R(t)$, де R – ім'я відношення; цей атом означає, що t є кортеж у відношенні R
2	$(s[i]\theta u[j])$, де s і u – змінні кортежі; θ – арифметичний оператор відношення; i і j – номери або імена компонентів (стовпчиків) у відповідних кортежах; $s[i]$ – позначення i-го компонента в кортежі-змінній s; $u[j]$ – позначення j-го компонента в кортежі-змінній u
3	$(s[i]\theta a)$ або $(a\theta s[i])$, де a – константа

Формули будуються за певними правилами. Правила побудови формул наведені в табл. 3.6.

Таблиця 3.6

Правила побудови формул

Номер правила	Правила побудови формул
1	Кожен атом є формула
2	Якщо f_1 і f_2 – формули, то вирази: – $f_1 \wedge f_2$; – $f_1 \vee f_2$; – $\neg f_1$; також є формулами
3	Якщо f – формула, то вирази: – $\forall t\, f(t)$; – $\exists t\, f(t)$; також є формулами
4	Якщо f – формула, то (f) – також є формулою
5	Ніщо інше не є формулою

Приклад.

1. Вираз $\{t/R_1 \wedge R_2\}$ означає, що в якості формули виступає запис $R_1 \wedge R_2$ і що тут визначається множина кортежів, яка одночасно належить відношенням R_1 і R_2. Цей вираз є еквівалентним до виразу реляційної алгебри $R_1 \cap R_2$.
2. Визначити всіх студентів спеціальності ІУСТ.

   $\{t(прізвище)/\exists tСтудент(t) \wedge t(спеціальність)=\text{"ІУСТ"}\}$

3. Визначити всіх студентів, для яких у 7 семестрі викладач Гавриш проводить заняття.

$\{t(прізвище) / \exists t\ Студент(t) \wedge \exists s\ Дисципліна(s) \wedge$
$s(семестр)=7 \wedge s(викладач)=\text{"Гавриш"} \wedge$
$t(курс, спеціальність) =$
$s(курс, спеціальність)\ \}$

3.5. Обчислення доменів

Будь-який вираз обчислення доменів може бути представлений у такому вигляді:

$$\{ x_1,x_2,\ldots,x_n / f(x_1,x_2,\ldots,x_n) \}$$

де f – формула; $x_1,x_2,\ldots,x_n$ – вільні змінні

В обчисленні доменів не існує змінних кортежів. Замість них вводяться змінні на доменах. У всіх інших випадках реляційне обчислення зі змінними на доменах будується так само, як і реляційне обчислення зі змінними на кортежах.

Формули в реляційному обчисленні доменів будують з атомів і сукупності операторів (арифметичних і логічних). Вираз в реляційному обчисленні доменів будують над множиною відношень. Типи можливих атомів подані в табл. 3.7.

Для побудови формули (запиту) використовуються логічні зв'язки ($\wedge$, $\vee$, $\neg$), а також квантори загальності $\forall x$ і квантори існування $\exists x$.

Таблиця 3.7

Правила побудови атомів

Номер типу атому	Правила побудови атомів
1	$R(x_1,x_2,\ldots,x_n)$, де R – n-арне відношення; x_i – константа або змінна на деякому домені. Атом $R(x_1,x_2,\ldots,x_n)$ вказує на те, що значення тих x_i, які є змінними, повинні бути вибрані так, щоби $(x_1,x_2,\ldots,x_n)$ було кортежем відношення R
2	$(x\theta y)$, де x і y – константи або змінні на деякому домені; θ – арифметичний оператор відношення; Атом $(x\theta y)$ вказує на те, що x і y являють собою значення, при яких істинно $(x\theta y)$

Приклад.

1. Визначити всіх студентів спеціальності ІУСТ.

   $\{$*прізвище*$/\exists x \exists y$ *Студент*(*прізвище*,*x*,*y*, "ІУСТ")$\}$

2. Визначити всіх студентів, для яких у 7 семестрі викладач Гавриш проводить заняття.

   $\{$*прізвище*$/\exists x \exists y \exists z$*Студент*(*прізвище*,*x*,*y*,*z*)$\wedge$

   $\exists s$ *Дисципліна*(*s*,*x*,*z*,"Гавриш",7)$\}$

Контрольні запитання

1. Які головні переваги реляційної моделі?
2. Дати визначення термінів: *відношення*, *схема відношення*, *кортеж*, *ключ*.
3. Які види ключів існують і навіщо вони потрібні?

4. Що таке домени і навіщо вони потрібні?
5. Що таке цілісність БД і як вона підтримується?
6. Що таке логічна і фізична цілісність БД?
7. Що таке посилкова цілісність і як вона підтримується?
8. Перелічити правила вилучення і оновлення даних у зв'язаних відношеннях.
9. Навести приклади обмежень значень і структурних обмежень.
10. Назвати основні операції реляційної алгебри.
11. Назвати додаткові операції реляційної алгебри, навести приклади.
12. Охарактеризувати варіанти реляційного обчислення.
13. Що таке обчислення кортежів?
14. Що таке обчислення доменів?
15. Перелічити правила побудови атомів в обчисленні кортежів і в обчисленні доменів.
16. Перелічити правила побудови формул в обчисленні кортежів і в обчисленні доменів.
17. Порівняти можливості реляційної алгебри і реляційного обчислення.

Частина 2. ПРОЕКТУВАННЯ БАЗ ДАНИХ

Глава 4. ЖИТТЄВИЙ ЦИКЛ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Життєвий цикл бази даних

Інформаційна система виконує функції збору та збереження даних, а також обробку та маніпулювання даними. Вона забезпечує структурування даних і управління ними.

Ефективність роботи інформаційної системи залежить від таких складових:

- проекту та реалізації бази даних;
- проекту та реалізації застосувань;
- супроводження інформаційної системи.

База даних є фундаментальним компонентом інформаційної системи і проектування БД виконується в рамках проектування інформаційної системи.

Інформаційна система має *життєвий цикл* (*Systems Development Life Cicle, SDLC*), який складається з таких етапів:

- планування;
- збір і аналіз вимог;
- проектування;
- реалізація;
- тестування;
- супроводження.

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. 4.1). Життєвий цикл БД складається з таких етапів:

- планування БД;
- аналіз вимог до БД;
- проектування БД (концептуальне, логічне, фізичне);
- розробка застосувань;
- реалізація і завантаження даних;
- тестування;
- експлуатація.

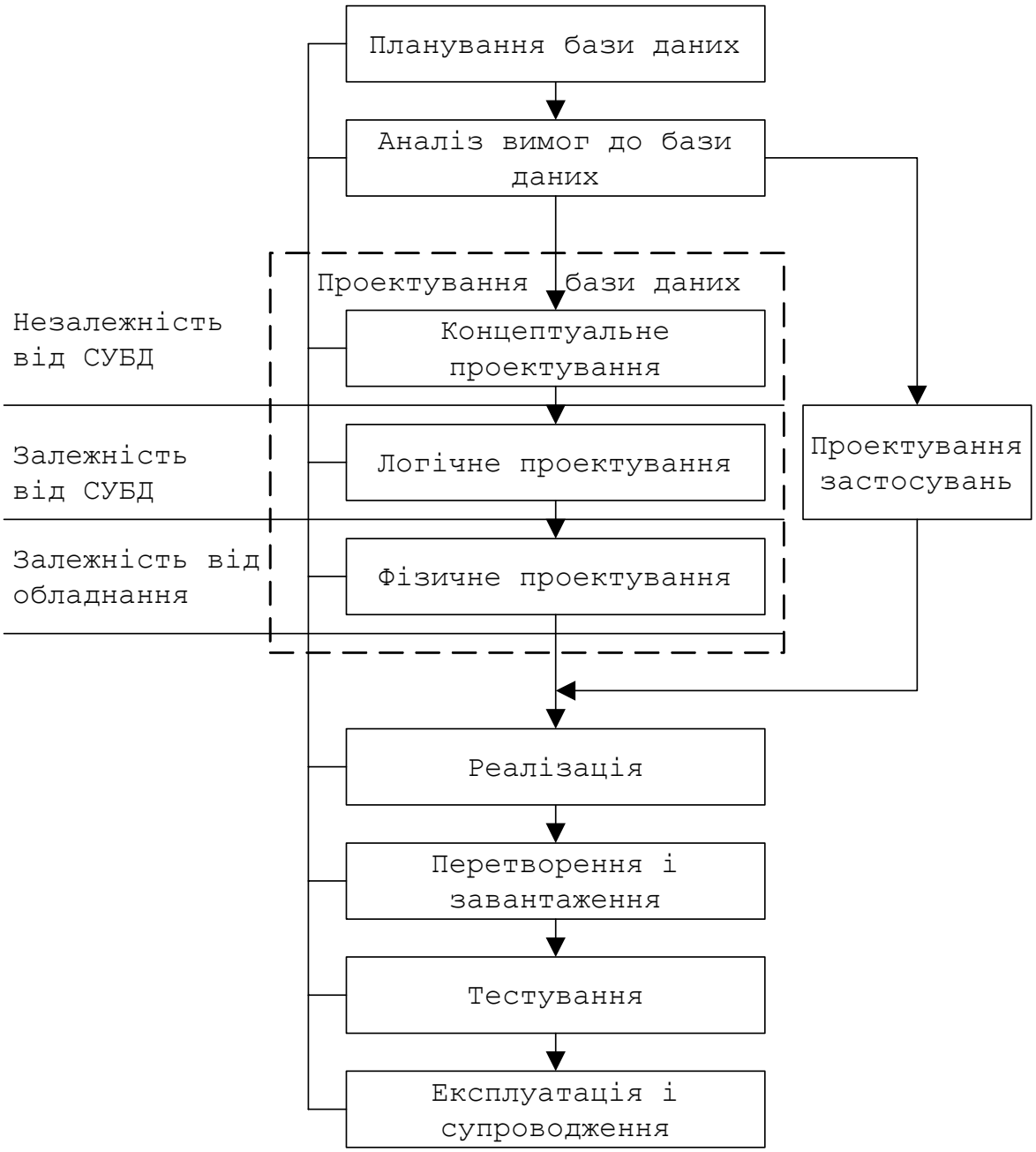


Рис. 4.1. Етапи життєвого циклу бази даних

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.

4.2. Планування бази даних

Етап планування бази даних передбачає розробку загального стратегічного плану, який дозволить ефективно реалізувати етапи життєвого циклу БД. Тут вирішуються такі питання:

- аналіз існуючих інформаційних систем;
- доцільність зміни існуючої інформаційної системи;
- обсяг робіт і ресурсів, вартість проекту;
- визначення технічного завдання для проекту бази даних;
- визначення технічних вимог;
- розробка методології збору даних, визначення їх формату;
- визначення необхідної документації;
- визначення послідовності проектування і реалізації застосувань.

4.3. Аналіз вимог до бази даних

На етапі аналізу вимог до бази даних вирішуються такі задачі:

- визначення діапазону дії і границь застосувань БД;
- визначення складу користувачів і областей застосування;
- визначення представлень користувачів, що підтримуються БД.

На цьому етапі також збираються і аналізуються вимоги користувачів:

- опис даних, що застосовуються (вхідні і вихідні документи);
- детальні відомості про транзакції;
- відомості про засоби застосування даних.

На основі всієї цієї інформації складаються *специфікації вимог* користувачів.

4.4. Проектування бази даних

Процес проектування БД являє собою послідовність переходів від неформального мовного опису інформаційної структури предметної області до формалізованого опису об'єктів предметної області в термінах деякої моделі. Проектування БД складається з таких етапів:

- системний аналіз предметної області;
- концептуальне проектування;
- логічне проектування;
- фізичне проектування.

Системний аналіз передбачає мовний опис реальних об'єктів предметної області, визначення зв'язків між об'єктами, дослідження характеристик об'єктів і зв'язків. Результати дослідження використовуються при концептуальном проектуванні БД.

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

Функціональний підхід застосовує рух "від задач" і використовується у тих випадках, коли заздалегідь відомі функції майбутніх користувачів БД, а також відомі всі задачі, для інформаційних потреб яких створюються БД. В цьому випадку на основі виробничих документів, опитувань замовників можна чітко визначити мінімальний набір об'єктів предметної області та їх взаємозв'язок.

Предметний підхід застосовується у тому випадку, коли інформаційні потреби майбутніх користувачів чітко не визначені. В цьому випадку не можна чітко визначити мінімальний набір об'єктів предметної області. В опис предметної області включаються об'єкти та зв'язки, які є найбільш характерними та найбільш суттєвими для неї. БД називається предметною і може використовуватися для розв’язання задач, які заздалегідь не визначені.

У практичній діяльності використовується комплексний підхід, який з одного боку дозволяє розв’язувати конкретні

інформаційні та функціональні задачі, а з іншого боку – враховує можливість додавання нових застосувань.

У загальному випадку існує два підходи до проектування БД: низхідне проектування і висхідне проектування (рис. 4.2).

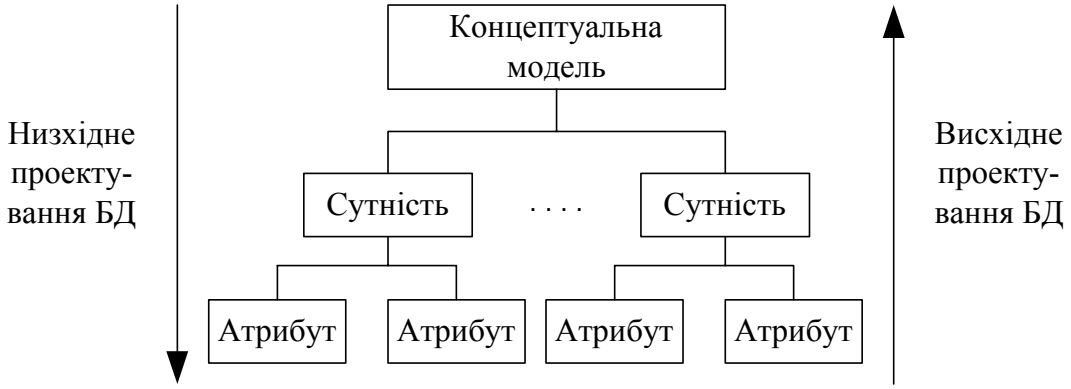


Рис. 4.2. Схема підходів до проектування бази даних

Низхідне проектування починається з визначення наборів даних, потім визначаються елементи даних для кожного з таких наборів. Цей процес включає в себе ідентифікацію різних типів сутностей і визначення атрибутів кожної сутності. Низхідне проектування включає операції *декомпозиції*, що передбачає заміну вихідної множини відношень, що входять в схему БД, іншою множиною відношень, які є проекціями вихідних відношень.

Цей підхід рекомендується застосовувати у тих випадках, коли кількість, різноманітність та складність сутностей, зв'язків і транзакцій значна за розмірами. Найбільш поширеними моделями для цього проектування є моделі "сутність – зв'язок" (ER-моделі, Entity-Relationship model).

Висхідне проектування починається з виявлення елементів даних, які потім групуються в набори даних. Спочатку визначаються атрибути, які потім об'єднуються в сутності. Висхідне проектування включає операції *синтезу*, що передбачає виконання компоновки із заданої множини

функціональних залежностей між об'єктами предметної області вихідних відношень схеми БД.

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

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

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

Фізичне проектування полягає в описі засобів фізичної реалізації логічного проекту БД. *Фізичні моделі* визначають засоби розміщення даних в середовищі зберігання і засоби доступу до цих даних, які підтримуються на фізичному рівні.

4.5. Розробка застосувань

Застосування – програма або програмна система, яка призначена для рішення деякої сукупності задач в даній предметній області, або яка являє собою типовий інструментарій, що застосовується в різних областях.

На цьому етапі вирішуються такі задачі:

- проектування транзакцій;
- проектування інтерфейсів користувачів.

Транзакція може складатися з декількох операцій по роботі з БД, які переводять БД з одного цілісного стану в інший. Розрізняють транзакції по отриманню певної інформації

з БД і транзакції по зміні даних в БД (оновлення, вилучення, додавання). Транзакції також можуть бути змішані.

Інтерфейс користувача – сукупність функціональних компонентів, які забезпечують взаємодію користувача з системою.

4.6. Реалізація

На етапі реалізації вирішуються такі задачі:

- встановлюється технічне і програмне забезпечення СУБД;
- реалізується проект БД;
- реалізуються прикладні програми;
- реалізуються форми вводу/виводу даних і звіти;
- наповнення БД даними;
- захист БД від несанкціонованого втручання;
- підтримка цілісності БД.

Реалізація БД виконується за допомогою створення опису на мові визначення даних певної СУБД або з використанням графічного інтерфейса користувача. Застосування реалізуються на мовах третього та четвертого покоління або на розширеннях мов БД. Реалізація може виконуватися за допомогою інструментів автоматизованого проектування.

4.7. Тестування

На етапі тестування вирішуються такі задачі:

- перевіряється вірність роботи окремих модулів або функціональних компонентів (*альфа-тестування*);
- проводяться виміри продуктивності роботи системи, визначаються потреби в ресурсах;
- здійснюється дослідницька експлуатація (*бета-тестування*), при якій перевіряється відповідність розробленої системи її специфікаціям.

Для покращання роботи системи можлива модифікація логічного і фізичного проекту, оновлення або зміна програмного забезпечення СУБД, зміна технічного забезпечення. Також для покращання роботи виконується налагодження системних параметрів і параметрів СУБД.

4.8. Експлуатація

На етапі експлуатації вирішуються такі задачі:

- контроль продуктивності роботи системи і в разі потреби підвищення продуктивності (наприклад за рахунок створення додаткових індексів);
- супроводження і модернізація застосувань БД;
- профілактичне обслуговування (резервне копіювання);
- корегуюче обслуговування (відновлення БД);
- призначення прав доступу для нових користувачів;
- ведення статистики доступу до БД для підвищення ефективності роботи системи;
- періодична перевірка безпеки;
- періодичні зведення використання системи.

Контрольні запитання

1. Перелічити основні етапи життєвого циклу бази даних.
2. Які зв'язки існують між життєвим циклом інформаційної системи і життєвим циклом бази даних?
3. Яка різниця між функціональним і предметним підходами до проектування бази даних?
4. У чому полягає спільність і різниця трьох етапів проектування бази даних?
5. Назвати етапи проектування БД.
6. У чому полягає планування бази даних?
7. У чому полягає аналіз вимог до бази даних?
8. У чому полягає розробка застосувань?
9. Які задачі вирішуються на етапі експлуатації?

Глава 5. КОНЦЕПТУАЛЬНЕ ПРОЕКТУВАННЯ БАЗ ДАНИХ

5.1. Концептуальні моделі

З концептуального проектування починається створення концептуальної схеми БД, в основі якої лежить *концептуальна модель даних*. Концептуальна модель представляє загальний погляд на дані. Розрізняють два головних підходи до моделювання даних при концептуальному проектуванні:

- 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). *ER-модель* складається із сутностей, зв'язків, атрибутів, доменів атрибутів, ключів. Моделювання даних відображає логічну структуру даних, так само, як блок-схеми алгоритмів відображають логічну структуру програми.

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

Зближення цих моделей реалізується в розширеному ER-моделюванні (Extended Entity Relationship model, EER-модель).

5.2. Модель "сутність-зв'язок"

ER-моделювання являє собою низхідний підхід до проектування БД, який починається з визначення найбільш важливих даних, які називаються *сутностями* (entities), і *зв'язків* (relationships) між даними, які повинні бути представлені в моделі. Потім в модель заноситься інформація про властивості сутностей і зв'язків, яка називається *атрибутами* (attributes), а

також всі обмеження, які відносяться до сутностей, зв'язків і атрибутів. ER-модель дає графічне представлення логічних об'єктів і їх відношень в структурі БД. Послідовність проведення ER-моделювання показана на рис. 5.1.

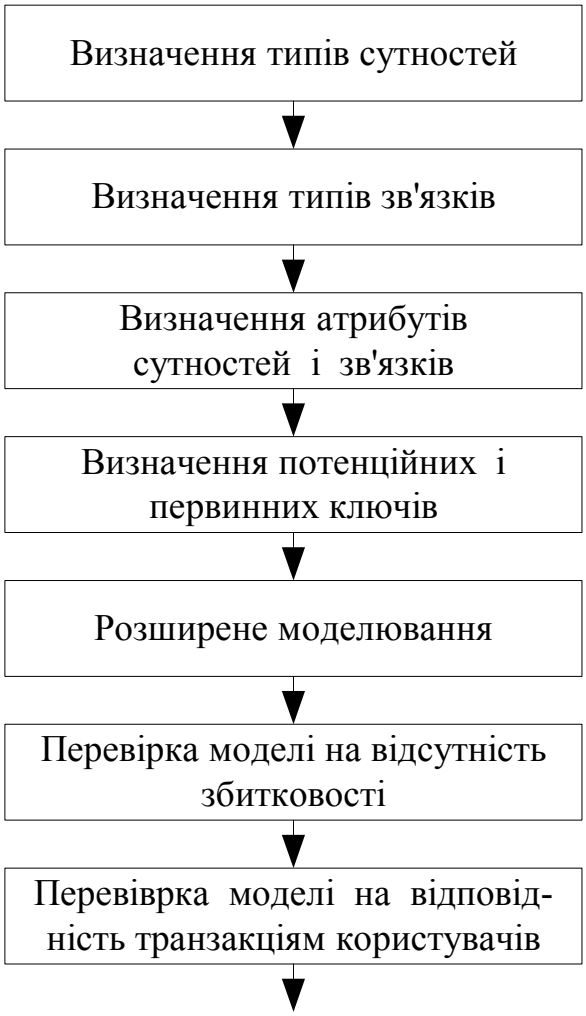


Рис. 5.1. Етапи побудови моделі "сутність – зв'язок".

Вперше поняття ER-моделі запровадив П.Чен. Підхід П.Чена дозволив концептуальне моделювання перевести в практичну площину проектування БД. У подальшому діаграми Чена набули розвитку у багатьох інших роботах з ER-моделювання. До них належать такі моделі:

- "пташина лапка", розроблена К.М. Бахманом;
- IDEF1X, розроблена Т.Ремеєм;
- на основі UML;
- модель Баркера і багато інших моделей.

Сутності

Сутність дозволяє моделювати клас однотипних об'єктів. Сутність має унікальне ім'я у межах системи, що моделюється. Оскільки сутність відповідає деякому класу однотипних об'єктів, то передбачається, що в системі існує багато екземплярів даної сутності. Об'єкт, якому відповідає сутність, має набір атрибутів, які характеризують його властивості. При цьому набір атрибутів повинен бути таким, щоби можна було розрізняти конкретні екземпляри сутності.

Приклад. Сутність *Викладач* може мати такі атрибути: *Табельний номер, Прізвище, Ім'я, По батькові, Посада, Вчений ступінь*.

Набір атрибутів, що однозначно ідентифікує конкретний екземпляр сутності, називають *ключовим*. У наведеному прикладі для сутності *Викладач* ключем буде *Табельний номер*, оскільки для всіх викладачів табельні номери різні. Екземпляром сутності *Викладач* буде опис конкретного викладача. Загальноприйняте позначення сутності – прямокутник (рис. 5.2).

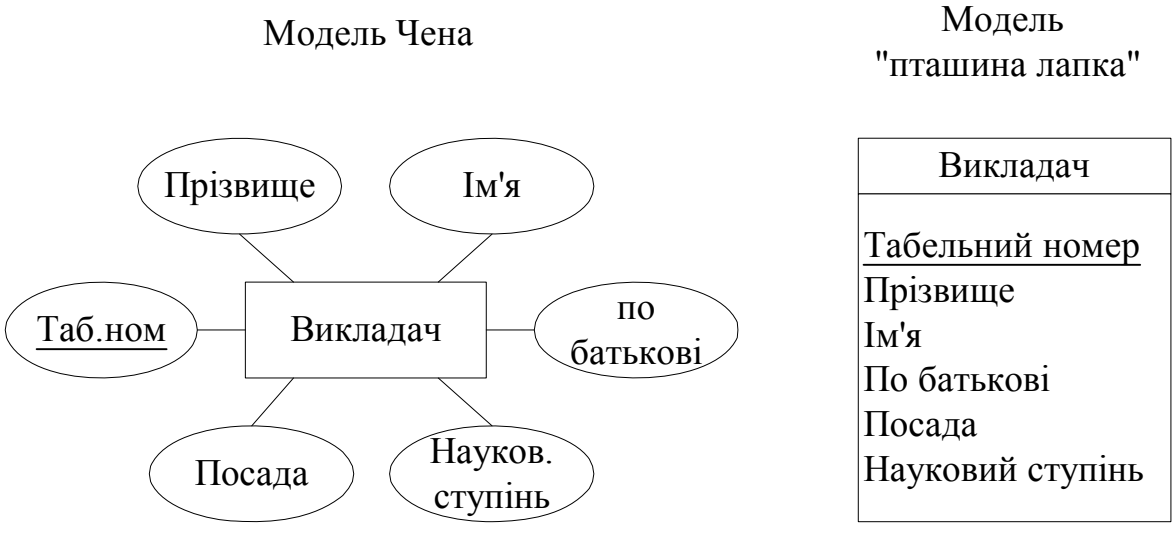


Рис. 5.2. Представлення сутностей і атрибутів у ER-діаграмах П. Чена і ER-діаграмах "пташина лапка"

Зв'язки

Між сутностями встановлюються зв'язки, які вказують яким чином сутності співвідносяться або взаємодіють між собою. Розрізняють такі зв'язки:

- між двома сутностями (бінарний зв'язок);
- між трьома сутностями (тернарний зв'язок);
- між N сутностями (N-арний зв'язок);
- між однією сутністю (рекурсивний зв'язок).

Найбільш поширеними є бінарні зв'язки. Зв'язок показує яким чином екземпляри сутностей зв'язані між собою. Бінарні зв'язки бувають:

- 1:1 (один до одного);
- 1:M (один до багатьох);
- N:M (багато до багатьох).

На рис. 5.3, 5.4 показані відображення цих зв'язків у різних ER-моделях.

Зв'язок "один до одного" (1:1): завідуючий кафедрою може керувати тільки однією кафедрою, а кожною кафедрою керує тільки один завідуючий

а

Зв'язок "один до багатьох" (1:M): на кафедрі працює багато викладачів, а кожен викладач працює тільки на одній кафедрі

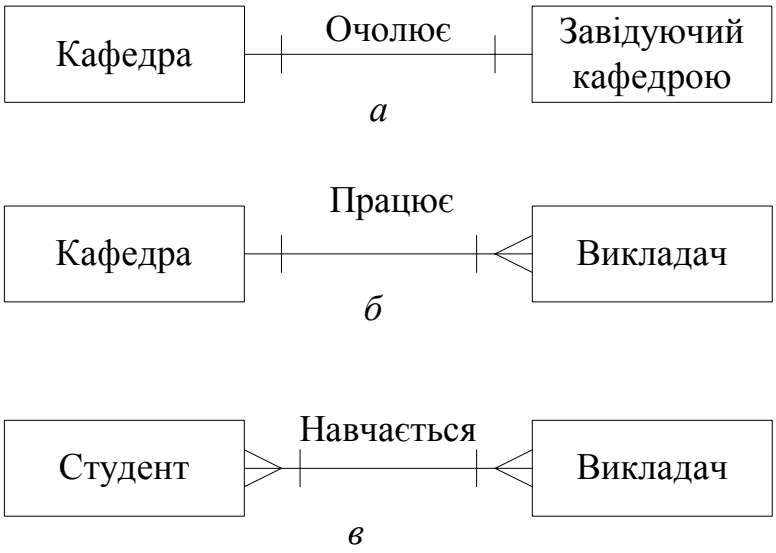
б

Рис. 5.3. Представлення зв'язків між відношеннями на діаграмі Чена: *а* –1:1; *б* – 1:M (див. також с. 56)

Зв'язок "багато до багатьох" (N:M): студент займається у багатьох викладачів, а кожен викладач навчає багатьох студентів

в

Рис. 5.3. Закінчення: *в* – N:M



а

б

в

Рис. 5.4. Представлення зв'язків між відношеннями на діаграмі "пташина лапка": *а* –1:1; *б* – 1:M; *в* – N:M

Атрибути

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

Приклад. Атрибут *Оцінка* може приймати чотири значення: 2, 3, 4, 5. Ці значення і складають домен цього атрибута.

Атрибути залежно від складності значень, які вони можуть приймати поділяються на певні категорії (табл. 5.1).

Таблиця 5.1

Типи атрибутів

Тип	Властивість
Простий	Атрибут, який не може бути поділений на інші атрибути. *Приклад*. Прізвище; посада
Складовий	Атрибут, який може бути поділений на інші атрибути. *Приклад*. Адреса; прізвище, ім'я, по батькові
Однозначний	Атрибут, який може приймати тільки одне значення. *Приклад*. Табельний номер, номер залікової книжки
Багатозначний	Атрибут, який може приймати багато значень. *Приклад*. Телефон, Адреса (постійне місце проживання і гуртожиток)
Похідний	Атрибут, який не зберігається в БД, а обчислюється за допомогою певного алгоритма *Приклад*. Вік (обчислюється по даті народження), кількість студентів в групі

Приклад. Розглянемо сутність *Студент* (рис. 5.5).

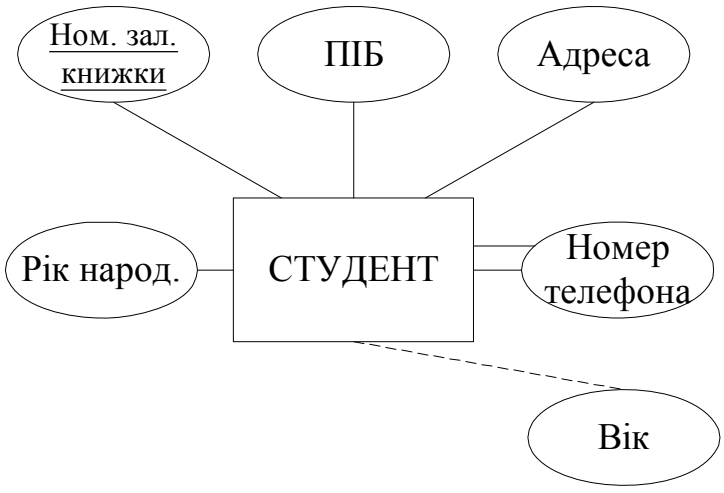


Рис. 5.5. ER-діаграма сутності *Студент*

Атрибути *номер залікової книжки*, *рік народження* є простими.

Атрибути *ПІБ* і *Адреса* є складовими. *ПІБ* може бути поділений на атрибути: *прізвище*, *ім'я*, *по батькові*, а *Адреса* – на індекс, місто, вулиця, будівля, квартира.

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

Атрибут *Номер залікової книжки* є однозначним: він не може приймати два значення для одного студента.

Атрибут *Номер телефону* є багатозначним: він може приймати декілька значень для одного студента (зображається подвійною лінією).

Атрибут або набір атрибутів сутності, які застосовуються для ідентифікації екземпляра сутності, називаються ***потенційним ключем***. Сутність може містити декілька потенційних ключів. В прикладі в якості потенційних ключів можуть бути такі атрибути: *Номер залікової книжки*, *Прізвище Ім'я по Батькові*.

Потенційний ключ, який вибрано 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, які входять в його склад. В прикладі в якості первинного ключа служить *Номер залікової книжки*.

Потужність зв'язків

Потужність зв'язку (***кардинальність***) відображає певне число екземплярів сутностей, які зв'язані з одним екземпляром зв'язаної сутності. В моделі Чена потужність зв'язку відображається вказівкою відповідних чисел поруч з сутностями у форматі (*x*, *y*). Перше число визначає мінімальне значення потужності зв'язку, а друге – його максимальне

значення. Потужність вказує на число екземплярів у зв'язаній сутності.

Відомості про максимальне і мінімальне значення потужності зв'язку може застосовуватися у прикладному програмному забезпеченні або за допомогою тригерів; на рівні таблиць СУБД не може оперувати з потужністю зв'язків.

Приклад. Розглянемо зв'язок *Група-Студент* (рис. 5.6).

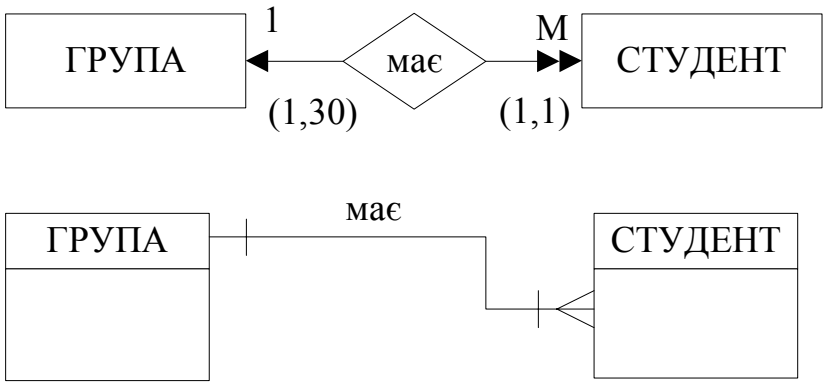


Рис. 5.6. Зв'язок і потужність в ER-діаграмах

Потужність (1,30) поруч із сутністю *Група* вказує, що в групі може займатися від 1 до 30 студентів. Потужність (1,1) поруч із сутністю *Студент* вказує, що студент може займатися тільки в одній групі (мінімум 1, максимум 1). У моделі "пташина лапка" числовий діапазон значень потужності не відображається в ER-діаграмах.

Сильні і слабкі зв'язки

Якщо сутність може існувати незалежно від інших сутностей, то вона є *незалежною від існування*. Якщо сутність залежить від існування інших сутностей, то вона є *залежною від існування*. Наприклад, сутності *Студент* і *Група* можуть існувати незалежно одна від одної, а сутність *Нагорода студента* є залежною від сутності *Студент* й існувати без неї не може.

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 відображається штриховою лінією.

Ідентифікаційний зв'язок або *сильний зв'язок* має місце у тому випадку, коли одна зв'язана сутність залежить від існування іншої. На ER-діаграмах "пташина лапка" сильний зв'язок відображається суцільною лінією.

Приклад. Розглянемо зв'язки *Група-Студент* і *Студент-Нагорода* (рис. 5.7).

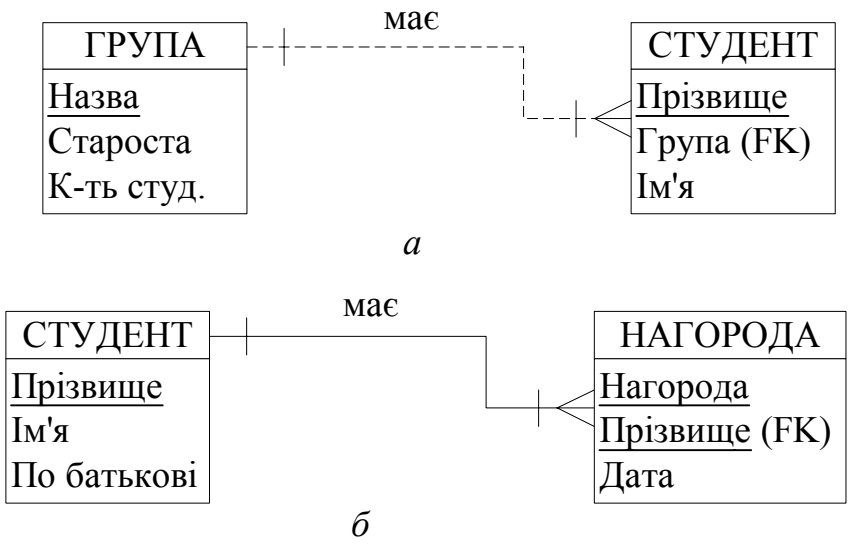


Рис. 5.7 Неідентифікаційний (*а*) та ідентифікаційний (*б*) зв'язки

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

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

Атрибути зв'язків

Так само як і сутності зв'язки можуть мати атрибути.

Приклад. Атрибути *День* і *Аудиторія* належать до зв'язку між сутностями *Студент* і *Дисципліна* (рис. 5.8).

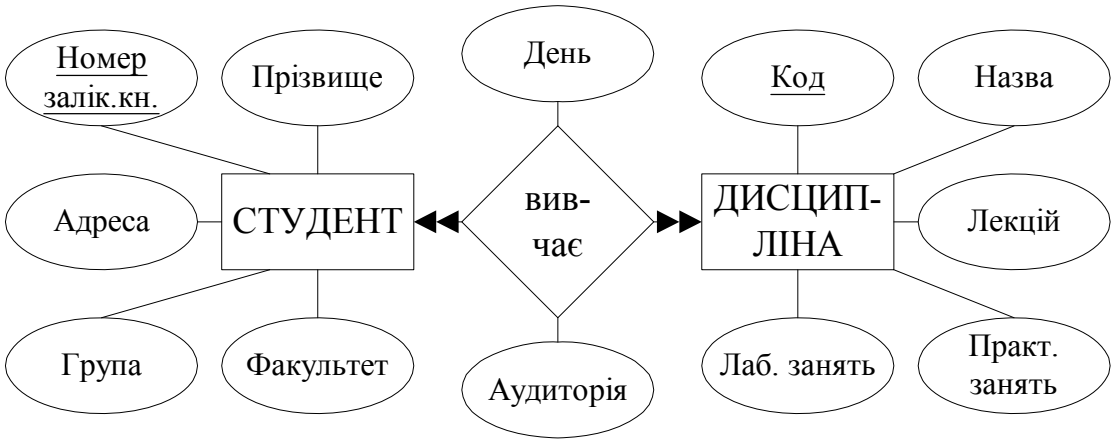


Рис. 5.8. ER-діаграма з атрибутами зв'язку *День* і *Аудиторія*

Обов'язкові і необов'язкові зв'язки

Участь сутності у зв'язку може бути обов'язковою або необов'язковою. Якщо один екземпляр сутності не потребує наявності відповідного екземпляра сутності в окремому зв'язку, то участь сутності у зв'язку є ***необов'язковою***. Наприклад, у зв'язку сутностей *Студент-Нагорода* – не кожен студент має нагороди. Тобто не кожен екземпляр в таблиці *Студент* потребує обов'язкової наявності екземпляра сутності в таблиці *Нагорода*. Сутність *Нагорода* розглядається як необов'язкова по відношенню до сутності *Студент*. 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. 5.9). Необов'язково кожен студент має нагороду, але якщо є нагорода, то вона обов'язково зв'язана з певним студентом.

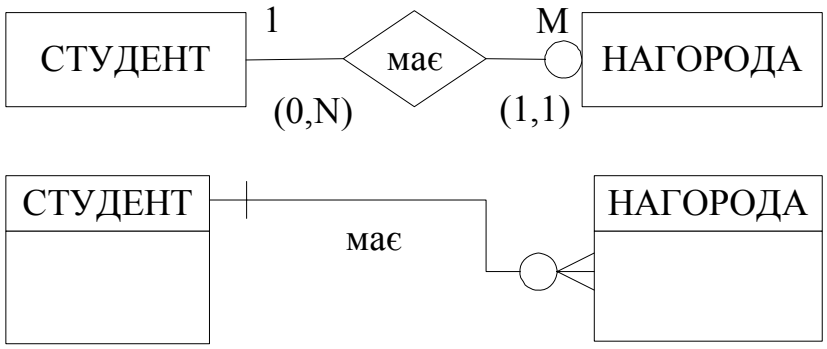


Рис. 5.9. Необов'язкова сутність *Нагорода* в ER-діаграмах П. Чена і "пташина лапка"

Слабкі сутності

Слабкою сутністю називається сутність, яка задовольняє таким умовам:

- залежності від існування сутності з якою вона зв'язана;
- первинний ключ цієї сутності частково або повністю отриман з іншої сутності.

Слабка сутність на діаграмі Чена позначається подвійним прямокутником, а на діаграмі "пташина лапка" невеликими сегментами в кожному з кутів прямокутника.

Приклад. Сутність *Нагорода* є слабкою по відношенню до сутності *Студент*: вона залежить від існування цієї сутності і в її первинний ключ входить первинний ключ сутності *Студент* (рис. 5.10).

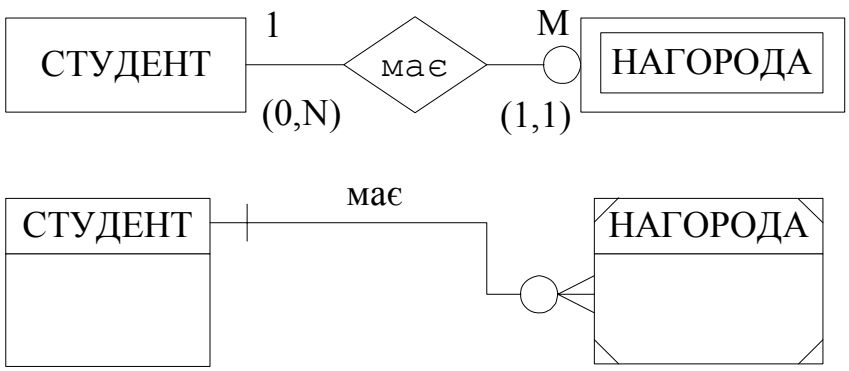


Рис. 5.10 Слабка сутність на діаграмах
П. Чена і "пташина лапка"

Складні зв'язки

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

Приклад. Сутності *Викладач*, *Дисципліна* і *Екзамен* утворюють тернарний зв'язок (рис. 5.11).

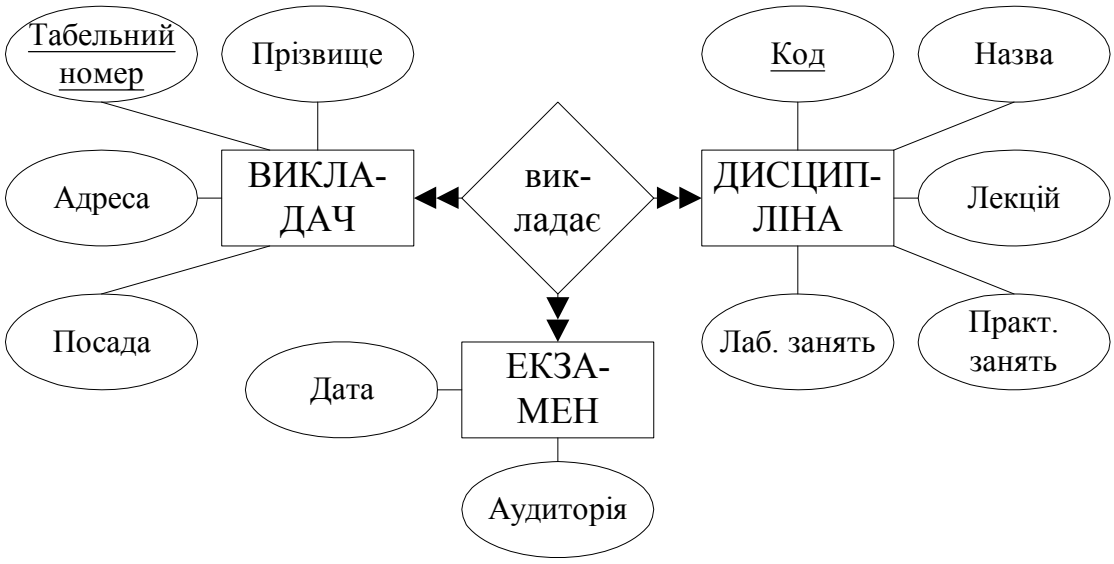


Рис. 5.11. Тернарний зв'язок між трьома сутностями
на діаграмі П. Чена

Рекурсивні зв'язки

Рекурсивний зв'язок має місце, коли є зв'язок між екземплярами одного і того ж набору сутностей.

Приклад. Розглянемо можливі варіанти рекурсивних зв'язків (рис. 5.12).

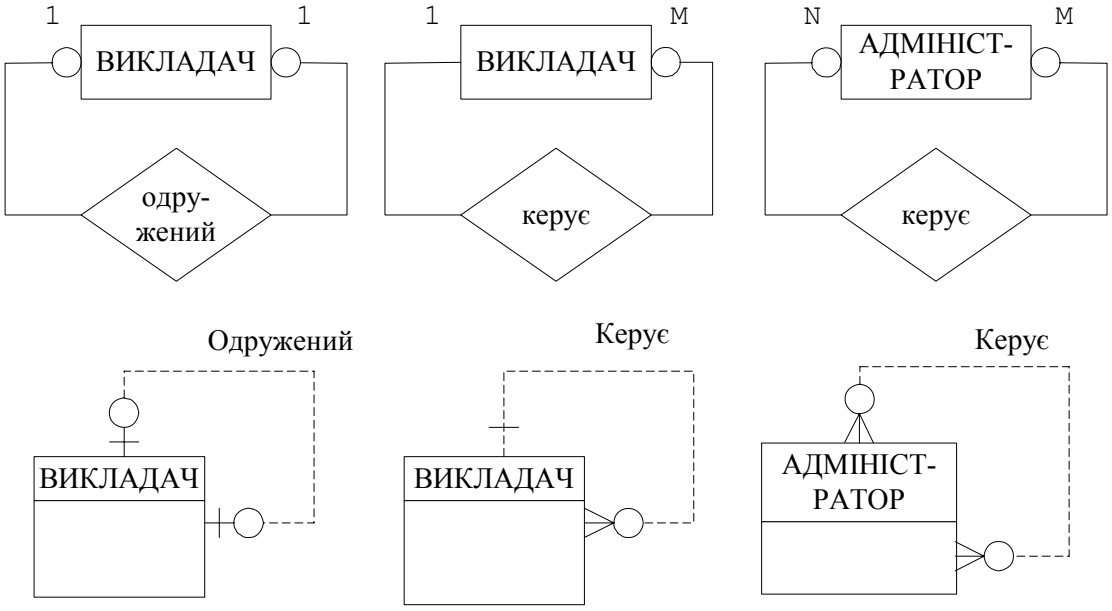


Рис. 5.12. Представлення рекурсивних зв'язків на діаграмі П. Чена і моделі "пташина лапка"

Зв'язок 1:1 представляє висловлювання: "викладач може бути одружений тільки з одним співробітником". Зв'язок 1:M представляє таке висловлювання: "викладач, якщо він є завідуючим кафедрою, керує декількома викладачами, а викладачі мають тільки одного керівника – завідуючого кафедрою". Зв'язок M:N представляє висловлювання: "адміністратор має декілька підлеглих-адміністраторів, і в свою чергу адміністратор має декілька керівників-адміністраторів".

Між двома сутностями може бути декілька зв'язків з різними змістовними навантаженнями.

Приклад. Між сутностями *Викладач* і *Студент* можна встановити такі змістовні зв'язки: *Викладає* і *Керівництво дипломним проектуванням*. Викладач викладає для багатьох

студентів, для кожного студента викладає багато викладачів. Кожен студент обов'язково повинен мати одного керівника дипломного проекту, але необов'язково кожен викладач керує дипломниками (рис. 5.13).

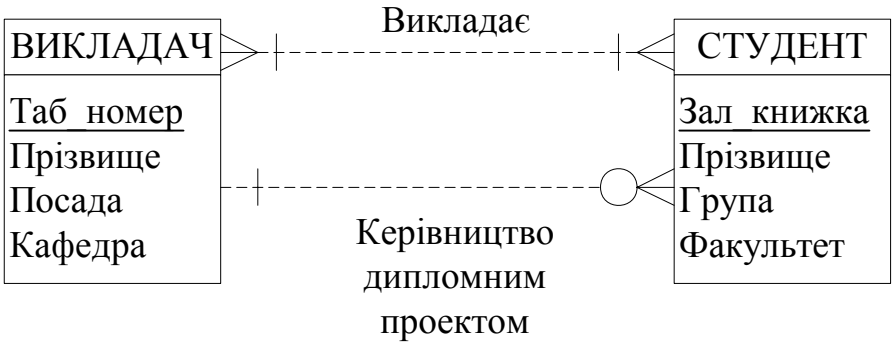


Рис. 5.13. Різні зв'язки між двома сутностями

5.3. Розширена модель "сутність – зв'язок"

Для задоволення нових потреб, що висуваються більш складними застосуваннями, в семантичне моделювання були введені додаткові концепції, що розширюють його можливості. Така модель має назву *розширеної ER-моделі* (Enhanced Entity Relationship, EER-модель). Вона включає всі концепції ER-моделі плюс концепції *уточнення*, *узагальнення*, *агрегування* і *композиції*. Додаткові концепції базуються на таких поняттях, як *суперклас* і *підклас*. Суперклас може мати декілька підкласів. Наприклад підкласи *Викладач*, *Керівник*, *Лаборант* є членами суперкласу *Співробітник*. Це означає, що кожен екземпляр підкласу є в той же час і екземпляром суперкласу. Зв'язок між суперкласом і підкласом відноситься до типу 1:1.

Використання понять суперклас і підклас дозволяє визначити для підкласів власні атрибути і атрибути, що наслідуються від суперкласу. Так, наприклад, підклас *Викладач* повинен мати ті ж атрибути, що і всі *Співробітники*. Однак він має і свої власні атрибути, які не визначені для інших категорій працівників університету. До цих атрибутів можна віднести

вчене звання, *номер диплому про вчене звання*, *кількість навчально-методичних праць* тощо. При відсутності підкласів для об'єкту *Співробітник* слід було б вводити атрибути, які б мали невизначене значення для інших співробітників (наприклад для лаборантів). Підклас може мати свої власні зв'язки, які не підходять для всіх екземплярів суперкласу. Наприклад, *Викладач* може мати підкласи *Професор*, *Доцент*, *Асистент*. Підклас наслідує не тільки атрибути, але і всі зв'язки суперкласу.

Уточнення це процес збільшення різниці між окремими екземплярами об'єкта за рахунок 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 їх 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 набора 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 (позначається *Or*). Зв'язки, які не перетинаються позначаються символом "*G*". Наприклад, співробітник може працювати або на посаді доцента, або на посаді професора, і не може бути одночасно і професором, і доцентом.

Якщо підкласи суперкласу *перетинаються*, то це означає, що будь-який екземпляр сутності може бути елементом декількох з підкласів (позначається *And*). Зв’язки які перетинаються позначаються символом "*Gs*". Наприклад, завідуючий кафедрою проводить заняття, і одночасно виконує

обов'язки викладача і керівника. На рис. 5.14 показана ієрархія сутностей з підкласами, що перетинаються і не перетинаються.

Приклад. Відношення суперклас – підклас.

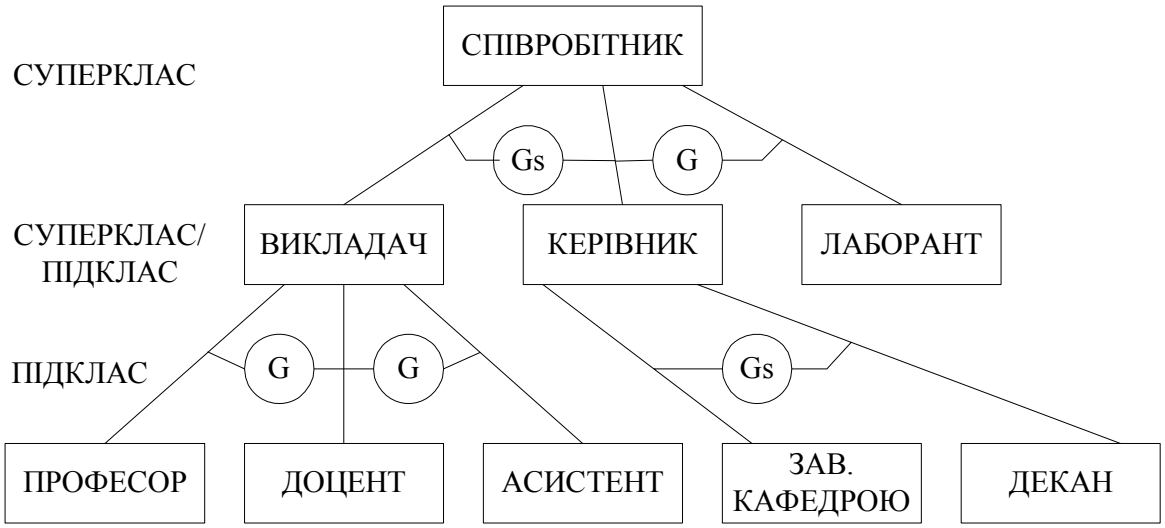


Рис. 5.14. Діаграма з використанням понять суперклас і підклас

Не кожен елемент суперкласу повинен бути елементом одного з підкласів.

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

Приклад. Студент обов'язково займається або на денній, або на заочній, або на вечірній формі навчання, або екстернатом (рис. 5.15).

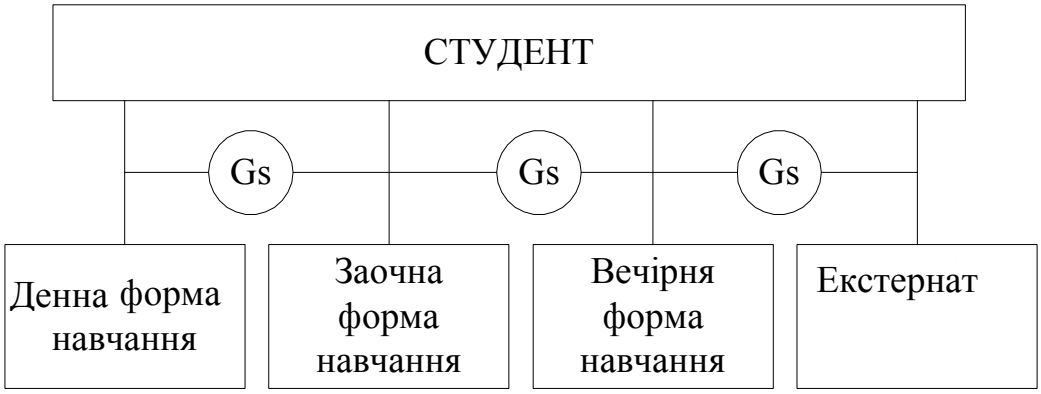


Рис. 5.15. Діаграма зв'язку суперкласу з підкласом
з обов'язковою участю

Зв'язок суперкласу з підкласом з *необов'язковою участю*, вказує на те, що деякі елементи суперкласу можуть не належати жодному з підкласів (*Optional*). Наприклад, можуть бути викладачі, які не займають посади професора, доцента або асистента (наприклад старший викладач).

Введення понять суперкласів і підкласів дозволяє уникнути опису різних екземплярів сутності, які можуть мати різні атрибути. Це може привести до появи великої кількості незаповнених атрибутів. До того ж індивідуальні атрибути можуть показати зв'язки, які притаманні тільки для конкретних екземплярів, а не всім екземплярам сутностей.

Приклад. Екземпляри сутності *Викладач*, які належать до викладачів на посаді професора, у порівнянні з викладачами на посаді асистента, мають такі додаткові атрибути: вчене звання, вчений ступінь і т.ін. Крім того, вони можуть бути зв'язані індивідуальними атрибутами із сутністю *Аспірант*.

Один підклас може бути зв'язаний з декількома суперкласами, які в свою чергу є підкласами одного спільного для них суперкласу.

Приклад. Підкласи *Викладач* і *Керівник* наслідують суперклас *Співробітник*. Підклас *Завідуючий кафедрою* є одночасно екземпляром суперкласу *Викладач* і суперкласу *Керівник* (рис. 5.16).

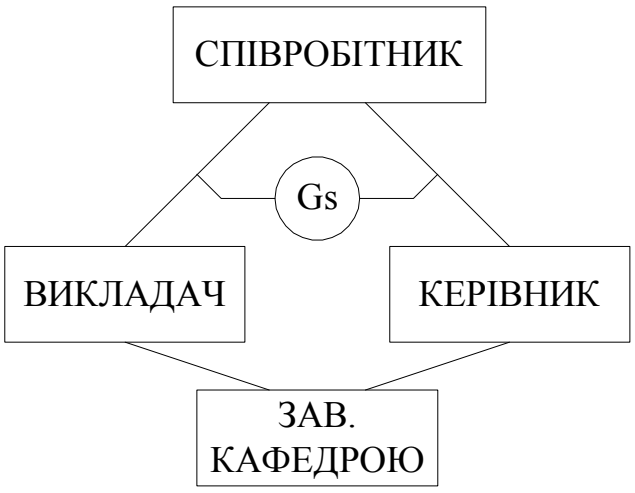


Рис. 5.16. Діаграма з підкласом, що сумісно використовується

Підклас є *категорією*, якщо він зв'язаний одразу з декількома суперкласами різних типів. Категорія може бути з повною участю і з частковою участю екземплярів. Повна участь передбачає, що кожен екземпляр всіх суперкласів повинен бути представлений у даній категорії. При частковій участі присутність в категорії всіх екземплярів всіх суперкласів необов'язкова.

Введення понять суперкласів і підкласів дозволяє ввести в проект більший обсяг семантичної інформації. Наприклад, твердження "Асистент є викладачем" дозволяє встановити певні ієрархічні відношення між об'єктами *Асистент* і *Викладач*. EER-діаграми повинні використовуватися, якщо структура даних є занадто складною, для того щоби її можна було легко представити з використанням ER-діаграм.

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

Агрегування являє собою зв'язок типу "входить в склад" або "включає" між двома сутностями, одна з яких представляє "ціле", а інша – "частину".

Композиція – це особлива форма агрегування, яка представляє залежність між сутностями, що характеризуються повною приналежністю і співпаданням термінів існування між "цілим" і "частиною".

5.4. Проблеми побудови моделей "сутність – зв'язок"

При недостатньому розумінні суті встановлених зв'язків може бути створена модель, яка не буде повною мірою відображати зв'язки між реальними об'єктами. Визначають *дефекти з'єднання*, які виникають при невірній інтерпретації змісту деяких зв'язків: дефекти розгалуження і дефекти розриву.

Дефекти розгалуження мають місце, коли модель вірно відображає зв'язки між сутностями, але шлях між окремими

сутностями визначений неоднозначно. Цей дефект виникає в тому випадку, коли два або більше зв'язків типу 1:M виходять з однієї сутності.

Приклад. Розглянемо такі зв'язки: на факультеті займається багато студентів, у склад факультету входить багато груп (рис. 5.17). Ці зв'язки вірно відображають зміст предметної області, але при спробі з'ясувати, в яких групах займаються конкретні студенти, виникають проблеми. Із сутності *Факультет* виходять два зв'язки 1:M.

Усунути цю проблему можна шляхом перебудови моделі для представлення вірної взаємодії цих сутностей (рис. 5.18).

Рис. 5.17. Приклад дефекту розгалуження в ER-моделі

Рис. 5.18. 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 у разі, коли існує один або декілька зв'язків з мінімальною потужністю рівною 0, яка визначає необов'язкову участь, і ці зв'язки складають частину шляху між взаємозв'язаними сутностями.

Приклад. Розглянемо такі зв'язки: в склад факультету входить багато кафедр, кожна кафедра може відповідати за декілька комп'ютерних класів (від 0 до M) (рис. 5.19). Тобто деякі кафедри можуть не бути відповідальними за комп'ютерний клас. У свою чергу комп'ютерний клас може

підпорядковуватися певній кафедрі, а може підпорядковуватися безпосередньо факультету (факультетський комп'ютерний клас). Ці зв'язки вірно відображають зміст предметної області, але при спробі з'ясувати, які комп'ютерні класи підпорядковані певному факультету, виникають проблеми. Зв'язок між сутностями *Кафедра* і *Комп'ютерний клас* передбачає необов'язкову участь сутностей, і він є частиною шляху між сутностями *Факультет* і *Кафедра*.

Усунути цю проблему можна шляхом введення додаткового зв'язку між сутностями *Факультет* і *Комп'ютерний клас* (рис. 5.20).

Рис. 5.19. Приклад дефекту розриву в ER-моделі

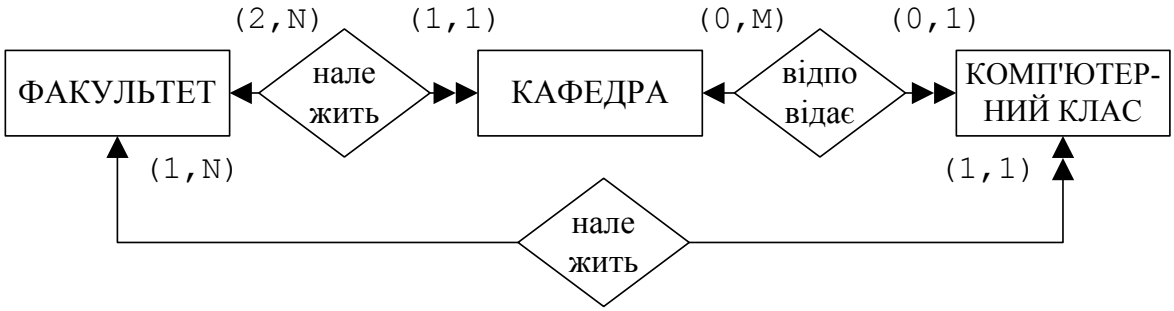


Рис. 5.20. Усунення дефекту розриву в прикладі ER-моделі

5.5. Приклад побудови моделі "сутність – зв'язок"

Процес концептуального проектування БД є ітеративний й заснований на операціях і процедурах, що повторюються. Спочатку створюється базова ER-модель певної предметної області. При дослідженні цієї моделі як правило з'являться додаткові сутності, атрибути і зв'язки. Після цього ER-модель буде змінюватися. Процес змін повторюється до тих пір, поки концептуальна модель не буде відображати предметну область.

Розробники БД на основі опитування фахівців визначають сутності, атрибути, зв'язки у предметній області, що розглядається, досліджені документів, які застосовуються в роботі на підприємстві.

Розглянемо приклад створення ER-моделі предметної області ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД (ВНЗ).

Задачі інформаційної системи

Мета створення бази даних полягає у такому:

- забезпечення адміністрації університету довідковою інформацією по факультетах, кафедрах, спеціальностях, викладачах, студентах і дисциплінах, що викладаються у ВНЗ;
- контроль успішності студентів.

Аналіз предметної області

У результаті дослідження й аналізу предметної області були визначені такі сутності, атрибути і первинні ключі (табл. 5.2).

Зв'язки сутностей визначаються на основі бізнес-правил, які побудовані з урахуванням організаційної структури та операцій, що виконуються в системі:

- в університеті існує декілька факультетів;
- на факультеті навчаються групи студентів за певними спеціальностями;
- у склад групи входять студенти;
- факультет об'єднує декілька кафедр;
- на кожній кафедрі працює 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, по якій викладача ще не призначено).

Таблиця 5.2

Сутності, атрибути і первинні ключі предметної області ВНЗ

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*

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

Дослідження предметної області виявило, що всі сутності є сильними, а зв'язки між сутностями – неідентифікуючими. Зв'язок між сутностями *Студент* і *Дисципліна* має атрибут *Оцінка*.

Побудова ER-діаграми

На основі задач, які були поставлені перед інформаційною системою, і на основі аналізу предметної області, побудована ER-діаграма ВНЗ (рис. 5.21).

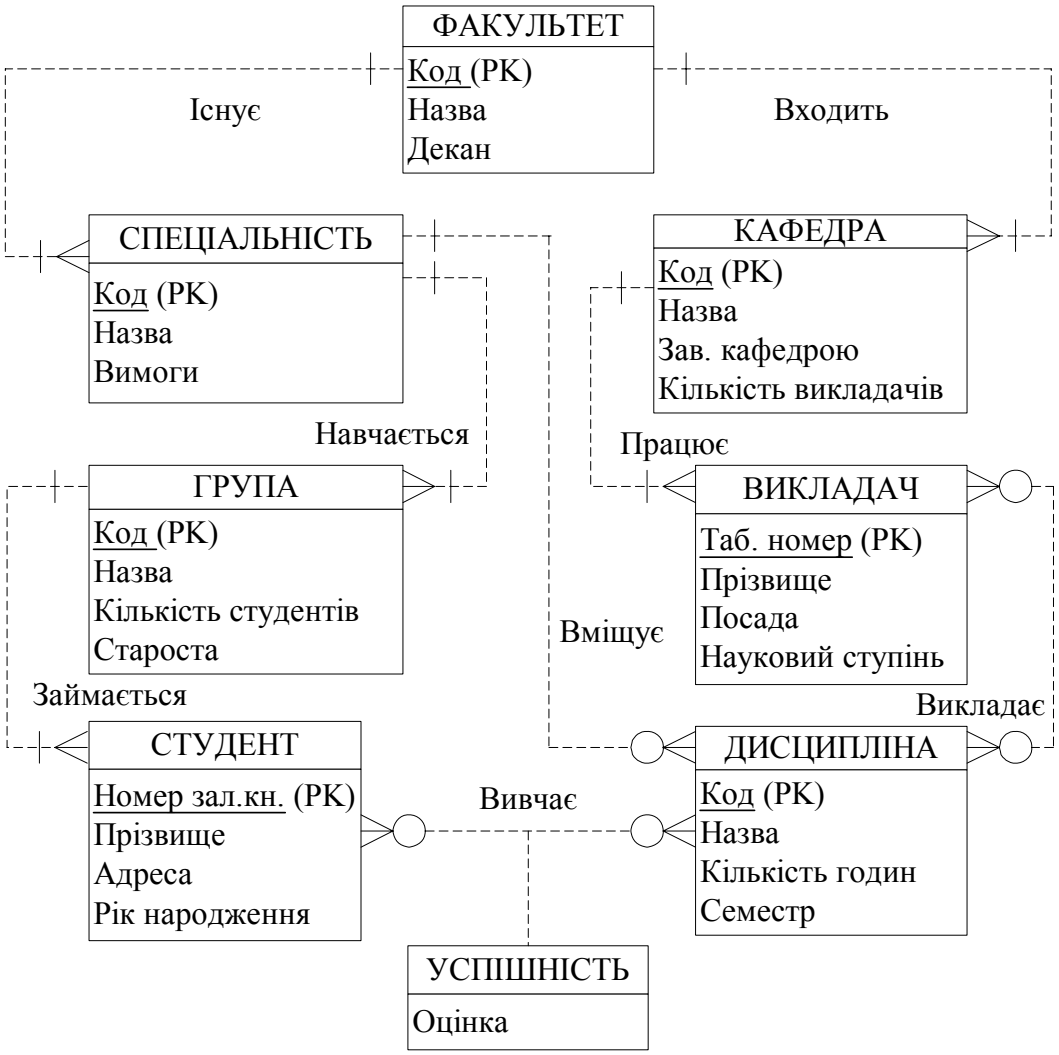


Рис. 5.21. ER-діаграма предметної області
ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

Розроблений концептуальний проект необхідно перевірити на збитковість та на відповідність транзакціям користувачів.

Перевірка на збитковість передбачає перевірку ER-моделі з метою виявлення збиткових даних і вилучення їх, в тому випадку, якщо вони визначені. Збиткові зв'язки виявляються в тому, що між двома сутностями є декілька

шляхів і вони дублюють один одний (це не відноситься до зв'язків, які представляють різні асоціації).

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

- перевірка того, чи представляє модель всю інформацію (сутності, атрибути, зв'язки), яка необхідна для кожної транзакції;
- перевірка по ER-діаграмі маршруту кожної транзакції.

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

Слід звернути увагу на те, що розроблений концептуальний проект не є єдиним проектом, який відповідає поставленій задачі. Можливі варіанти розробки системи із застосуванням інших зв'язків між сутностями, або із застосуванням розширеної ER-моделі.

Застосування ER-діаграм дозволяє забезпечити просте і наглядне уявлення про головні логічні об'єкти БД і про зв'язки, які між цими об'єктами існують. Також до переваг ER-діаграми слід віднести те, що вони добре інтегрують з реляційною моделлю.

Недоліком ER-моделей є те, що вони мають недостатні можливості для представлення відношень і обмежень, можуть бути складні при наявності багатьох об'єктів, не мають засобів для опису операцій маніпулювання даними.

Контрольні запитання

1. Дати визначення *сутності*. Що таке сильна сутність, слабка сутність?
2. Дати визначення *атрибута*. Що таке простий атрибут, складний атрибут, композитний атрибут?
3. Дати визначення *ступеня зв'язку*. Що таке кардинальне число?
4. Що таке ідентифікаційний і неідентифікаційний зв'язок?

5. Навести символьні позначення, які застосовуються в діаграмах "сутність – зв'язок".
6. Пояснити, що таке *підтипи сутностей* і навести приклади.
7. Як відображається наслідування на діаграмах "сутність – зв'язок"?
8. Навести приклади зв'язків 1:N для таких різновидів зв'язків: *необов'язково-необов'язково*, *необов'язково-обов'язково*, *обов'язково-необов'язково*, *обов'язково-обов'язково*.
9. Дати визначення *рекурсивного зв'язку* і навести приклади рекурсивних зв'язків 1:1, 1:N, M:N.
10. Пояснити переваги і недоліки ER-моделі.

Глава 6. ЛОГІЧНЕ ПРОЕКТУВАННЯ БАЗ ДАНИХ

6.1. Етапи логічного проектування

Логічне проектування виконується для певної моделі даних. Для реляційної моделі даних логічне проектування полягає у створенні реляційної схеми, визначенні числа і структури таблиць, формуванні запитів до БД, визначенні типів звітних документів, розробці алгоритмів обробки інформації, створенні форм для вводу і редагування даних в БД і рішенні цілого ряду інших задач. Концептуальні моделі за певними правилами перетворюються в логічні моделі даних. Коректність логічних моделей перевіряється за допомогою *правил нормалізації*, які дозволяють переконатися в структурній узгодженості, логічній цілісності і мінімальній збитковості прийнятої моделі даних. Модель також перевіряється з метою виявлення можливостей *виконання транзакцій*, які будуть задаватися користувачами. Проектування являє собою циклічний процес. Етапи логічного проектування наведені на рис. 6.1.

Рис. 6.1. Етапи логічного проектування бази даних

6.2. Спрощення концептуальної моделі

Першим кроком спрощення концептуальної моделі є попередні перетворення з метою усунення зв'язків, які є несумісними з реляційною моделлю.

На цьому етапі виконуються такі операції:

- вилучення двосторонніх зв'язків M:N;
- вилучення складних зв'язків;
- вилучення багатозначних атрибутів;
- вилучення рекурсивних зв'язків;
- вилучення зв'язків з атрибутами.

Вилучення двосторонніх зв'язків "багато до багатьох"

Перетворення зв’язку "багато до багатьох" виконується шляхом введення проміжної сутності із заміною одного зв'язку M:N двома зв'язками 1:N з новою сутністю.

Приклад. *Викладач* може викладати багато *Дисциплін*, одну *Дисципліну* викладає багато *Викладачів* (рис. 6.2).

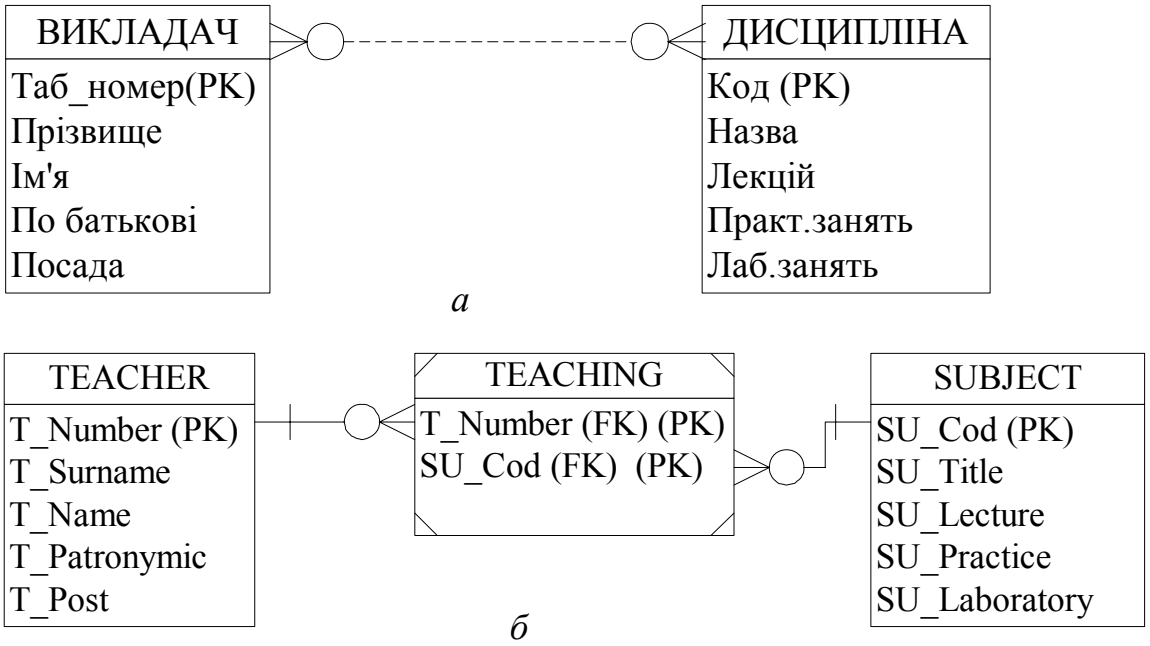


Рис. 6.2. Перетворення зв'язку "багато до багатьох": *а* – зв'язок M:N; *б* – результат перетворення – два зв'язку 1:N

У результаті перетворення отримана нова сутність, яка є слабкою і залежить від двох інших сутностей. Її первинний ключ складається з первинних ключів двох сутностей, а кожен атрибут окремо є вторинним ключем.

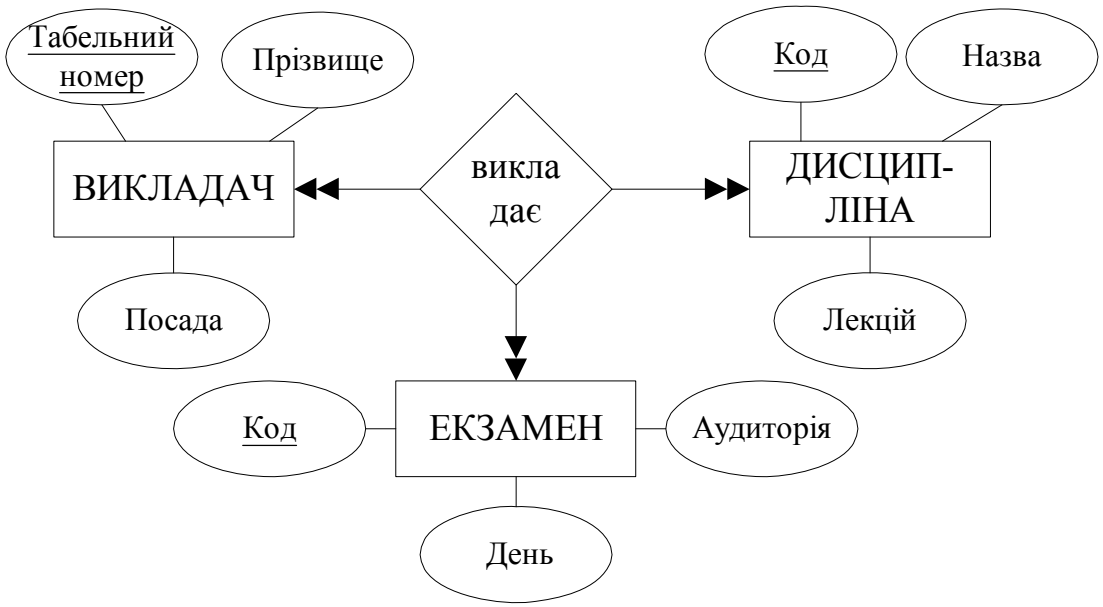
Вилучення складних зв'язків

Для вилучення складних зв'язків виконуються такі операції:

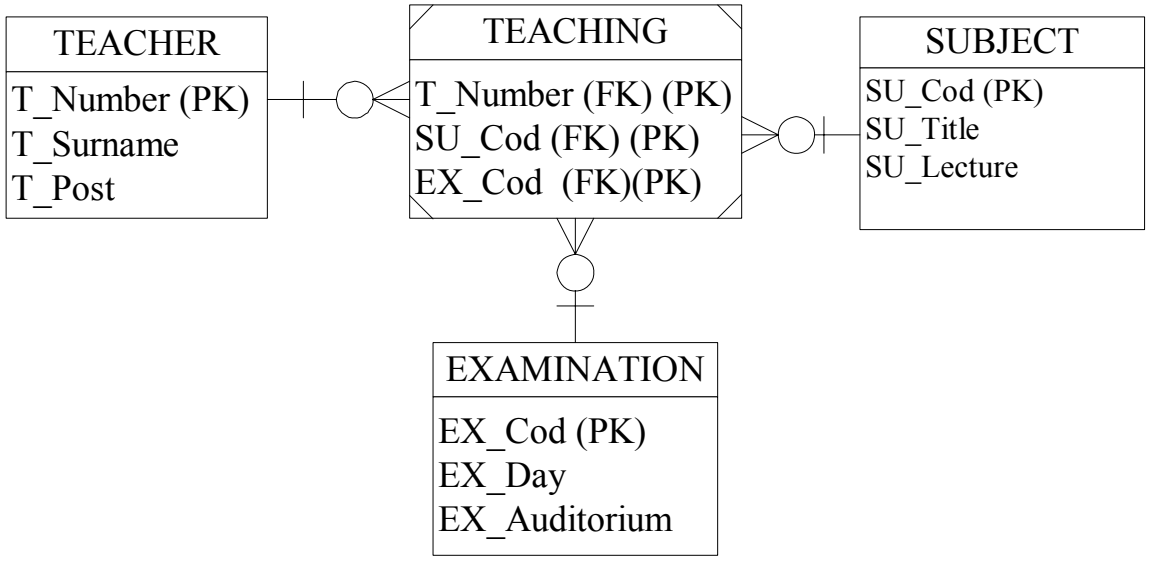
- у модель вводиться нова сутність;
- складний зв'язок замінюється бінарними зв'язками "один до багатьох" зі знов створеною сутністю;

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

Приклад. *Викладач* може викладати багато *Дисциплін*, одну *Дисципліну* викладає багато *Викладачів*. З *Дисципліни* *Викладач* проводить *Екзамен* (рис. 6.3).



а



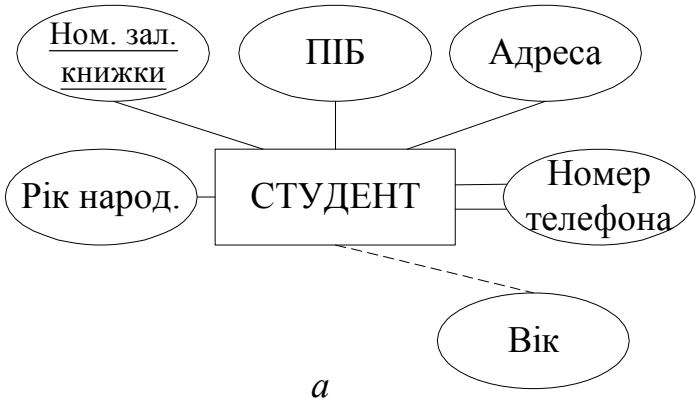
б

Рис. 6.3. Представлення тернарного зв'язку бінарними зв'язками: *а* – складний зв'язок *Викладає;* *б* – декомпозиція складного зв'язку на три двосторонні зв'язки і нову сутність *Екзамен*

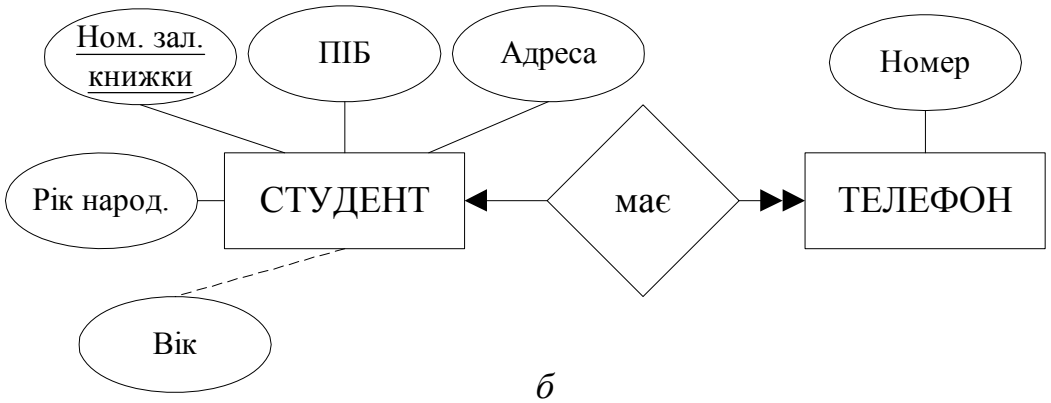
Вилучення багатозначних атрибутів

Якщо в концептуальній моделі даних присутній багатозначний атрибут, то може бути виконана декомпозиція цього атрибуту для визначення деякої сутності.

Приклад. *Студент* може мати декілька телефонів (рис.6.4).



а



б

Рис.6.4. Вилучення багатозначного атрибуту:
а – сутність *Студент* з багатозначним атрибутом *Номер* т*елефону*;
б – нова сутність *Телефон*

Вилучення рекурсивних зв'язків

На етапі спрощення концептуальної моделі рекурсивні зв'язки 1:1 і 1:М (рис. 6.5) можуть бути перетворені у одне відношення. У випадку, коли є необов'язкова сутність з боку "багато" для зв'язку 1:М для зменшення пустих значень

створюється нове відношення. Зв'язок M:N перетворюється на дві сутності (рис.6.6).

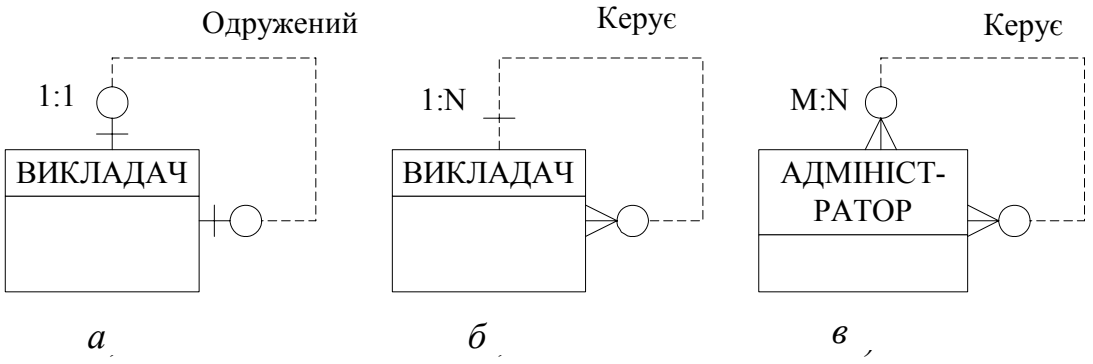


Рис. 6.5. Види рекурсивних зв'язків: *а* – 1:1; *б* – 1:N: *в* – M:N

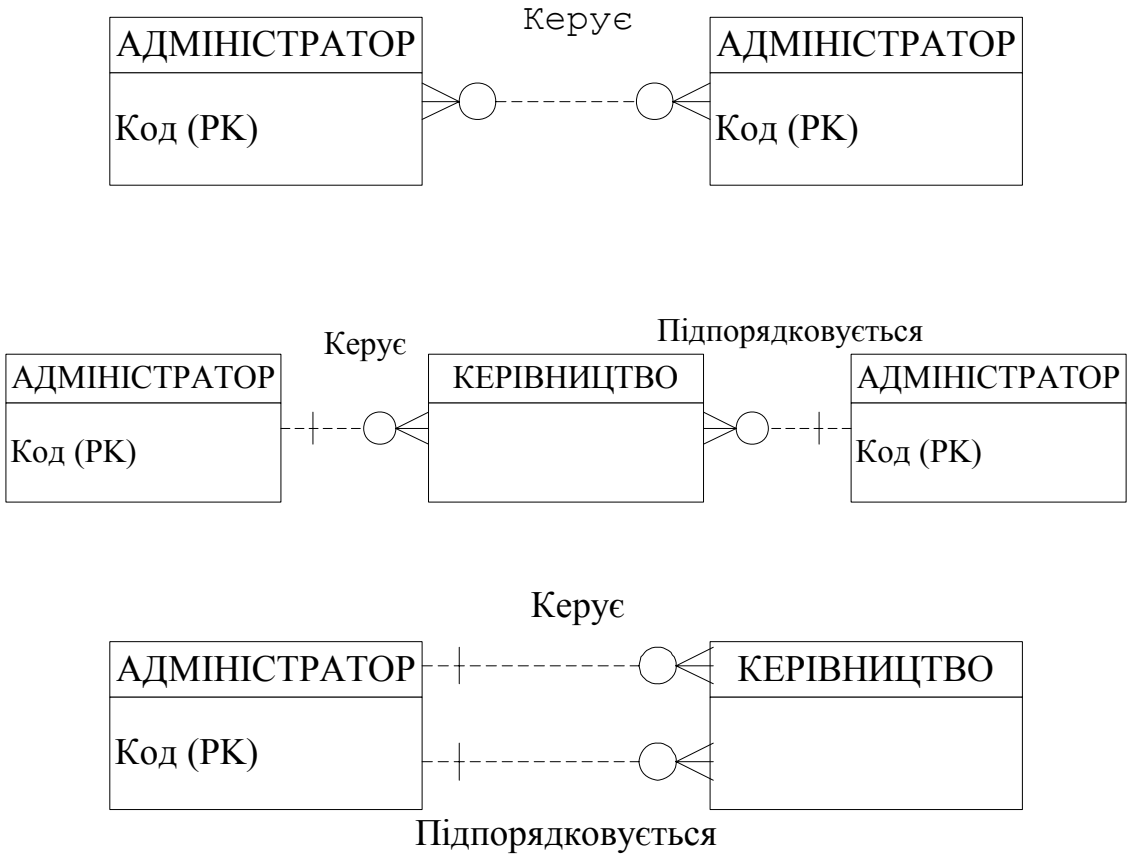


Рис. 6.6. Етапи послідовного перетворення рекурсивного зв'язку M:N

Вилучення зв'язків з атрибутами

Вилучення зв'язків з атрибутами виконується шляхом додавання у модель нової сутності для відношення M:N з атрибутами зв'язку. Для відношення 1:M атрибути зв'язку передаються у сутність "багато" без створення нової сутності.

Приклад. Розглянемо сутності *Студент-Дисципліна* (рис.6.7).

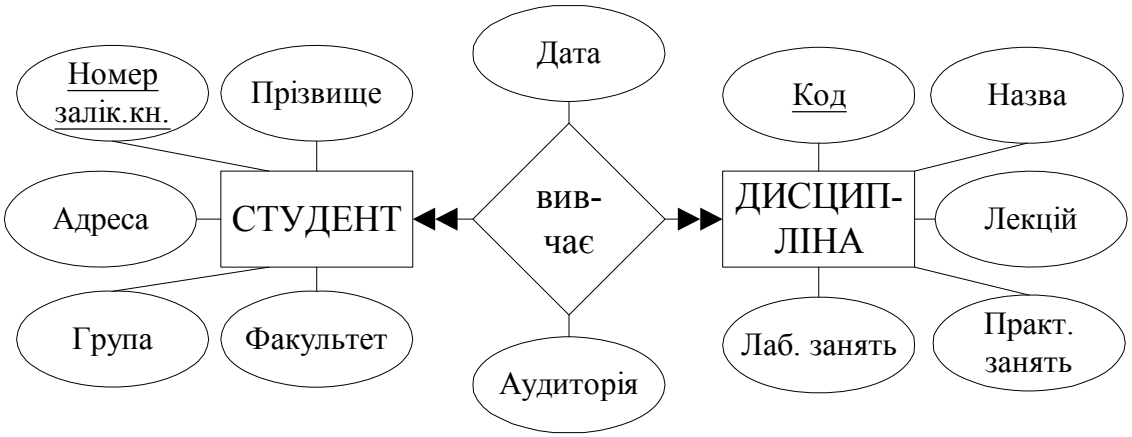


Рис. 6.7. Зв'язок "багато до багатьох" з атрибутами зв'язку *Дата* і *Аудиторія*

У результаті перетворення зв'язку з атрибутами отримано реляційну схему (рис. 6.8).

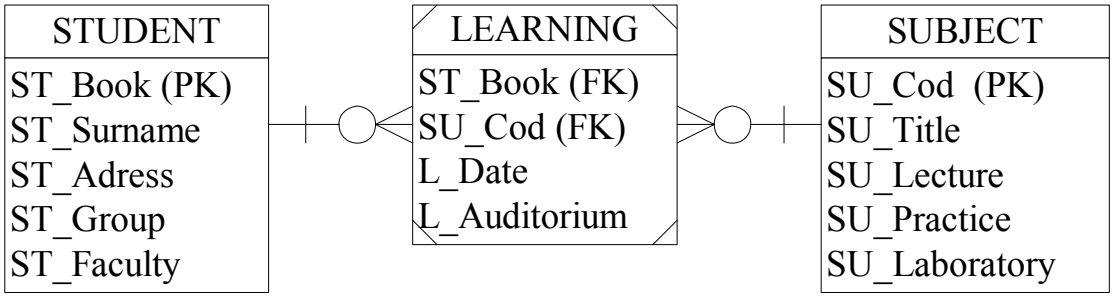


Рис. 6.8. Представлення атрибутів зв'язку *L_Date* і *L_Auditorium* у новому відношенні *Learning*

Спрощення концептуальної моделі передбачає також вилучення збиткових зв'язків. Збиткові зв'язки

характеризуються тим, що одна і та ж інформація може бути отримана не тільки через них, але і через інші зв'язки.

Після спрощення в концептуальній моделі можуть бути присутні тільки такі елементи:

- об'єкти і атрибути;
- зв'язки типу 1:1 і 1:M;
- зв'язки типу суперклас-підклас.

6.3. Методика перетворення ER-діаграм в реляційні структури

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

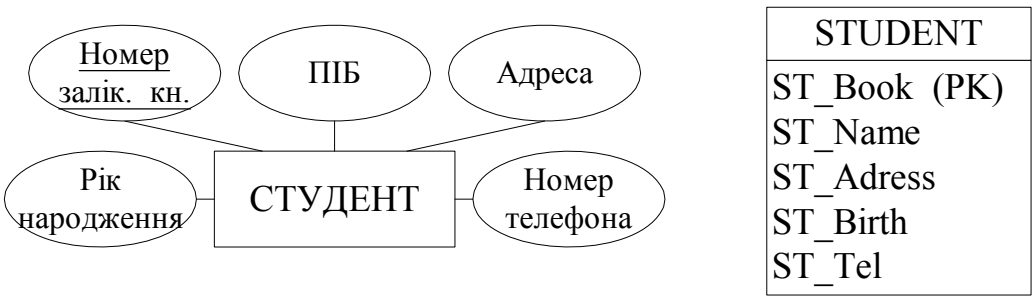
Розглянемо правила перетворення ER-моделі в реляційну модель.

Сутності і атрибути

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

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

Приклад. Розглянемо сутність *Студент* (рис.6.9).



а *б*

Рис. 6.9. Перетворення сутності *Студент* (*а*) у відношення *Student* (*б*)

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

Приклад. Розглянемо перетворення сильної сутності *Студент* і слабкої сутності *Нагорода* (рис. 6.10).

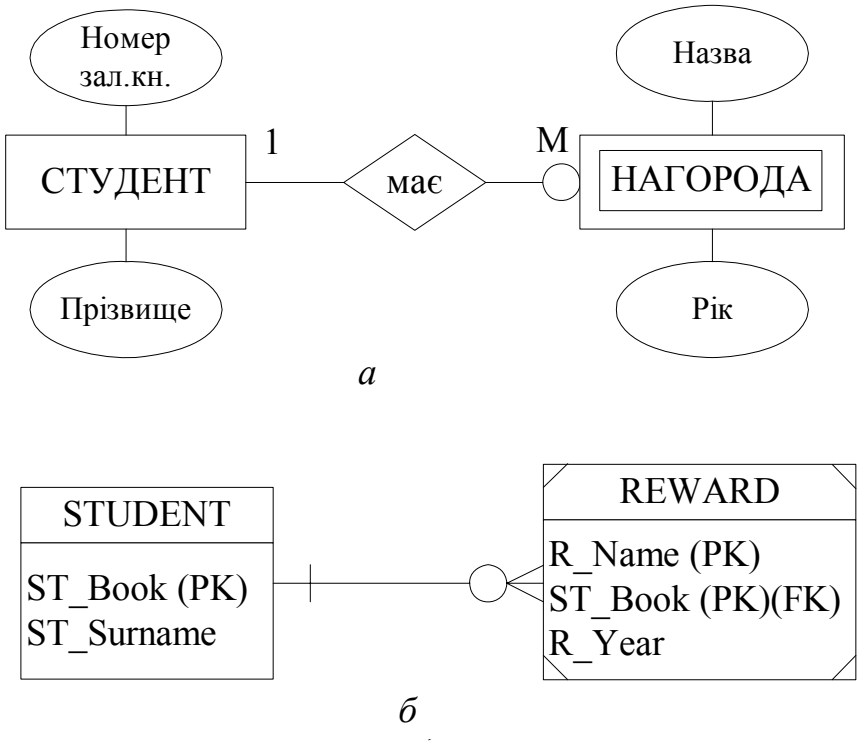


Рис. 6.10. Перетворення сильної сутності *Студент* і слабкої сутності *Нагорода* з ідентифікуючим зв'язком між ними (*а*) у зв'язані відношення *Student* і *Reward* (*б*)

Зв'язки

Після перетворення концептуальної моделі залишаються такі типи зв'язків:

- "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.

Зв'язки "один до одного"

В концептуальних моделях даних визначають такі обмеження ступеня участі сутностей:

- 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. 6.11).

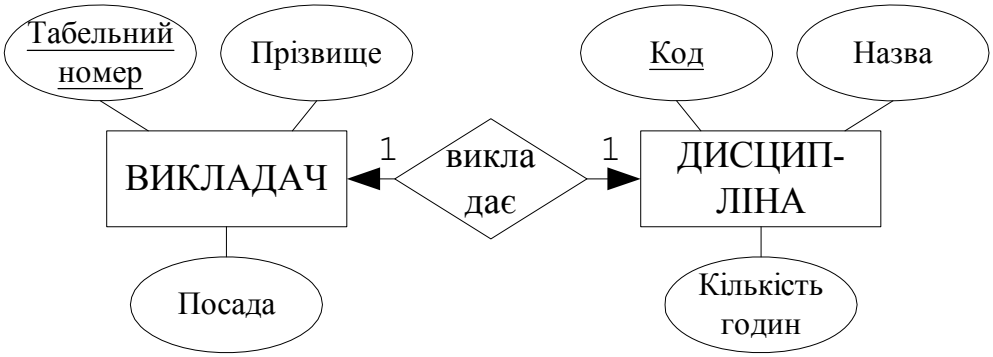
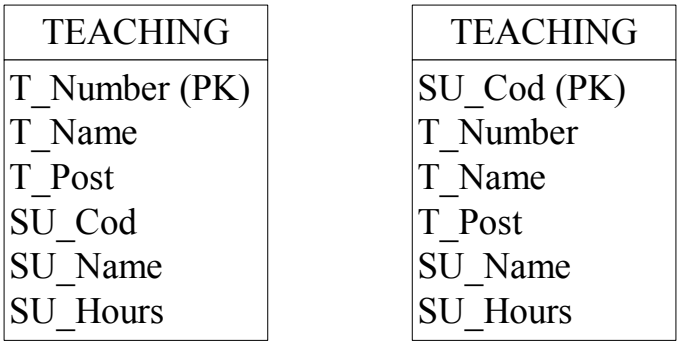


Рис. 6.11. Зв'язок 1:1 між сутностями *Викладач* і *Дисципліна*

1. *Обов'язкова участь для обох сутностей*

Припустимо, що кожен викладач обов'язково викладає одну дисципліну і кожну дисципліну обов'язково викладає один викладач. В цьому випадку реляційна структура буде складатися з одного відношення і мати один з таких варіантів (рис. 6.12).



а *б*

Рис. 6.12. 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. В цьому випадку сутність, яка є необов'язковою (*Дисципліна*) виступає в якості батьківської сутності, а обов'язкова сутність визначається як дочірня (*Викладач*). Реляційна структура показана на рис. 6.13.

Зовнішній атрибут SU_Cod також може бути ключем для *Викладача*.

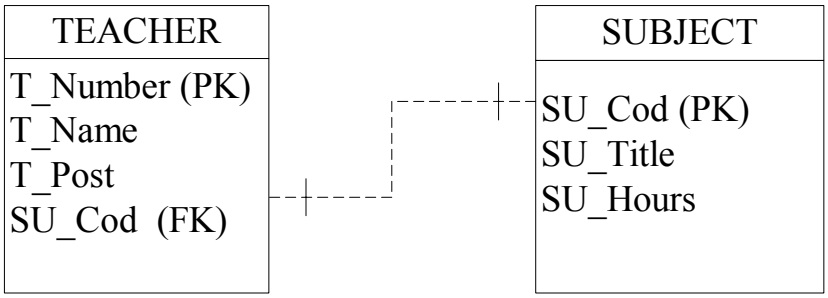


Рис.6.13. Перетворення зв'язку 1:1 з обов'язковою участю сутності *Викладач* і необов'язковою участю сутності *Дисципліна*

3. *Необов'язкова участь для обох сутностей*

Припустимо, що необов'язково кожен викладач викладає дисципліни і необов'язково за кожною дисципліною закріплений викладач. В цьому випадку можливі три реляційні структури (рис. 6.14).

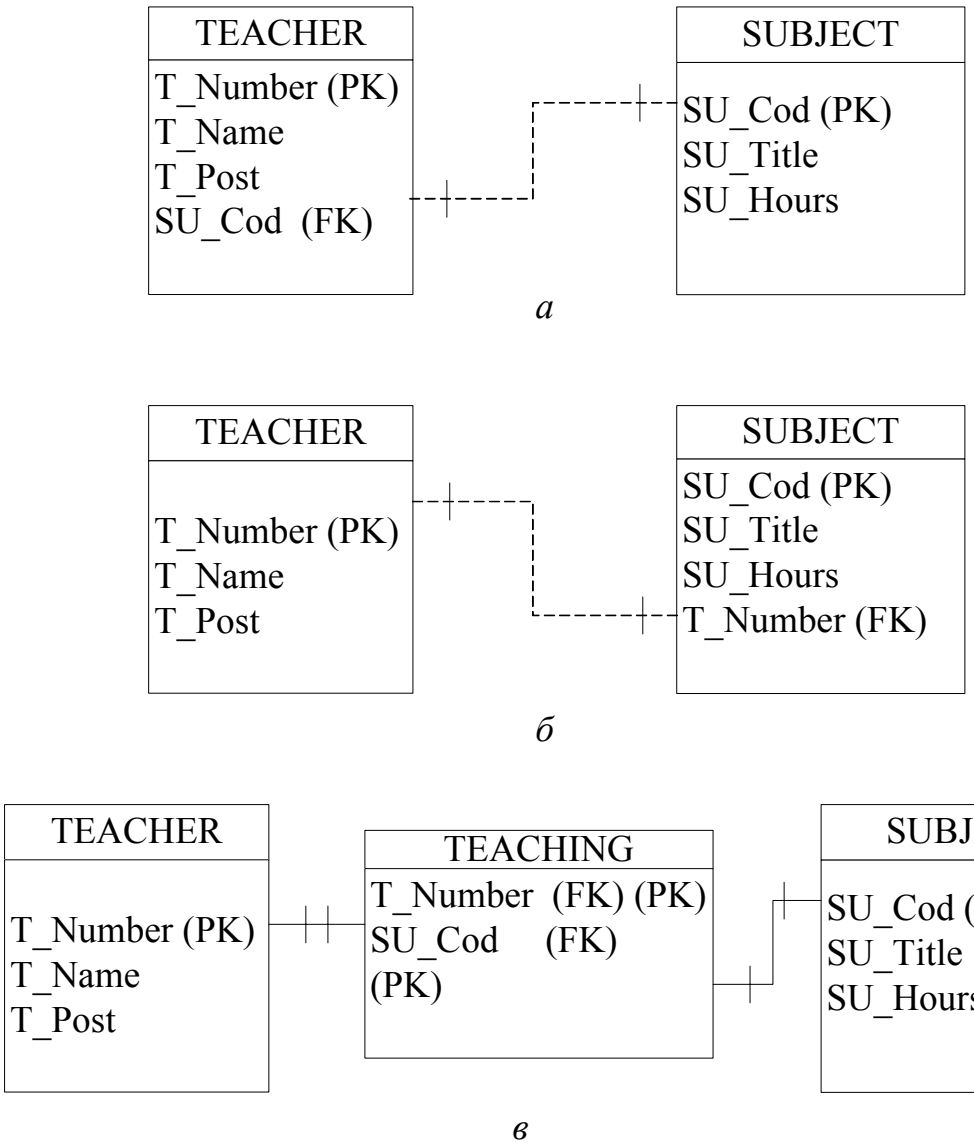


Рис. 6.14. Варіанти реляційних схем відношень для перетворення зв'язку 1:1 з необов'язковою участю для обох сутностей

Зв'язки "один до багатьох"

У кожне відношення, яке 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 (FOREIGN KEY, FK).

Для моделювання необов'язкового типу зв'язку у атрибутів, що відповідають зовнішньому ключу, встановлюється властивість допустимості невизначених значень (NULL). У разі

обов'язкового типу зв'язку атрибути набувають властивості відсутності невизначених значень (NOT NULL).

Приклад. Розглянемо можливі варіанти перетворення зв'язку між сутностями *Викладач* і *Дисципліна* (рис. 6.15).

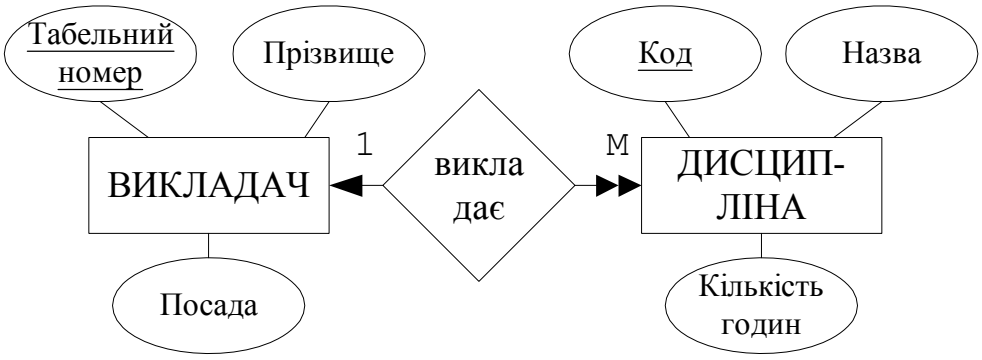


Рис. 6.15. Зв'язок 1:M між сутностями *Викладач* і *Дисципліна*

1. Необов'язкова участь сутності *Викладач* і обов'язкова участь сутності *Дисципліна* (рис. 6.16).

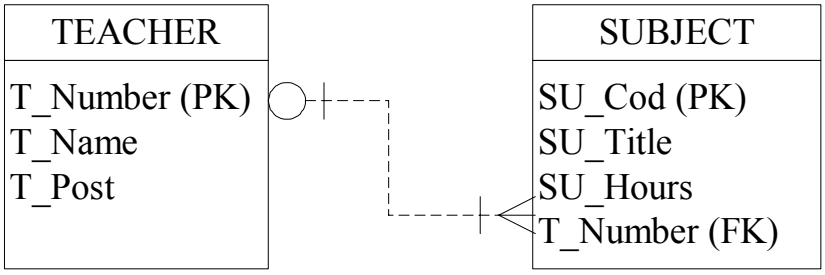


Рис. 6.16. Перетворення зв'язку 1:M з необов'язковою участю сутності *Викладач* і обов'язковою участю сутності *Дисципліна*

2. Необов'язкова участь сутності *Викладач* і необов'язкова участь сутності *Дисципліна* (рис. 6.17).

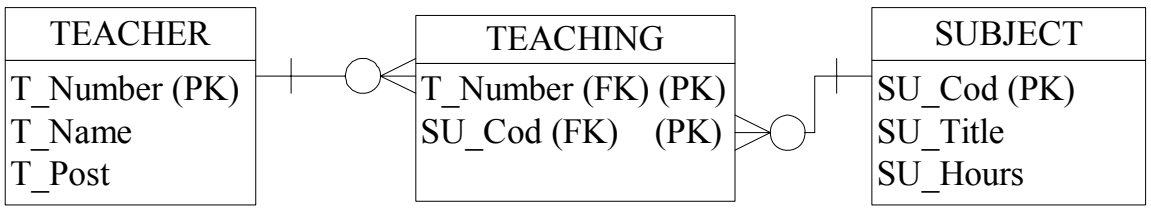


Рис. 6.17. Перетворення зв'язку 1:M з необов'язковою участю сутності *Викладач* і необов'язковою участю сутності *Дисципліна*

Зв'язки "багато до багатьох"

Для кожного зв'язку M:N необхідно створювати додаткове відношення, яке представляє цей зв'язок і включати в нього всі атрибути, які входять в склад цього зв'язку. Копії атрибутів первинного ключа сутностей, які беруть участь у зв'язку, передаються у нове відношення для використання в якості зовнішніх ключів. Ці зовнішні ключі утворюють також первинний ключ нового відношення.

Приклад. Розглянемо можливі варіанти перетворення зв'язку між сутностями *Викладач* і *Дисципліна* (рис. 6.18).

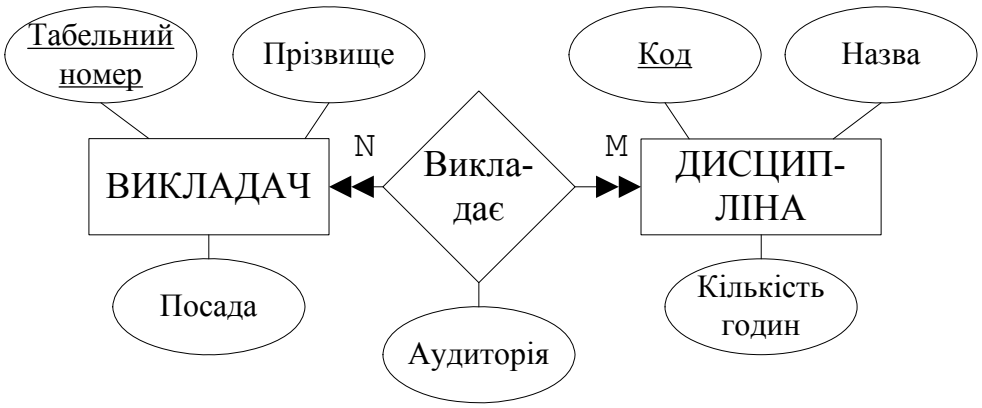


Рис. 6.18. Зв'язок N:M між сутностями *Викладач* і *Дисципліна*

В цьому випадку існує єдина схема перетворення (рис. 6.19).

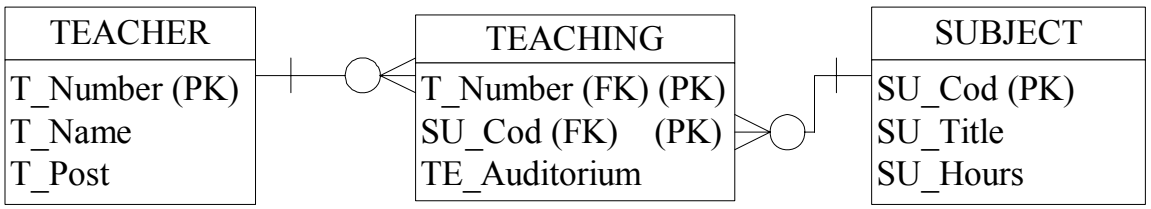


Рис. 6.19. Перетворення зв'язку N:M у реляційну схему

Інші види зв'язків

Для *рекурсивних зв'язків* 1:1 виконуються правила визначені раніше для зв'язку між двома сутностями 1:1. Для

рекурсивного зв'язку 1:1 з обов'язковою участю двох сторін, реляційна схема представляється у вигляді одного відношення з двома копіями первинного ключа (див. рис. 6.12). Одна копія відповідає зовнішньому ключу. Для рекурсивного зв'язку 1:1 з обов'язковою участю тільки однієї сторони створюється або одне відношення, або нове відношення, яке відображає цей зв'язок (див. рис. 6.13). Для рекурсивного зв'язку 1:1 з необов'язковою участю обох сторін створюється нове відношення (див. рис. 6.14).

Для *складних типів зв'язків* створюється відношення, яке відображає цей зв'язок і включає всі атрибути, які входять в склад цього зв'язку. Копії атрибутів первинного ключа сутностей, які беруть участь у зв'язку, передаються у нове відношення для використання в якості зовнішніх ключів. Ці зовнішні ключі утворюють також первинний ключ нового відношення (див. рис. 6.3).

Для *багатозначного атрибуту* створюється нове відношення, яке відповідає багатозначному атрибуту, і в це нове відношення передається первинний ключ сутності для використання в якості зовнішнього ключа (див. рис. 6.4).

Зв'язки "суперклас – підклас"

Для виконання перетворення зв'язку типу суперклас – підклас у реляційну модель необхідно враховувати також обмеження ступеня участі у зв'язку (Mandatory або Optional) і обмеження неперетинання (And або Or). Можливі чотири сполучення, перетворення яких дає чотири реляційні схеми. На схему також впливає те, чи беруть участь підкласи в різних зв'язках, кількість сутностей в зв'язку і т.ін. Діапазон можливих варіантів рішення є достатньо великим і конкретна схема вибирається в кожному конкретному випадку з урахуванням багатьох факторів.

Приклад. Розглянемо суперклас *Викладач*, який має атрибути *Табельний номер*, *Прізвище*, *Посада*. Підкласами суперкласу виступають об'єкти *Професор*, *Доцент*, *Асистент* (рис. 6.20). Кожен екземпляр підкласу може бути екземпляром суперкласу, тобто суперклас може мати свої екземпляри (Optional). Кожен викладач обов'язково належить тільки одному підкласу (Or). Ця діаграма перетворюється в реляційну схему відношень показану на рис. 6.21. Зв'язок між відношеннями виконується за допомогою ключа суперкласу (*Табельний номер*).

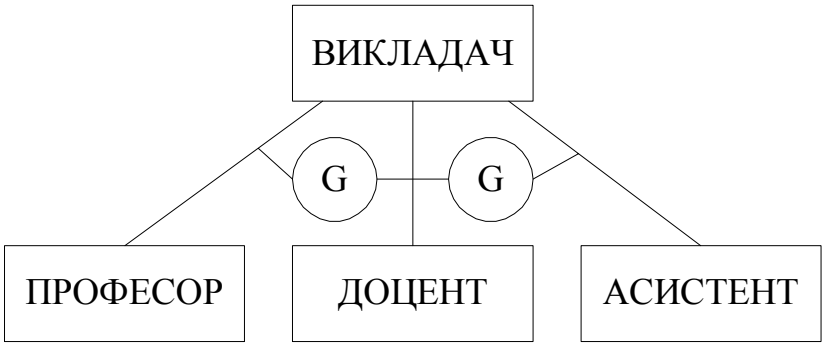


Рис. 6.20. Зв'язок суперклас – підклас з обмеженнями *Optional* і *Or*

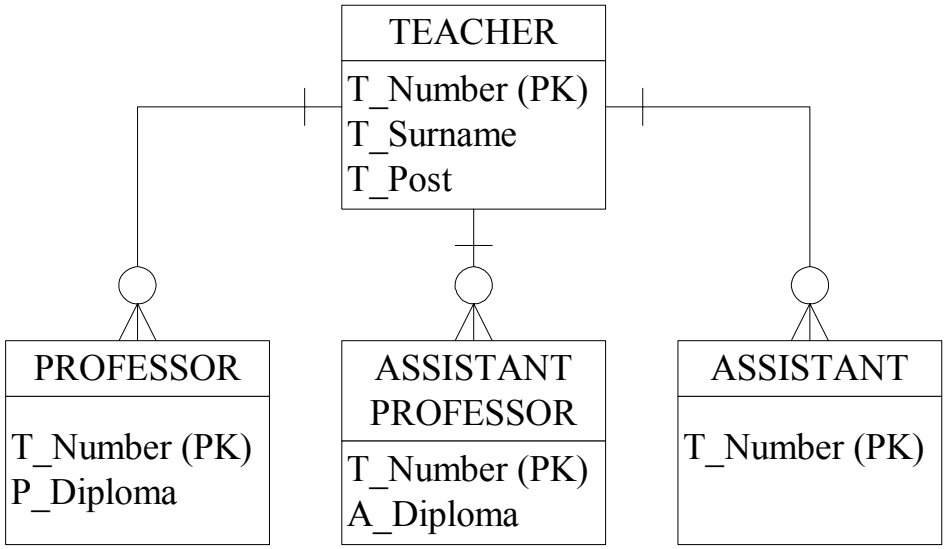


Рис. 6.21. Реляційна схема, яка відповідає попередньому зв'язку суперклас – підклас

Приклад. Розглянемо суперклас *Студент*, який має атрибути *Номер залікової книжки*, *Прізвище*, *Група*. Підкласами суперкласу виступають об'єкти *Очна, Заочна,*

Вечірня і Дистанційна форми навчання (рис. 6.22). Кожен екземпляр підкласу є одночасно екземпляром суперкласу (Mandatory). Кожен студент може належати до декількох підкласів, тобто одночасно може займатися на різних формах навчання (And). Ця діаграма перетворюється в наступну реляційну схему відношень (рис. 6.23).

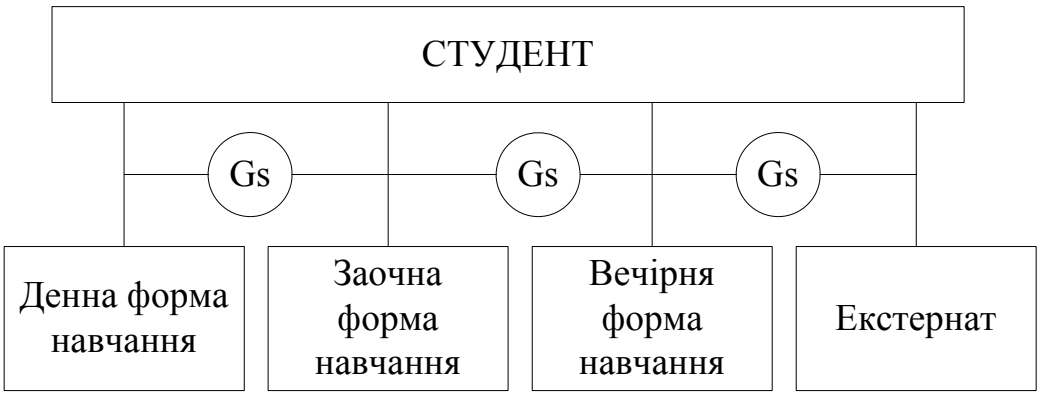


Рис. 6.22. Зв'язок суперклас–підклас з обмеженнями *Mandatory* і *And*

STUDENT
ST_Book (PK)
ST_Surname
ST_Group
ST_Confront
ST_Correspond
ST_Nightclasses
ST_Distance

Рис. 6.23. Реляційна схема, яка відповідає попередньому зв'язку суперклас–підклас

6.4. Перевірка відношень за допомогою правил нормалізації

Створений на попередніх етапах набір відношень логічної моделі БД повинен бути перевірений на коректність об'єднання атрибутів у кожному відношенні. Перевірка виконується шляхом застосування до кожного відношення процедури послідовної нормалізації. Нормалізація гарантує, що отримана модель не буде мати протирічь і буде мати мінімальну збитковість. Атрибути в результаті нормалізації будуть

згруповані відповідно до існуючих між ними логічних зв'язків. Для забезпечення коретності логічної моделі, у разі виявлення відношень, які не відповідають вимогам нормалізації, необхідно повернутися на попередні етапи проектування і перебудувати помилково створені елементи моделі.

Приклад. В результаті проектування отримано відношення показана на рис. 6.24.

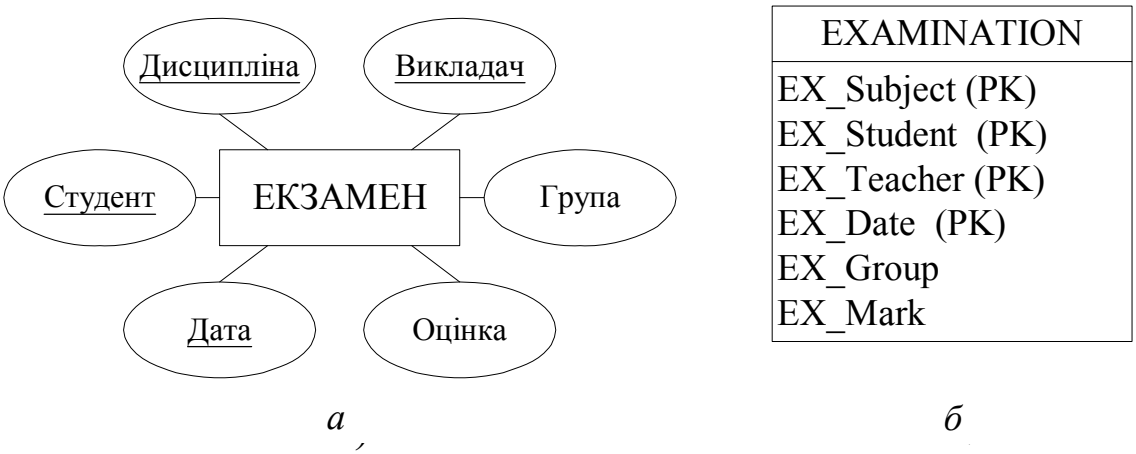


Рис. 6.24. Перетворення сутності *Екзамен* (*а*)
у відношення *Examination* (*б*)

При дослідженні даного відношення були виявлені такі функціональні залежності:

Дисципліна, *Викладач*, *Студент*, *Дата* → *Оцінка*

Студент → *Група*

У наведеній схемі існують аномалії і необхідно продовжити нормалізацію. В результаті декомпозиції вихідного відношення буде отримана схема показана на рис. 6.25.

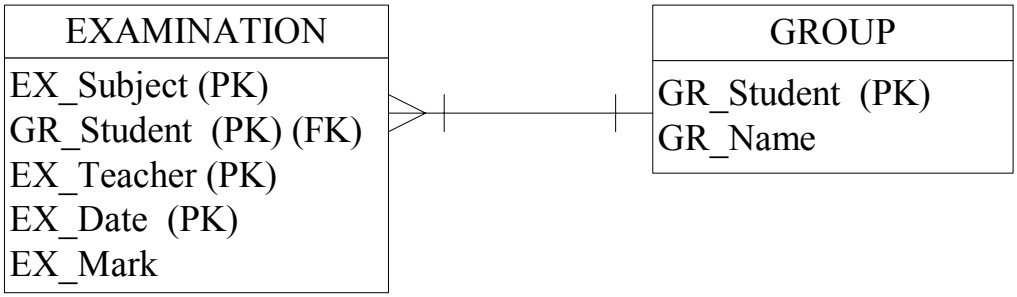


Рис. 6.25. Реляційна схема, яка відповідає сутності *Екзамен*

6.5. Перевірка відповідності відношень вимогам транзакцій користувачів

Перевірка полягає в нанесенні безпосередньо на ER-діаграму всіх шляхів, які потрібні для виконання кожної з транзакцій. Якщо таким чином вдається виконати всі транзакції, то перевірка на цьому завершується. У протилежному випадку необхідно повернутися до попередніх етапів і перевірити, а у разі потреби і змінити ті фрагменти моделі, які не відповідають необхідній роботі транзакцій.

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

6.6. Перевірка підтримки цілісності

Обмеження цілісності запобігають появі в БД суперечливих даних. Вирішення цієї проблеми на стадії проектування полягає у такому:

- наявність обов'язкових і необов'язкових значень даних для атрибутів (NULL, NOT NULL);
- наявність обмежень для доменів атрибутів (визначення області значень або діапазону значень);
- цілісність сутностей (обов'язкова наявність Primary Key в кожному відношенні);
- посилкова цілісність (зв'язування таблиць за допомогою Foreign Key);
- обмеження предметної області (бізнес правила), які реалізуються як засобами БД, так і на рівні застосувань.

У табл. 6.1 наведені правила зовнішнього ключа для відношення "один до багатьох" для сильної сутності.

Таблиця 6.1

Підтримка посилкової цілісності для сильної сутності

Тип зв'язку	Вимоги до зовнішнього ключа
Обов'язкова наявність значень відповідних екземплярів у батьківській і залежній таблицях	NOT NULL ON DELETE RESTRICT ON UPDATE CASCADE
Необов'язкова наявність значень відповідних екземплярів у батьківській і залежній таблицях	NULL ALLOWED ON DELETE SET NULL ON UPDATE CASCADE
Обов'язкова наявність значень відповідних екземплярів у залежній таблиці і *необов'язкова* наявність значень в батьківській таблиці	NULL ALLOWED ON DELETE SET NULL ON DELETE RESTRICT ON UPDATE CASCADE
Обов'язкова наявність значень відповідних екземплярів у батьківській таблиці і *необов'язкова* наявність значень в залежній таблиці	NOT NULL ON DELETE RESTRICT ON UPDATE CASCADE

Для слабкої сутності використовуються ті ж самі правила за винятком обмежень на зовнішній ключ: NOT NULL, ON DELETE CASCADE, ON UPDATE CASCADE.

6.6. Приклад створення логічної моделі бази даних

У прикладі (п.5.4) розроблено концептуальний проект бази даних для предметної області ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД (ВНЗ). ER-діаграма відображає всі бізнес правила, які в свою чергу визначають сутності, атрибути, зв'язки і т.д.

Наступним етапом проектування бази даних є створення логічної моделі бази даних на основі створеної ER-моделі (рис. 5.21). Створення логічної моделі бази даних виконується шляхом застосування правил перетворення (гл. 6.) ER-діаграми в логічну модель (рис. 6.26).

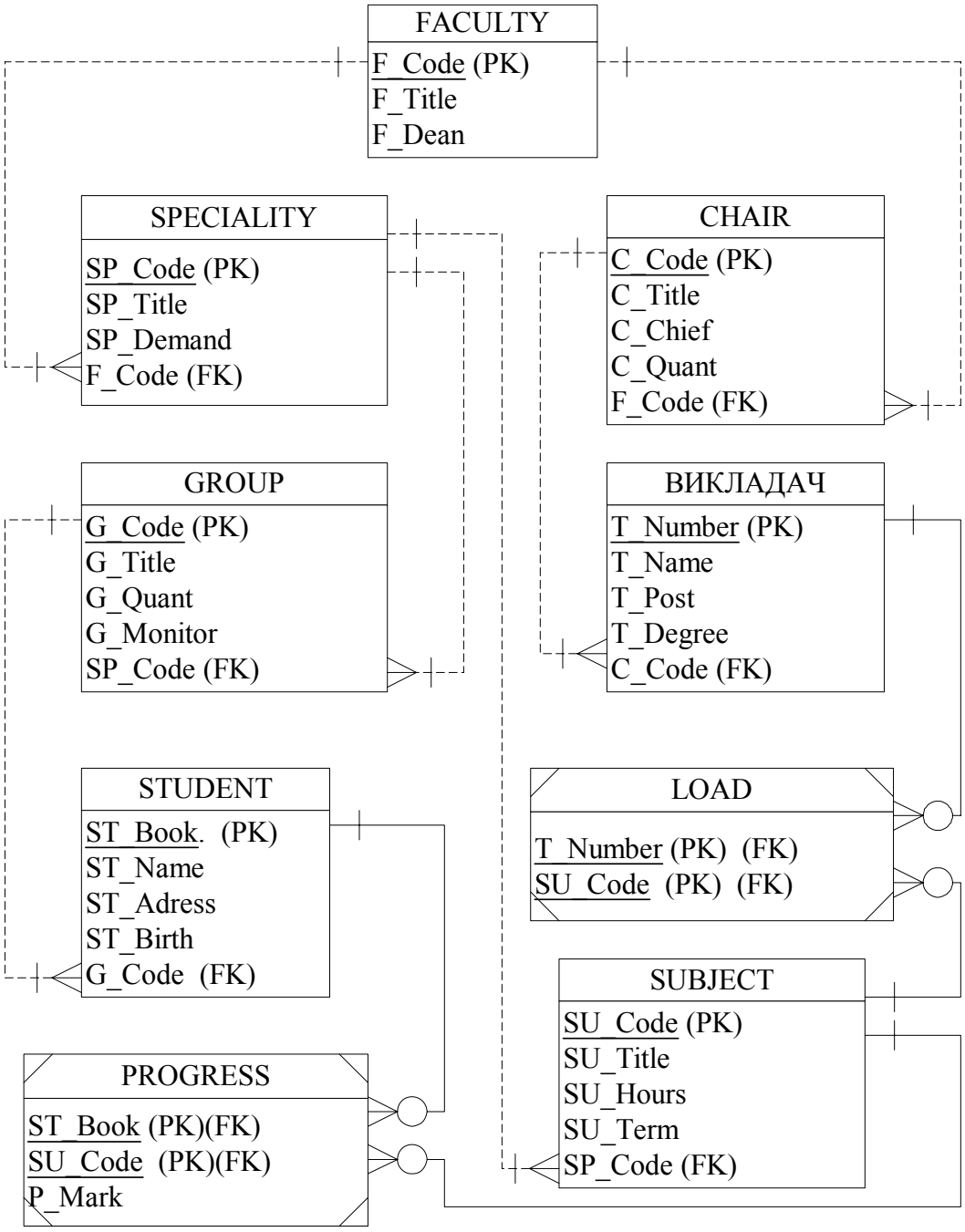


Рис. 6.26. Логічна модель бази даних для предметної області ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД

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 3-й нормальній формі або в нормальній формі Бойса-Кодда. Якщо ця умова не виконується, то необхідно повернутися на попередні етапи проектування і перебудувати помилково створені фрагменти моделі. Перевірка логічної моделі бази даних ВНЗ показує, що реляційна схема знаходиться в 4-й нормальній формі й корегування моделі не потрібно.

Після перевірки логічної моделі за допомогою правил нормалізації система аналізується на предмет виконання транзакцій користувачів, які задаються на початкових етапах проектування. У разі неможливості виконання певних транзакцій необхідне корегування моделі бази даних (див. рис. 6.1).

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

Правила перетворення ER-діаграми в логічну модель наведені в дод. 2.

Контрольні запитання

1. Що називається логічним проектуванням?
2. Яка інформація є вихідною для логічного проектування?
3. Перелічити *етапи логічного проектування*.
4. Які зв'язки ER-діаграм не підтримуються в реляційній схемі?

5. Як відображаються сильні і слабкі сутності та їх атрибути в реляційну схему?
6. Як відображається необов'язковість зв'язку між сутностями в реляційну схему?
7. Як відображається зв'язок "багато до багатьох" в реляційну схему?
8. Як відображаються складні зв'язки і зв'язки з атрибутами в реляційну схему?
9. Як відображаються рекурсивні зв'язки в реляційну схему?
10. Як відображається зв'язок "один до одного" залежно від рівня участі сутностей в реляційну схему?
11. Як відображається зв'язок "один до багатьох" залежно від рівня участі сутностей в реляційну схему?
12. Як відображається зв'язок суперклас–підклас залежно від ступеня участі сутностей і обмеження неперетинання в реляційну схему?
13. Навіщо потрібно виконувати перевірку реляційної схеми на відповідність правилам нормалізації?
14. Як виконується перевірка реляційної схеми на відповідність вимогам транзакцій користувачів?
15. Як перевіряється цілісність реляційної бази даних?
16. Навести приклади створення логічної моделі бази даних.

Глава 7. НОРМАЛІЗАЦІЯ

7.1. Постановка задачі

Нормалізація – це процедура визначення того, які атрибути зв'язані у відношенні. Одна з головних задач при розробці реляційної БД – об'єднання в одному відношенні тих атрибутів, які зв'язані між собою (між якими є функціональні залежності). Нормалізація являє собою поетапний процес заміни сукупності відношень іншою сукупністю (схемою), в

якій відношення мають просту і регулярну структуру. Результатом нормалізаії є логічна модель БД.

Надлишковість даних в БД є небажаним явищем, оскільки призводить до збільшення об'єму пам'яті, уповільнює роботу БД. Надлишковість даних є результатом в першу чергу дублювання даних. Розрізняють *незбиткове* та *збиткове* дублювання даних. Повністю усувати надлишковість не потрібно, оскільки при цьому неможливо буде підтримувати БД як єдине ціле. Слід тільки мінімізувати надлишковість, залишивши необхідне дублювання даних.

Дублювання даних створює проблеми при виконанні операцій з БД. Ці проблеми виникають при спробі зробити операції: редагування, додавання або вилучення даних.

Аномаліями називається така ситуація в БД, яка призводить до протирічь у БД, або суттєво ускладнює обробку даних. Розрізняють аномалії модифікації, додавання і вилучення.

Приклад. Розглянемо відношення *Студент* (табл. 7.1).

Таблиця 7.1

Студент

Номер залікової книжки	Прізвище	Група	Факультет	Декан
1010	Бойко	ІТП-31	АІТ	Барков
2020	Лемешко	ІУСТ-22	АІТ	Барков
1030	Шевченко	ІТП-31	АІТ	Барков
1121	Петренко	БМО-32	АІТ	Барков
2231	Грицюк	ТБ-21	БТ	Тимчук

Аномалія модифікації виникає при спробі змінити прізвище декана. В цій ситуації необхідно переглянути всі кортежі. При великих розмірах БД це потребує значного часу,

при цьому можливі помилки (у разі невірного введення прізвища), які порушать цілісність БД.

Аномалія додавання виникає при додаванні інформації про нового студента, при цьому необхідно вводити інформацію, яка вже є в БД: назва факультету, прізвище декана. Крім того неможливо створити нову групу поки не існує студентів, які в ній займаються.

Аномалія вилучення виникає при спробі вилучити дані про студента, який в групі поки ще один, наприклад Лемешко. В цьому випадку зникне інформація про групу ІУСТ-22.

Виконання декомпозиції наведеного відношення дозволяє позбутися вищеозначених аномалій (табл. 7.2...7.4).

Таблиця 7.2

Студент

Номер залікової книжки	Прізвище	Група
1010	Бойко	ІТП-31
2020	Лемешко	ІУСТ-22
1030	Шевченко	ІТП-31
1121	Петренко	БМО-32
2231	Грицюк	ТБ-21

Таблиця 7.3

Група

Група	Факультет
ІТП-31	АІТ
ІУСТ-22	АІТ
БМО-32	АІТ
ТБ-21	БТ

Таблиця 7.4

Факультет

Факультет	Декан
АІТ	Барков
БТ	Тимчук

Процес проектування БД з використанням декомпозиції являє собою процес послідовної нормалізації схем відношень, при цьому кожна наступна ітерація відповідає нормальній формі більш високого рівня і має кращі властивості у порівнянні з попередньою. Кожній нормальній формі (НФ) відповідає деякий набір обмежень. Визначають такі нормальні форми: 1НФ, 2НФ, 3НФ, НФБК (нормальна форма Бойса-Кодда), 4НФ, 5НФ.

При виконанні декомпозиції зберігається множина вихідних функціональних залежностей між атрибутами і виконується зворотність. З*воротність* означає можливість відновлення вихідної схеми. *Функціональні залежності* відображають зв'язки між атрибутами, які властиві реальному об'єкту.

Атрибут *B функціонально* залежить від *A*, якщо кожному значенню *A* відповідає в точності одне значення *B*. Математичний запис функціональної залежності (ФЗ): $A \rightarrow B$

Приклад. Функціональні залежності:

Студент → *Група; Група* → *Факультет;*

Викладач, *Студент*, *Дисципліна* → *Оцінка*.

Якщо існує ФЗ $A \rightarrow B$, то це означає, що у всіх кортежах з однаковим значенням атрибуту *A* атрибут *B* буде мати також одне й те ж значення. *A* і *B* можуть складатися з декількох атрибутів.

7.2. Нормальні форми

Перша нормальна форма. Відношення знаходиться в 1НФ тоді і тільки тоді, коли всі його атрибути є атомарними.

Значення атрибуту вважається *атомарним*, якщо воно є неподільним у всіх застосуваннях.

Приклад. Представлення даних у таблицях може вважатися як атомарним, так і неатомарним залежно від використання. Засіб представлення визначається необхідним

ступенем деталізації і повинен підтримуватися у всіх застосуваннях (табл. 7.5).

Таблиця 7.5

Дата народження

Прізвище	Дата народження
Бойко	15 лютого 1991

Прізвище	Дата і місяць	Рік
Бойко	15 лютого	1991

Прізвище	День	Місяць	Рік
Бойко	15	лютий	1991

Друга нормальна форма. Відношення знаходиться в 2НФ, якщо воно знаходиться в 1НФ і кожен його непервинний атрибут функціонально повно залежить від первинного ключа.

Неповною функціональною залежністю називається залежність неключового атрибуту від частини ключа, що складається з декількох атрибутів. Повна функціональна залежність передбачає залежність неключового атрибуту від всіх атрибутів одночасно, що входять до складу ключа.

Приклад. Розглянемо відношення *Студент* (табл. 7.6).

Таблиця 7.6

Студент

Номер залікової книжки	Прізвище	Група	Дисципліна	Оцінка
1010	Бойко	ІТП-31	Бази даних	5

Функціональні залежності:

№ залік. кн., *Дисципліна* → *Прізвище*, *Група*, *Оцінка*

№ залік. кн. → *Прізвище*, *Група*

Для приведення даного відношення до 2НФ необхідно розбити його на проекції, при цьому повинна бути виконана

умова відновлення вихідного відношення без втрат. Проекції мають такий вигляд (табл. 7.7...7.8).

Таблиця 7.7

Студент

Номер залікової книжки	Дисципліна	Оцінка
1010	Бази даних	5

Таблиця 7.8

Група

Номер залікової книжки	Прізвище	Група
1010	Бойко	ІТП-31

Відсутність втрат при декомпозиції відношення $R(x,y,z)$ на відношення $R_1(x,y)$ і $R_2(x,z)$ виконується, якщо від спільного атрибуту двох отриманих відношень (x) залежить хоча б один атрибут з двох, що залишилися (y або z), тобто якщо виконується ($x \to y$) або ($x \to z$).

Третя нормальна форма. Відношення знаходиться в 3НФ, якщо воно знаходиться в 2НФ і жоден з непервинних атрибутів у відношенні не є транзитивно залежним від первинного ключа.

Атрибут *C транзитивно* залежить від атрибуту *A*, якщо для атрибутів *A*, *B*, *C* виконуються такі умови $A \to B$ і $B \to$ C, але зворотня залежність відсутня.

Приклад. Розглянемо відношення *Студент* (табл. 7.9).

Таблиця 7.9

Студент

Номер залікової книжки	Прізвище	Група	Факультет
1010	Бойко	ІТП-31	АІТ

Функціональні залежності:

№ залік. кн. → *Прізвище*, *Група*, *Факультет*

Група → *Факультет*

Між атрибутами існує транзитивна залежність. Для того щоби запобігти цьому необхідно виконати декомпозицію відношення (табл. 7.10, 7.11):

Таблиця 7.10

Студент

Номер залікової книжки	Прізвище	Група
1010	Бойко	ІТП-31

Таблиця 7.11

Група

Група	Факультет
ІТП-31	АІТ

Нормальна форма Бойса-Кодда. Відношення знаходиться в НФБК, якщо воно знаходиться в 3НФ і у ньому відсутні залежності атрибутів первинного ключа від неключових атрибутів.

Приклад. Розглянемо відношення *Спеціальність* (табл. 7.12).

Таблиця 7.12

Спеціальність

Спеціальність	Дисципліна	Викладач
ІТП	Бази даних	Барко
ІУСТ	Бази даних	Шевченко

Припустимо, що на кожній спеціальності певну дисципліну може викладати тільки один викладач і кожен викладач викладає тільки одну дисципліну. У цьому випадку мають місце такі залежності:

Спеціальність, *Дисципліна* → *Викладач*

Викладач → *Дисципліна*

Відношення знаходиться в 3НФ, але неключовий атрибут *Викладач* визначає атрибут *Дисципліна*, що входить у ключ.

Для того щоби позбутися аномалій необхідно виконати декомпозицію відношення (табл. 7.13, 7.14).

Таблиця 7.13

Спеціальність

Спеціальність	Дисципліна
ІТП	Бази даних
ІУСТ	Бази даних

Таблиця 7.14

Дисципліна

Викладач	Дисципліна
Барко	Бази даних
Шевченко	Бази даних

Четверта нормальна форма. Відношення знаходиться в 4НФ тоді і тільки тоді, коли у випадку існування багатозначної залежності $A \rightarrow\rightarrow B$ всі інші атрибути відношення функціонально залежать від A.

У відношенні $R(A,B,C)$ існує *багатозначна залежність* $A \rightarrow\rightarrow B$ в тому і тільки в тому випадку, коли множина значень B, що відповідає парі значень A і C залежить тільки від A і не залежить від C.

Відношення $R(A,B,C)$ можна розбити без втрат інформації на відношення $R_1(A,B)$ і $R_2(A,C)$ в тому і тільки в тому випадку, якщо існують *багатозначні залежності* $A \rightarrow\rightarrow B$ і $A \rightarrow\rightarrow C$.

Приклад. Розглянемо відношення *Кафедра* (табл.7.15).

Таблиця 7.15

Кафедра

Кафедра	Викладач	Група
ІТ	Барко	ІТП-31
ІТ	Барко	ІТП-32
ІТ	Шевченко	ІТП-31
ІТ	Шевченко	ІТП-32

У даному відношенні існують дві багатозначні залежності:

Кафедра →→ *Викладач*

Кафедра →→ *Група*

Це означає, що кожній кафедрі відповідає перелік викладачів, які на ній працюють і кожній кафедрі відповідає перелік груп, яким ця кафедра викладає дисципліни.

Для того, щоби звести відношення до 4НФ, необхідно виконати його декомпозицію (табл. 7.16, 7.17).

Таблиця 7.16

Кафедра

Кафедра	Викладач
ІТ	Барко
ІТ	Шевченко

Таблиця 7.17

Група

Кафедра	Група
ІТ	ІТП-31
ІТ	ІТП-32

П'ята нормальна форма. Відношення знаходиться в 5НФ тоді і тільки тоді, коли будь-яка залежність з'єднання у відношенні виходить з існування деякого можливого ключа у відношенні.

Відношення *R*(*X*,*Y*,…,*Z*) задовольняє *залежності з'єднання* (*X*,*Y*,…,*Z*) тоді і тільки тоді, коли *R* відновлюється без втрат інформації шляхом з'єднання своїх проекцій на *X*, *Y*,…, *Z*. Залежність з'єднання є узагальненням функціональної і багатозначної залежностей.

Приклад. Розглянемо відношення *Заняття*:

Заняття (*Студент*, *Викладач*, *Дисципліна*)

Кожен студент слухає лекції багатьох викладачів, кожен викладач викладає для багатьох студентів, кожен студент вивчає багато дисциплін, кожен викладач викладає багато

дисциплін. У відношенні відсутні багатозначні і функціональні залежності й воно знаходиться в 4НФ. У відношенні можливі аномалії, які пов'язані з повтором значень атрибутів в декількох кортежах. Наприклад, якщо студент навчається у багатьох викладачів, то при його відрахуванні з університету необхідно знайти і вилучити декілька записів з відношення.

Утворимо такі складені атрибути відношення:

СВ (*Студент*, *Викладач*)

СД (*Студент*, *Дисципліна*)

ВД (*Викладач*, *Дисципліна*).

Якщо відношення *R* спроектувати на складені атрибути СВ, СД, ВД, то з'єднання цих проекцій дасть вихідне відношення. Це означає, що у відношенні *Заняття* існувала залежність з'єднання. Результатом декомпозиції відношення *Заняття* буде отримання таких відношень: R_1(*Студент*, *Викладач*), R_2(*Студент*, *Дисципліна*), R_3(*Викладач*, *Дисципліна*).

Для зведення вихідного відношення до 5НФ виконують його декомпозицію на відношення, кількість яких перевищує два.

Результати зведення до нормальних форм наведені в табл. 7.18.

Таблиця 7.18

Правила формування нормальних форм

Нормальні форми	Приклад
1НФ	$R(ABCD)$ – відношення A,B,C,D – атомарні атрибути
2НФ	$R(\underline{AB}CD)$ – відношення, $\underline{AB}$ – ключ, $\underline{AB} \to CD$, неможливі залежності: $\underline{A} \to CD$, $\underline{A} \to C$, $\underline{A} \to D$, $\underline{B} \to CD$, $\underline{B} \to C$, $\underline{B} \to D$
3НФ	$R(\underline{AB}CD)$ – відношення, $\underline{AB}$ – ключ, $\underline{AB} \to CD$, неможливі залежності: $C \to D$, $D \to C$

Закінчення табл. 7.18

Нормальні форми	Приклад
НФБК	$R(\underline{AB}CD)$ – відношення, $\underline{AB}$ – ключ, $\underline{AB} \to CD$, неможливі залежності: $C \to \underline{A}$, $C \to \underline{B}$, $D \to \underline{A}$, $D \to \underline{B}$, $C \to \underline{AB}$, $D \to \underline{AB}$
4НФ	$R(\underline{A}BC)$ – відношення, $A \to\to B$, $A \to C$, неможливі залежності: $A \to\to C$
5НФ	$R(ABC)$ – вихідне відношення; результат декомпозиції: $R_1(\underline{AB})$, $R_2(\underline{AC})$, $R_3(\underline{BC})$

7.3. Денормалізація

Денормалізація – модифікація реляційної моделі, при якій ступінь нормалізації модифікованого відношення стає нижче, ніж ступінь нормалізації щонайменше одного з вихідних відношень.

Денормалізація застосовується у тих випадках, коли нормалізована БД не задовольняє вимогам, що висуваються до продуктивності системи. Денормалізація може застосовуватися у таких випадках:

- об'єднання таблиць зі зв'язками "один до одного";
- дублювання неключових атрибутів у зв'язках "один до багатьох" для зменшення кількості з'єднань;
- дублювання атрибутів зовнішнього ключа у зв'язках "один до багатьох" для зменшення кількості з'єднань;
- дублювання атрибутів "багато до багатьох" для зменшення кількості з'єднань;
- створення таблиць з даних, що містяться в інших таблицях;
- введення груп полів, що повторюються.

Застосовуючи денормалізацію слід враховувати, що цей процес має такі негативні наслідки:

- призводить до появи аномалій БД;
- знижує гнучкість системи;
- може зменшити час на відповіді до БД, але при цьому уповільнює операції оновлення даних;
- може ускладнити фізичну реалізацію системи.

Контрольні запитання

1. Дати визначення терміну *функціональна залежність*. Навести приклади функціональних залежностей.
2. У чому полягає збиткове і незбиткове дублювання даних?
3. Що таке *аномалії* додавання, оновлення, вилучення?
4. Дати визначення 2НФ. Навести приклад відношення, яке знаходиться в 1НФ, але не знаходиться у 2НФ. Звести його до 2НФ.
5. Дати визначення 3НФ. Навести приклад відношення, яке знаходиться в 2НФ, але не знаходиться у 3НФ. Звести його до 3НФ.
6. Дати визначення НФБК. Навести приклад відношення, яке знаходиться в 3НФ, але не знаходиться у НФБК. Звести його до НФБК.
7. Дати визначення терміну *багатозначна залежність*. Навести приклади багатозначних залежностей.
8. Дати визначення 4НФ. Навести приклад відношення, яке знаходиться в НФБК, але не знаходиться у 4НФ. Звести його до 4НФ.
9. Дати визначення 5НФ. Навести приклад 5НФ.
10. Яке місце займає нормалізація в процесі проектування бази даних?
11. Що таке денормалізація бази даних і які її переваги і недоліки?
12. Навести приклади проекту бази даних, коли доцільно виконати денормалізацію.

Глава 8. ФІЗИЧНА ОРГАНІЗАЦІЯ БАЗ ДАНИХ

8.1. Організація зберігання інформації

Фізична організація даних відповідає за їх зберігання, управління, форми представлення і структури даних.

Фізичне проектування являє собою процес визначення характеристик сховища даних і доступу до них в БД. Властивості сховища даних залежать від пристроїв зберігання, засобів доступу до даних, що підтримуються системою і від СУБД. На етапі фізичного проектування визначається місцезнаходження даних на пристроях зберігання, загальна продуктивність системи.

В реляційних БД складні фізичні процеси організації даних приховані від користувача, але вони мають великий вплив на продуктивність роботи з БД.

Процес пошуку і представлення даних користувачу виконується в декілька етапів (рис. 8.1).

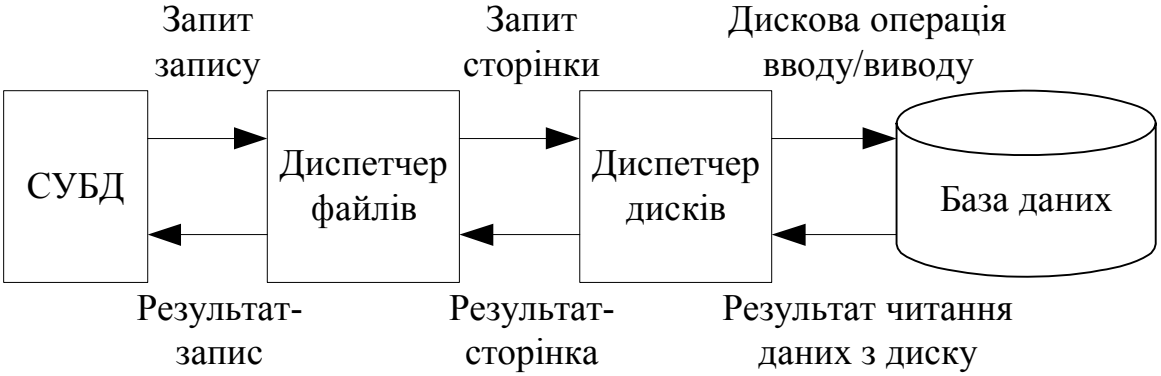


Рис. 8.1. Організація доступу до даних

Спочатку визначається необхідний запис, для знаходження якого викликається диспетчер файлів. Диспетчер файлів визначає сторінку, на якій знаходиться запис, і для її отримання викликає диспетчер дисків. Диспетчер дисків визначає фізичне положення необхідної сторінки на диску і передає її диспетчеру файлів, а той – СУБД. Головними

одиницями операцій обміну системи СУБД – БД є сторінки даних. З точки зору СУБД БД виглядає як набір записів, з точки зору диспетчера файлів БД виглядає як набір сторінок. Кожна сторінка пам'яті має унікальний ідентифікатор. Кожен запис зберігається повністю на одній сторінці.

Для організації зберігання даних застосовується технологія *кластерізації* – фізично близьке розташування записів у просторі пам'яті середовища зберігання БД.

Пам'ять сторінок – організація простору зовнішньої або віртуальної пам'яті в БД, яка передбачає поділ простору пам'яті на сторінки фіксованого розміру. Для розміщення сторінок, що викликаються, в оперативній пам'яті використовуються спеціальні області – *буфери*. Оновлений в буферах зміст сторінок повертається у зовнішню пам'ять. Розмір сторінок, кількість буферів, алгоритми вибору буферів для розміщення сторінок суттєво впливають на ефективність роботи системи.

Компоненти системи БД, які призначені для зберігання даних, мають різну ємкість і різну швидкість доступу до даних. Організація ієрархії пам'яті комп'ютера наведена на рис. 8.2.

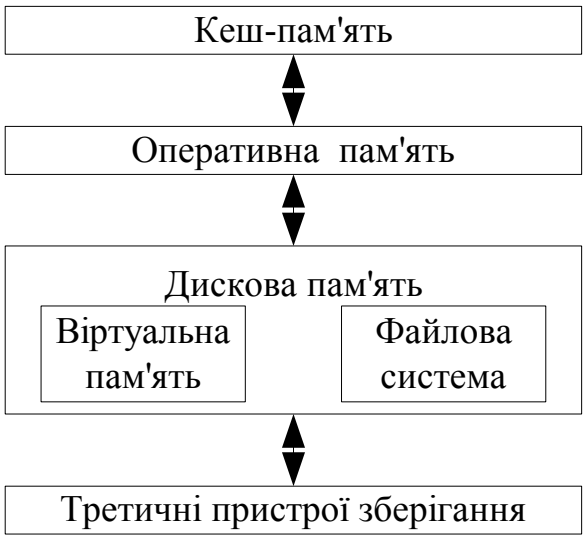


Рис. 8.2. Ієрархія пристроїв пам'яті бази даних

Кеш-пам'ять – швидка буферна пам'ять відносно невеликого розміру, яка зберігає останні дані до яких

виконувався доступ. Вона реалізується апаратними або програмно-апаратними засобами.

Віртуальна пам'ять – це пам'ять, яка моделюється за допомогою апаратних засобів і програмного забезпечення. Вона дозволяє користувачу оперувати з об'ємами даних, які значно перевершують простір пам'яті, що виділяється в оперативній пам'яті, таким чином нібито вони знаходились в оперативній пам'яті.

Пам'ять третична – запам'ятовуючий пристрій великого об'єму і з низькою вартістю. Найчастіше це стойки з компакт-дисками або касети магнітної стрічки.

Організація зберігання даних має таку ієрархію:

- атрибути відображаються у байтову послідовність постійної або змінної довжини, яка називається *полями*;
- поля об'єднуються в набори даних постійної або змінної довжини, які називаються *записом*;
- записи зберігаються у фізичних *блоках* (*сторінках*);
- набори записів, які відповідають відношенням, зберігаються у вигляді *файлів*.

Розрізняють такі засоби фізичної організації файлів даних: послідовний, індексно-послідовний, прямий.

Послідовний файл – файл, до записів якого можливий послідовний доступ у порядку їх фізичного розміщення в пам'яті. Послідовна організація ефективна, коли застосування при кожному зверненні до БД обробляє значну кількість записів.

Індексно-послідовний файл – файл, до записів якого можливий прямий доступ за допомогою створеного для нього *індексу* по заданому ключу, а також послідовний доступ згідно з їх впорядкуванням по цьому ключу індексування. Індексно-послідовна організація ефективна, коли достатньо багато застосувань потребують послідовної обробки і достатньо багато застосувань потребують прямого доступу.

Файл прямого доступу – файл, в якому забезпечується прямий доступ до його записів за їх безпосередньою або посередньою адресою у середовищі зберігання або за заданими ключем за допомогою будь-якого методу відображення ключа в адресу. Пряма організація необхідна, коли для більшості застосувань потрібен прямий доступ до записів.

8.2. Індексація

Індекс – структура даних, яка допомагає СУБД швидше знайти окремі записи в файлі і скоротити час виконання запитів користувачів. Основне призначення індексів полягає в забезпеченні ефективного прямого доступу до записів таблиці по ключу. Файл, що містить індексні записи називається *індексним файлом*.

Залежно від організації індексної і основної області розрізняють два типи файлів: зі щільним індексом і з розрідженим індексом. *Щільний індекс* вміщує індексні записи для всіх значень ключа пошуку в даному файлі. *Розріджений індекс* вміщує індексні записи тільки для деяких значень ключа пошуку в даному файлі.

У файлах зі щільним індексом основна область вміщує послідовність записів однакової довжини, *які розташовані у довільному порядку*, а структура індексного запису має такий вигляд: (значення ключа, номер запису).

В індексних файлах зі щільним індексом для кожного запису в основній області існує один запис з індексної області. *Всі записи* в індексній області *впорядковані* за значенням ключа. Файли зі щільним індексом називаються *індексно-прямими файлами*.

У файлах з розрідженим індексом основна область вміщує послідовність записів однакової довжини, *які розташовані у впорядкованому порядку*, а структура індексного запису має

такий вигляд: (значення ключа першого запису блока, номер блока з цим записом).

Розріджений індекс будується для впорядкованих файлів. Файли з розрідженим індексом називаються *індексно-послідовними файлами*.

Індекси доцільно створювати по стовпцю або групі стовпців у таких випадках:

- часто виконується пошук у БД за атрибутами, які перелічені в умові WHERE оператора SELECT;
- часто виконується об'єднання таблиць за певними атрибутами;
- часто виконується сортування таблиць (використання ORDER BY в операторі SELECT).

Індекси не доцільно створювати по стовпцю або групі стовпців у таких випадках:

- рідко використовуються для пошуку;
- містять значення, які часто змінюються, що призводить до частого оновлення індексу і, відповідно, уповільнює роботу;
- містять незначну кількість значень.

Як правило в БД визначають два види індексів:

- індекси, які використовуються запитами для доступу до даних;
- індекси, які забезпечують посилкову цілісність між таблицями.

Більша частина БД підтримує B-дерева, крім того деякі з них працюють із хеш-таблицями. Деякі системи представляють багатомірні структури даних, такі, як варіанти R-дерев і квадрадерев (*kd*-дерева). Окремі системи підтримують також бітові вектори і багатотабличні індекси з'єднання. Системи, що розташовуються в оперативній пам'яті, крім того підтримують структури даних, які мають невеликі коефіцієнти розгалуження, такі як T-дерева, бінарні дерева.

8.3. Хешування

Хешування – технологія прямого доступу до записів БД за відомими значеннями їхніх ключів. Час доступу суттєво не залежить від кількості записів, що зберігаються. Суть методу хешування полягає в тому, що за значеннями ключа k, або його певних характеристик, обчислюється деяка хеш-фукція $h(k)$ і отримане значення береться в якості адреси початку пошуку. Недоліком більшості хеш-функцій є те, що вони не гарантують отримання унікальної адреси, оскільки кількість можливих значень поля хешування може бути значно більшою за кількість адрес, які доступні для запису. Ситуація, коли різним значенням ключів відповідає одне значення хеш-функції називається *колізією*. Значення ключів, які мають одну й ту ж хеш-функцію називаються *синонімами*. Для вирішення колізій застосовують спеціальні методи: послідовного перебору, введення області переповнення, багатократне хешування. Хеш-структура показана на рис.8.3.

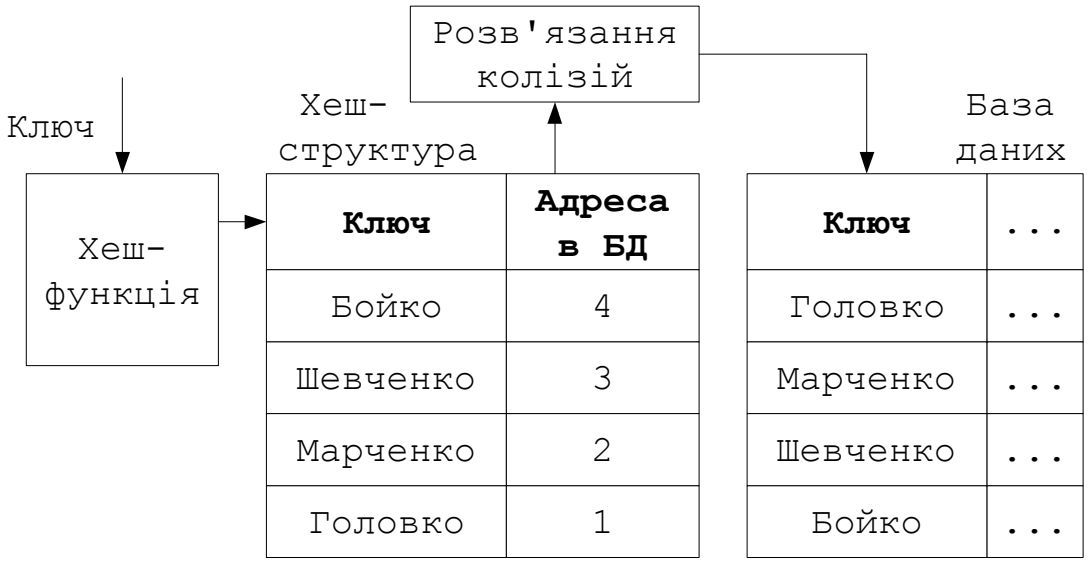


Рис. 8.3. Організація хеш-структури

Хеш-структури можуть забезпечити відповідь на запитання до БД за одне звернення до диску при умові відсутності синонімів. Хеш-структури мають низьку

ефективність при запитах на визначення діапазонів, знаходження екстремумів і деяких інших.

8.4. B-дерева

У багатьох СУБД для збереження індексів використовується структура даних, яка називається деревом. Серед дерев найбільш поширені бінарні дерева, B-дерева та їх різновиди (B^+-дерева, B^*-дерева і т.ін.). B-дерево є збалансованим, впорядкованим за значенням ключа деревом. Найбільш розповсюдженими серед B-дерев є B^+-дерева.

Приклад. На рис. 8.4 наведено фрагмент B^+-дерева і його зв'язок з БД.

8.5. Інвертовані файли

Для забезпечення прискореного доступу за вторинними ключами використовуються структури, які називаються *інвертованими списками*. Інвертований список являє собою дворівневу індексну структуру. На першому рівні знаходиться файл або частина файлу, в який впорядковано розташовані значення вторинних ключів. Кожен запис із вторинним ключем має покажчик на номер першого блоку в ланцюгу блоків, які містять номери записів з даним значенням вторинного ключа. На другому рівні знаходиться ланцюг блоків, які містять номери записів, що вміщують одне і те ж саме значення вторинного ключа. При цьому блоки другого рівня впорядковані за значенням вторинного ключа. На третьому рівні знаходиться безпосередньо файл даних. Пошук даних виконується у такій послідовності:

- на першому рівні знаходиться значення вторинного ключа;
- за знайденим покажчиком читається блок другого рівня, який містить номери всіх записів із заданим значенням вторинного ключа;

– у робочу область завантажується зміст всіх записів із заданим значенням вторинного ключа.

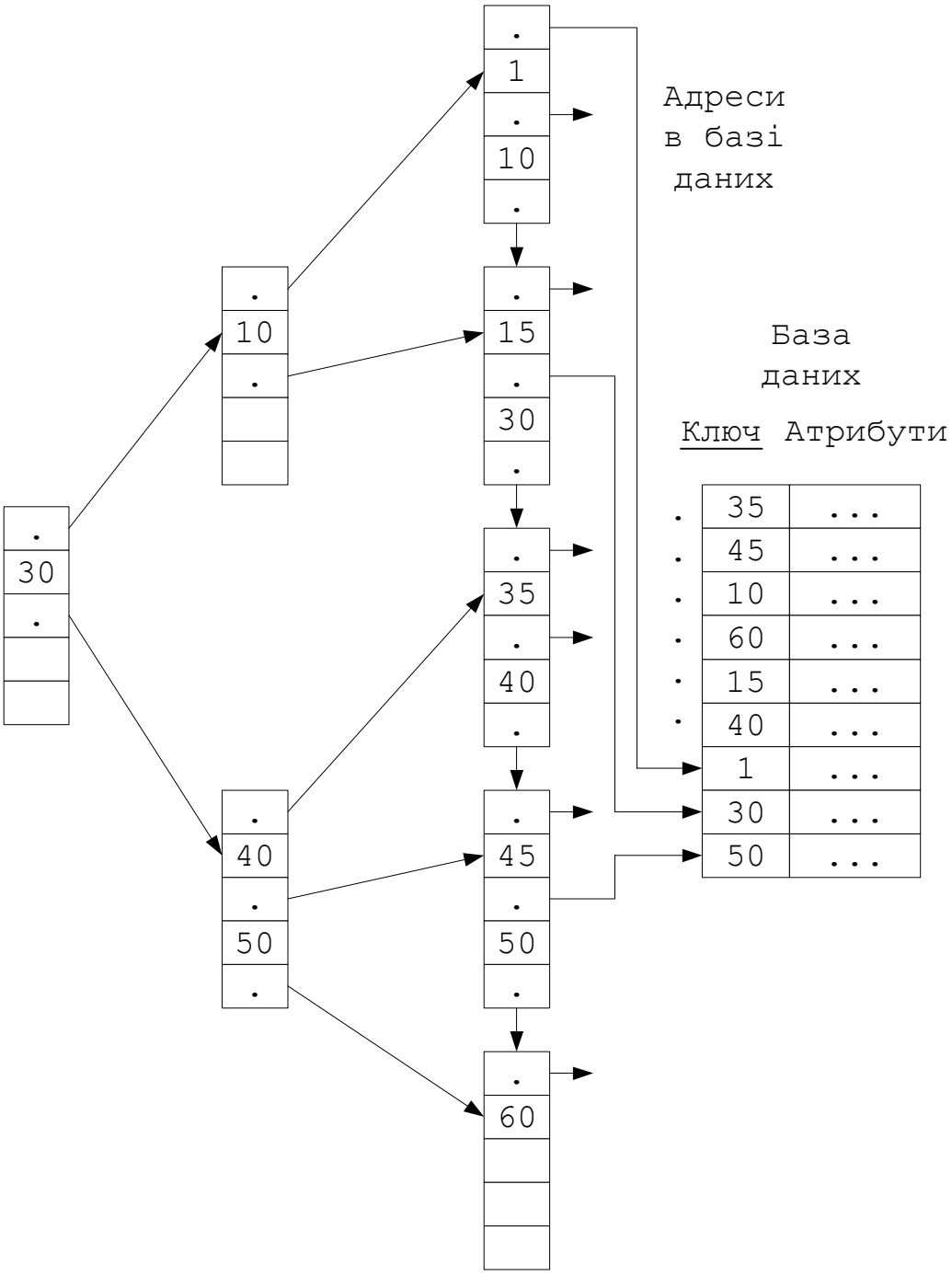


Рис. 8.4. Приклад побудови B^+-дерева

Контрольні запитання

1. Порівняти послідовну, індексно-послідовну і пряму організацію файлів.

2. Чому розподіл записів по блоках впливає на швидкість роботи системи?
3. Пояснити різницю між організацією пошукових структур за первинним ключем і за вторинним ключем.
4. Дати характеристику сторінкової організації даних в СУБД.
5. Що собою являє технологія хешування і для чого вона використовується в базах даних?
6. У чому полягає основне призначення індексів? Які переваги і недоліки застосування індексів?
7. Які індекси називаються щільними і які розрідженими? Яка різниця в їх застосуваннях?
8. Що собою являє ієрархія пам'яті бази даних? Які види пам'яті існують?

Глава 9. ЗАСОБИ АВТОМАТИЗАЦІЇ ПРОЕКТУВАННЯ БАЗ ДАНИХ

9.1. CASE-технології

Проблеми автоматизації проектування інформаційних систем викликали необхідність в програмно-технологічних засобах спеціального класу – CASE-засобах (Computer Aided Software Engineering). CASE-технологія являє собою сукупність методів проектування програмного забезпечення, а також набір інструментальних засобів, які дозволяють у наочній формі моделювати предметну область, аналізувати цю модель на всіх стадіях розробки і супроводження програмного забезпечення і розробляти застосування згідно з інформаційними потребами користувачів.

Методологія CASE-технологій ґрунтується на спадному підході до проектування і дає змогу стежити за всіма етапами життєвого циклу інформаційної системи або її окремих задач. Сутність спадного підходу до проектування полягає в тому, що

під час реалізації системи її характеристики конкретизуються все більше і більше.

Переваги від застосування CASE-технологій при проектуванні інформаційних систем полягають у такому:

- прискорюється та полегшується процес розробки, підвищується якість розроблюваних інформаційних систем;
- з'являється можливість переносу застосувань із середовища однієї СУБД в іншу за рахунок перетворення концептуальної моделі на фізичну і навпаки;
- з'являється можливість проведення більш досконалого моделювання системи на початкових етапах розробки.

У табл. 9.1 наведено, порівняння якісних змін процесу розробки інформаційної системи при переході до використання CASE-засобів.

Таблиця 9.1

Порівняння трудомісткості традиційної розробки інформаційної системи і розробки із застосуванням CASE-засобів

Традиційна розробка	**Розробка з використанням CASE-засобів**
Основні зусилля на кодування і тестування	Основні зусилля на аналіз і проектування
"Паперові" специфікації	Швидке ітераційне прототипування
Ручне кодування	Автоматична генерація коду
Ручне документування	Автоматична генерація документації
Тестування кодів	Автоматичний контроль проекту
Супроводження кодів	Супроводження специфікацій проектування

Одним з напрямів застосування CASE-засобів є інформаційне проектування, яке реалізує методики проектування БД. Крім того CASE-засоби забезпечують проектування застосувань.

Інструментальні CASE-засоби складаються з таких компонентів:

- графічні засоби, які призначені для розробки структурних діаграм;
- блоки дизайну і створення звітів для розробки інформаційних форматів вводу/виводу й інтерфейсів користувача;
- інтегрований архів (репозиторій) для збереження даних проекту системи зі словником даних;
- генератор програмної документації;
- засоби для забезпечення повністю автоматизованої перевірки несуперечливості системи, її синтаксису і повноти.

В базі даних CASE-системи зберігаються дані, які відносяться до різних етапів життєвого циклу розробки програмного забезпечення: планування, збору і аналізу вимог, проектування, реалізації, тестування, супроводження і документування.

В середовищі розробки CASE проектувальники БД і застосувань використовують CASE-інструментарій для зберігання опису схеми БД, елементів даних, прикладних процесів, екранів, звітів та інших даних, які зв'язані з процесом розробки. CASE-інструментарій інтегрує всю інформацію розробки системи в єдиному репозиторії, який адміністратор БД може перевіряти на несуперечливість і точність, і у разі потреби, виконувати виправлення.

Репозиторій – застосування БД, яке забезпечує зберігання і обробку даних і *метаданих*, а також їх представлення по запитах. Репозиторії можуть підтримувати різні представлення даних і метаданих, що зберігаються,

множину їх версій, відображати еволюцію структури даних і метаданих.

Метадані – дані про дані, які описують їх склад і структуру, формат представлення, методи доступу і необхідні для цього повноваження користувачів, місце зберігання, семантику даних і т.ін.

Метадані виконують такі функції:

- описують властивості інформаційної системи, її механізми й інформаційні ресурси в CASE-середовищах;
- використовуються для обміну відомостями між різними інструментами CASE і/або застосуваннями інформаційної системи;
- є джерелами відомостей про властивості і зміст інформаційних ресурсів для механізмів управління даними в інформаційних системах;
- забезпечують механізми інтеграції інформаційних ресурсів з різних джерел відомостями про властивості цих ресурсів;
- є джерелом інформації, яка необхідна для перебудови інформаційних систем;
- забезпечують представлення відомостей про систему, її ресурси для різних застосувань і користувачів.

Крім того CASE-інструментарій розширює і покращує якість взаємодії між адміністратором БД, прикладними програмістами і користувачами. Адміністратор БД за допомогою CASE-інструментарію може перевіряти схеми даних для застосувань, стежити за виконанням умов про найменування, дублювання елементів даних, перевіряти використання правил для елементів даних. CASE-засоби дозволяють каскадно передавати виправлення по всій інфраструктурі застосування, що значно спрощує роботу з впровадження системи БД.

Серед інструментальних CASE-систем розрізняють інтегровані комплекси інструментальних засобів для автоматизації всіх етапів життєвого циклу інформаційної системи (Workbench) і спеціалізовані інструментальні засоби для виконання окремих функцій (Tools).

Сучасні CASE-системи є або структурними, або об'єктно-орієнтованими.

У ***структурному підході*** до аналізу та проектування застосовуються такі види моделей:

- DFD (Data Flow Diagrams) − діаграми потоків даних;
- SADT (Structured Analysis and Design Technique − метод структурного аналізу і проектування) − моделі і відповідні функціональні діаграми;
- ERD (Entity-Relationships Diagrams) − діаграми "суть–зв'язок".

Діаграми потоків даних є основним засобом моделювання функціональних вимог до системи, що проектується. Ці вимоги представляються у вигляді ієрархії функціональних компонентів, які зв'язані потоками даних (рис.9.1).

Специфікація процесів представляється у вигляді текстового опису, схем алгоритмів, псевдокодів і т.ін. *Словник термінів* являє собою короткий опис основних понять, які використовуються при створенні специфікації. *Діаграма переходів станів* демонструє поведінку системи, що розробляється. *Моделювання даних* виконується за допомогою ER-діаграм. Головна мета такого представлення – продемонструвати, як кожен процес перетворює свої вхідні дані у вихідні, а також виявити зв'язки між цими процесами.

Функціональні моделі SADT призначені для опису функціональної структури системи, що проектується.

Крім DFD для функціонального структурного і потокового моделювання застосовуються методики IDEF0 і IDEF3.

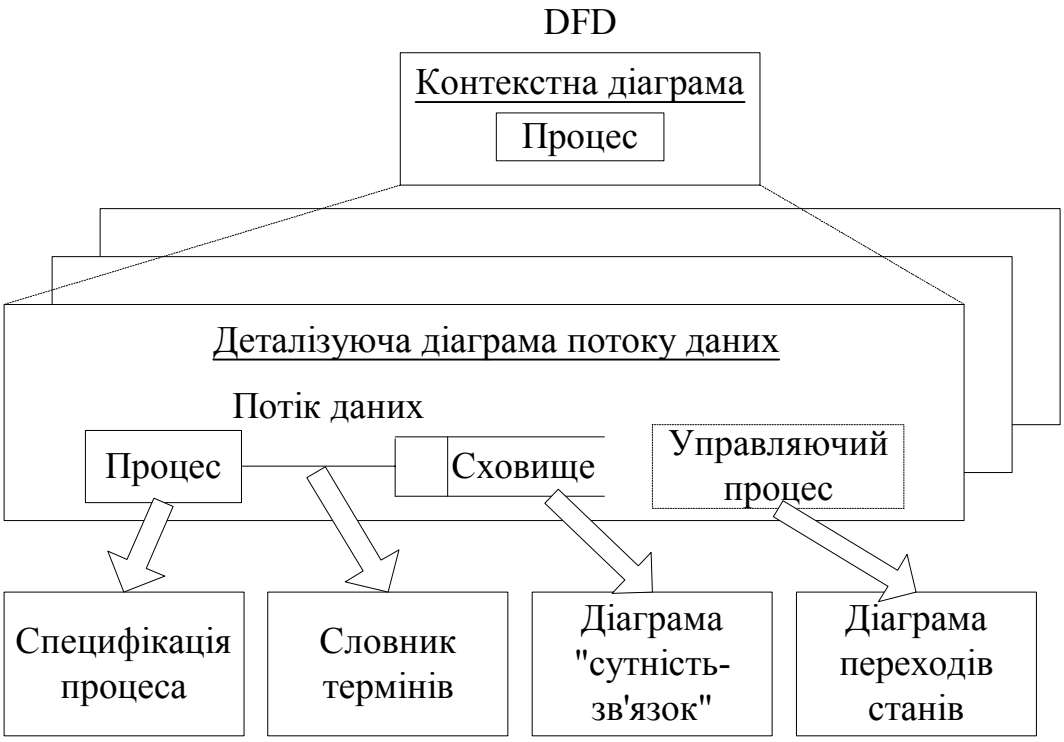


Рис. 9.1. Елементи специфікації методологій структурного аналізу і проектування інформаційних систем, які засновані на потоках даних

Серед ***об'єктно-орієнтованих*** моделей найбільш відомими є моделі побудовані за допомогою мови моделювання UML (Unified Modeling Language – уніфікована мова моделювання). Словник UML утворюють предмети, відношення, діаграми. *Предмети* розрізняють структурні, поведінки, групуючі і пояснюючі. *Відношення* існують таких видів: залежності, асоціації, узагальнення і реалізації. Проект інформаційної системи, який створюється за допомогою UML складається з *діаграм*: прецендентів використання, класів, станів, активності, слідування, співробітництва, компонентів, розміщення.

Основними компонентами об'єктно-орієнтованої CASE-системи є такі:

- репозиторій, який представляє об'єктно-орієнтовану БД;
- графічний інтерфейс користувача;

– засоби перегляду проекту, які дозволяють переміщуватися по елементах проекту, в тому числі по ієрархії класів і підсистем, переключення між видами діаграм;
– засоби контролю проекту;
– засоби збору статистики;
– генератор документів, який дозволяє формувати тексти вихідних документів на основі інформації з репозиторія.

Об'єктний підхід одночасно є і структурним, оскільки задовольняє основним його критеріям (розбиття на чорні скринькі, ієрархія, графічна нотація). Він також використовує діаграми потоків даних, діаграми "сутність-зв'язок", діаграми переходів станів.

Засоби моделювання даних інтегруються з іншими моделями проектування (діаграми потоків даних, об'єктно-орієнтоване моделювання і т.ін.) на основі розширених можливостей репозиторіїв, а також принципів і середовищ інтеграції CASE.

9.2. RAD-технології та компонентно-орієнтовані технології

Для прискорення розробки застосувань використовуються *RAD-технології*. Головними рисами RAD (Rapid Application Development, середовища швидкої розробки застосувань) є такі:

– наявність об'єктно-орієнтованої мови програмування;
– візуальні засоби розробки;
– підтримка стандартних протоколів обміну даними між застосуваннями, що дозволяє розробляти багаторівневі застосування, які не залежать від джерела даних.

Ця технологія орієнтована на максимально швидке отримання перших версій програмного продукту що розробляється. У разі такого підходу систему поділяють на підсистеми, які є слабко зв'язаними за даними і функціями і точно визначають інтерфейси між різними частинами.

Процес розробки поділяється на такі етапи: аналіз і планування вимог користувачів, проектування, реалізація і впровадження.

На етапі аналізу і проектування формулюються найбільш пріоритетні вимоги, що обмежує розмір проекту. На етапі проектування застосовуються CASE-засоби. На етапі реалізації виконується ітеративна побудова реальної системи. Для контролю над виконанням вимог до системи залучаються користувачі.

Компонентно-орієнтовані технології засновані на використанні попередньо розроблених готових програмних компонентів. Тут широко застосовуються бібліотеки класів. Включення готового модуля в систему виконується за допомогою його інтерфейсу. Специфікації, які визначають інтерфейс, відокремлені від модуля, а внутрішні деталі приховані від користувача. Компоненти постачаються у скомпільованому вигляді. Звернення до модуля можливо тільки через його інтерфейс.

Користувач звертається із запитом на виконання деякої процедури. Запит відправляється посереднику. У посередника є попередньо сформований *каталог* (реєстр або репозиторій) інтерфейсів процедур з покажчиком на компоненти-виконавці. Після виконання процедури отримані результати повертаються користувачу.

До найбільш відомих компонентно-орієнтованих технологій належать: CORBA, COM (DCOM), JavaBeans. В якості мови інтерфейса в технологіях CORBA і DCOM використовується мова IDL (Interface Definition Language). Всі об'єкти згруповані у класи, кожний клас має свій ідентифікатор, кожний інтерфейс класа має також свій ідентифікатор. Класи об'єктів реалізуються у певному середовищі. Компоненти системного середовища об'єднані у декілька сценарієв (потоків процедур або маршрутів), у яких виділяються точки входу для вставлення специфічних користувацьких фрагментів і розширень. Існує можливість не

тільки вставляти нові фрагменти, але і замінювати вихідні компоненти в потоках процедур зі збереженням інтерфейсу.

Контрольні запитання

1. Дати визначення CASE-засобам і CASE-технологіям.
2. У чому полягають головні переваги CASE-технологій в розробці інформаційних систем?
3. Що являє собою методологія функціонального моделювання?
4. Дати визначення RAD-технології.
5. Дати визначення компонентно-орієнтованої технолгії.
6. Порівняти CASE-системи, які базуються на структурному і об'єктно-орієнтованому підходах.
7. Що таке репозиторій і яка його роль у інтеграції інформації?
8. Назвати сучасні CASE-системи і дати їм характеристику.

Частина 3. СУЧАСНІ ТЕХНОЛОГІЇ БАЗ ДАНИХ

Глава 10. РОЗПОДІЛЕНА ОБРОБКА ДАНИХ

10.1. Основні поняття і визначення

Режими використання БД у загальному вигляді показані на рис. 10.1.

Якщо з БД працюють одночасно декілька користувачів, то в цьому випадку СУБД повинна забезпечувати коректну паралельну роботу всіх користувачів над одними і тими ж даними. Розрізняють розподілену обробку і розподілені БД.

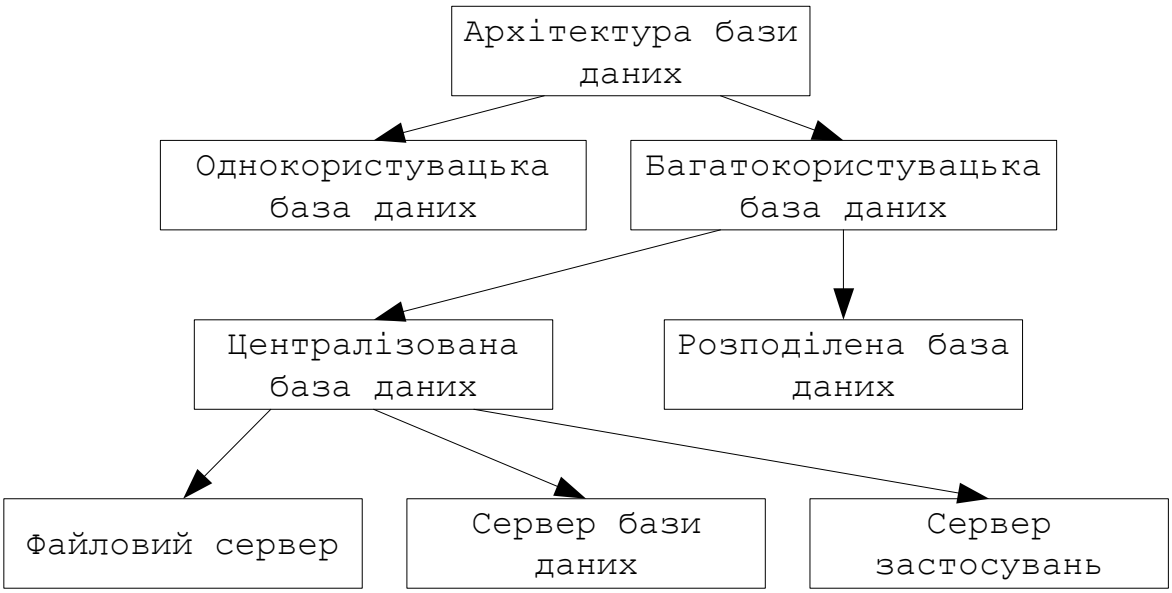


Рис. 10.1. Режими роботи з базою даних

Розподілена обробка – це обробка з використанням централізованої бази даних, доступ до якої може виконуватись з різних комп'ютерів мережі (рис. 10.2, *а*). Ця топологія часто називається "клієнт-сервер". В цій системі одні вузли – клієнти, а інші – сервери.

Сервер – комп'ютер, який надає деякі послуги іншим комп'ютерам, обмін повідомленнями з якими здійснюється за допомогою мережі, що їх з'єднує. Послуги полягають у наданні комп'ютеру, який звертається, ресурсів сервера (файлів, обчислювальних ресурсів і т.ін.) шляхом виконання вказаної програми і видачі результатів її роботи.

Клієнт – це процес, який посилає запит на обслуговування.

Розподілена база даних – це набір логічно зв'язаних між собою роздільних даних і їх описів, які фізично розподілені в деякій комп'ютерній мережі (рис. 10.2, *б*). Розподілена СУБД, в якій управління кожним із вузлів виконується зовсім автономно називається *мультибазовою системою*.

Розподілена СУБД – це програмна система, яка призначена для управління розподіленими базами даних і яка

забезпечує прозорий доступ користувачів до розподіленої інформації.

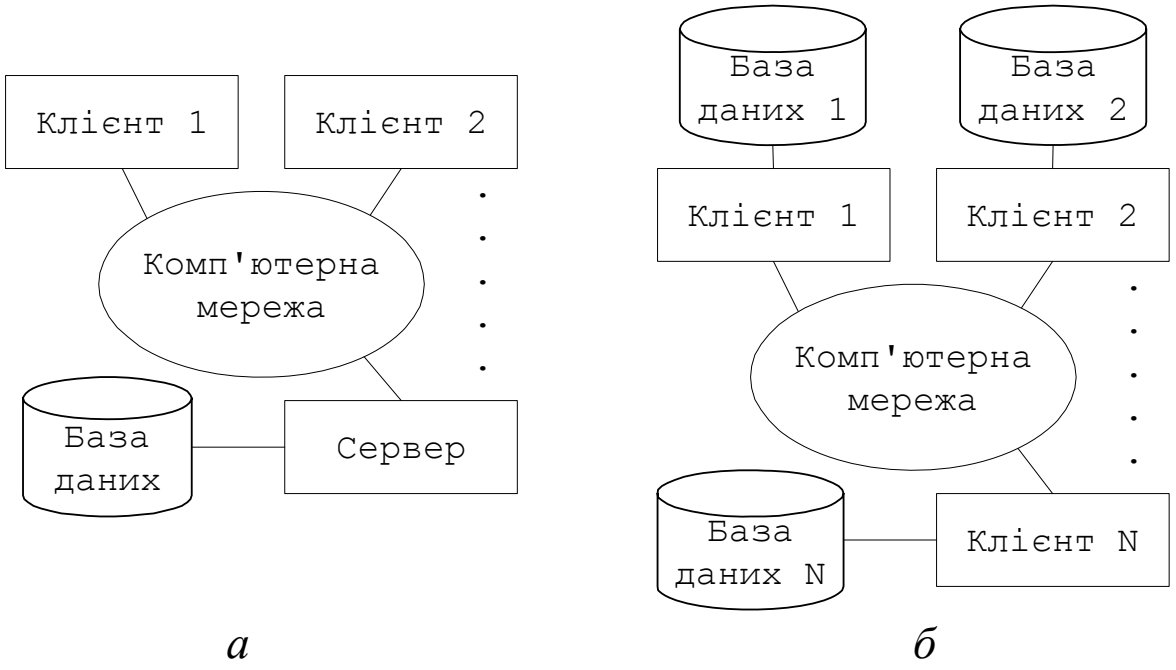


Рис. 10.2. Структура інформаційної системи:
а – розподілена обробка;
б – розподілена база даних

Якщо всі вузли розподіленої системи використовують той самий тип СУБД, то така система називається *гомогенною*. Якщо вузли розподіленої системи використовують різні типи СУБД, які обробляють різні моделі даних, то така система називається *гетерогенною*.

10.2. Управління паралельною обробкою

В багатокористувацьких системах з БД одночасно можуть працювати декілька користувачів або прикладних програм. Для збереження цілісності даних і забезпечення безпеки БД в цих умовах застосовуються транзакції, які забезпечують роботу кожного користувача з узгодженим станом БД.

Транзакція – неподільна з точки зору впливу на БД послідовність операторів маніпулювання даними, яка розглядається СУБД як єдине ціле. Або транзакція успішно

виконується, і СУБД фіксує зміни БД, які були зроблені цією транзакцією, у зовнішній пам'яті, або, у разі невдачі, жодна зміна не відображається на стані БД. Транзакція розглядається як логічна одиниця роботи з БД. Для того, щоби використання механізмів обробки транзакцій дозволило забезпечити цілісність даних й ізольованість користувачів, транзакція повинна мати такі властивості: атомарність (Atomicity), узгодженість (Cosistency), ізольованість (Isolation), довготерміновість (Durability). Транзакції, які мають ці властивості називаються ACID-транзакціями. Властивості транзакції означають таке:

- *атомарність* означає, що транзакція виконується, як єдина операція доступу до БД і виконується або повністю або не виконується зовсім;
- *узгодженість* гарантує взаємну цілісність даних, тобто виконання обмежень цілісності БД після завершення роботи транзакції;
- *ізольованість* означає, що транзакції, які конкурують за доступ до БД, фізично обробляються послідовно, ізольовано одна від одної, але для користувачів це виглядає так, ніби вони виконуються паралельно;
- *довготерміновість* означає, що коли транзакція виконана успішно, то всі зміни, які вона зробила в даних, не будуть втрачені ні за яких обставин.

Для обробки паралельних транзакцій застосовується серіалізація транзакцій і метод тимчасових міток.

Серіалізація транзакцій – процедура, яка забезпечує підтримку незалежного виконання трансакцій. Це означає, що дія двох паралельно діючих транзакцій буде така сама, як і їх послідовна дія: спочатку перша, а потім друга, або навпаки – спочатку друга, а потім перша. У ході виконання транзакції користувач бачить тільки узгоджені дані і не бачить

неузгоджених проміжних даних. Для підтримки паралельної роботи складається спеціальний план.

Для реалізації серіалізації транзакцій застосовується *механізм блокувань*. Блокування передбачає встановлення режиму доступу (монопольного або сумісного) до деякого ресурсу даних, що дозволяє виключити доступ до нього одночасно з даною транзакцією інших транзакцій, в результаті якого може бути порушена логічна цілісність даних БД. Об'єктом блокування може бути вся БД, окремі таблиці, сторінки, рядки.

Для підвищення ступеня паралельності доступу декількох користувачів до однієї БД використовуються такі блокування:

- *нежорстке* блокування або роздільне блокування (Shared – S-блокування); об'єкт блокується для виконання операції читання; об'єкти в цьому випадку не змінюються у ході виконання транзакції і доступні іншим транзакціям також, але тільки в режимі читання;
- *жорстке* блокування або монопольне (eXclusive – X-блокування); об'єкт блокується для виконання операції запису, модифікації або вилучення. В цьому випадку виконується монопольне блокування об'єкта і об'єкт залишається недоступним іншим транзакціям до моменту завершення роботи даної транзакції.

Застосування різних типів блокувань призводить до тупиків. *Тупикова ситуація* виникає тоді, коли дві і більш транзакції одночасно знаходяться у стані очікування, причому для продовження роботи кожна з транзакцій очікує завершення роботи іншої транзакції.

Приклад. Нехай транзакція 1 в момент часу t_1 блокує ресурс **A**, а транзакція 2 в момент часу t_2 блокує ресурс **B** (рис.10.3). В момент часу t_3 транзакції 1 потрібен ресурс **B** і вона очікує його звільнення транзакцією 2. В момент часу t_4 транзакції 2 потрібен ресурс **A** і вона очікує його звільнення

транзакцією 1. Транзакція 1 чекає транзакцію 2, транзакція 2 чекає транзакцію 1.

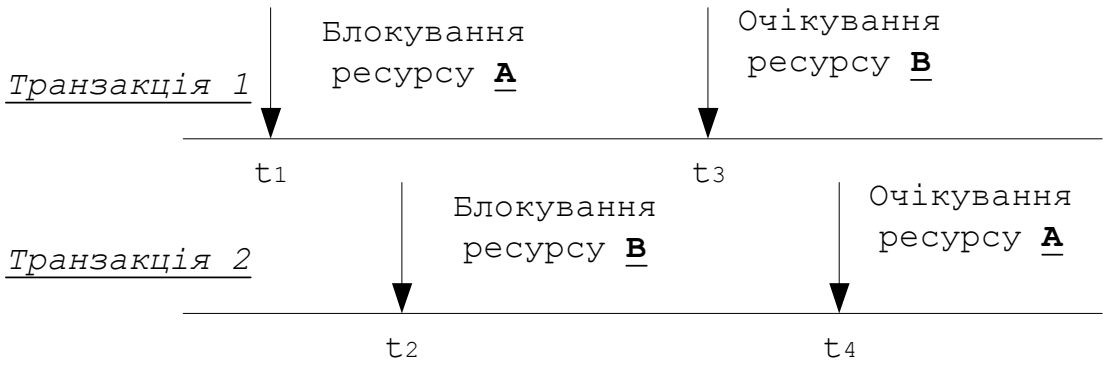


Рис.10.3. Взаємне блокування транзакцій

Основою визначення тупикових ситуацій є побудова графа очікування транзакцій. Алгоритм виходу із тупика передбачає визначення транзакції-жертви. Після вибору такої транзакції виконується її відкат.

Для серіалізації транзакцій також застосовується *двофазне блокування*, яке полягає у такому:

- перед виконанням операцій з будь-яким об'єктом транзакція блокує цей об'єкт (накопичення захватів);
- після зняття блокування транзакція не повинна накладувати ніяких інших блокувань (вивільнення захватів).

10.3. Багатокористувацькі СУБД

В якості сервера може розглядатися програма, яка надає деякі послуги по запитах інших програм, що називаються клієнтами. Взаємодія між клієнтами і сервером здійснюється за допомогою передачі повідомлень відповідно до заданого протоколу.

В технології роботи з БД виділяють функції:

- вводу і відображення даних (*представлення даних*);

– прикладні функції, які визначають основні алгоритми розв’язання прикладних задач (*застосування*);
– *обробки* даних у застосуваннях;
– управління інформаційними ресурсами (СУБД).

Представлення даних визначає те, що користувач бачить на своєму екрані. Тут визначаються екранні зображення, операції читання і запису даних, управління діалогом.

Прикладні функції (*бізнес-логіка*) визначають логіку роботи прикладних програм застосувань. Код застосування пишеться з використанням процедурних мов програмування.

Функції *обробки даних* пов'язані з обробкою даних всередині застосувань. Даними керує СУБД. Для забезпечення доступу до даних використовується мова SQL, яка найчастіше вбудовується в мови, які використовуються для створення коду застосування.

Функції *управління інформаційними ресурсами* – це СУБД, яка забезпечує зберігання і управління БД.

Залежно від того, де розташовані ці компоненти по відношенню одна до одної розрізняють монолітне виконання (найчастіше для персональних БД), дво- і трирівневе.

Дворівнева архітектура характеризується тим, що всі функції розподіляються між двома процесами, які виконуються на двох платформах: на клієнті і на сервері. В дворівневій архітектурі в свою чергу можлива реалізація таких моделей:

– модель файлового сервера;
– модель віддаленого доступу до даних;
– модель сервера бази даних.

У моделі *файлового сервера* (рис. 10.4) застосування розташовуються і виконуються на робочих станціях. На файловому сервері зберігаються тільки файли БД (файли даних, індекси і т.ін.) і деякі технологічні файли, які необхідні для роботи застосувань і самої СУБД. Клієнт звертається до СУБД на мові SQL, СУБД перекладає запит у послідовність файлових команд і передає файловому серверу. На кожну

команду сервер передає блок даних. Обробка інформації виконується на робочій станції за допомогою СУБД. Системи цього класу коштують недорого, прості в установці й освоєнні. До недоліків можна віднести таке:

- на кожній робочій станції знаходиться копія СУБД;
- низька продуктивність систем, оскільки всі проміжні дані передаються по мережі, а обробка виконується на робочих станціях, потужність яких значно поступається потужності сервера;
- складність розподіленої обробки.

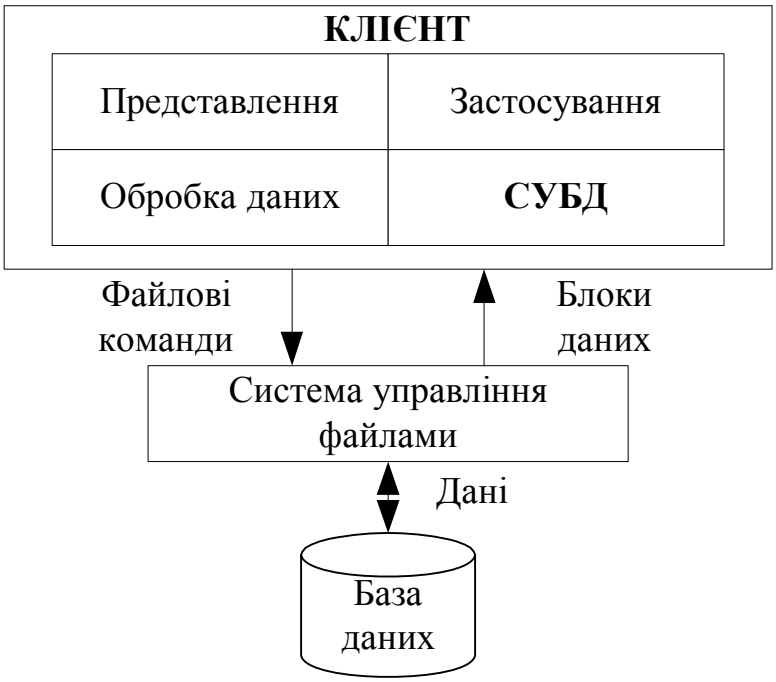


Рис. 10.4. Модель файлового сервера

У *моделі віддаленого доступу* до даних (рис. 10.5) БД і СУБД знаходяться на сервері, застосування розташовуються і виконуються на робочих станціях. Клієнт звертається до серверу на мові SQL. В цій архітектурі сервер виконує функції обробки транзакцій, даних і запитів. Сервер не перевантажений виконанням застосувань. Значно зменшується завантаження мережі у порівнянні з сервером файлів, оскільки по мережі від клієнта до сервера передаються команди на мові SQL, а не файлові команди, обсяг яких значно більший. Від сервера до

клієнта передаються дані, які відповідають запиту, а не блоки файлів. Недоліками моделей є таке:

- запити на мові SQL при інтенсивній роботі можуть значно завантажити мережу;
- прикладні функції застосувань необхідно повторювати для кожного клієнтського комп'ютера;
- сервер виконує пасивну роль і тому функції управління інформаційними ресурсами повинні виконуватись клієнтом.

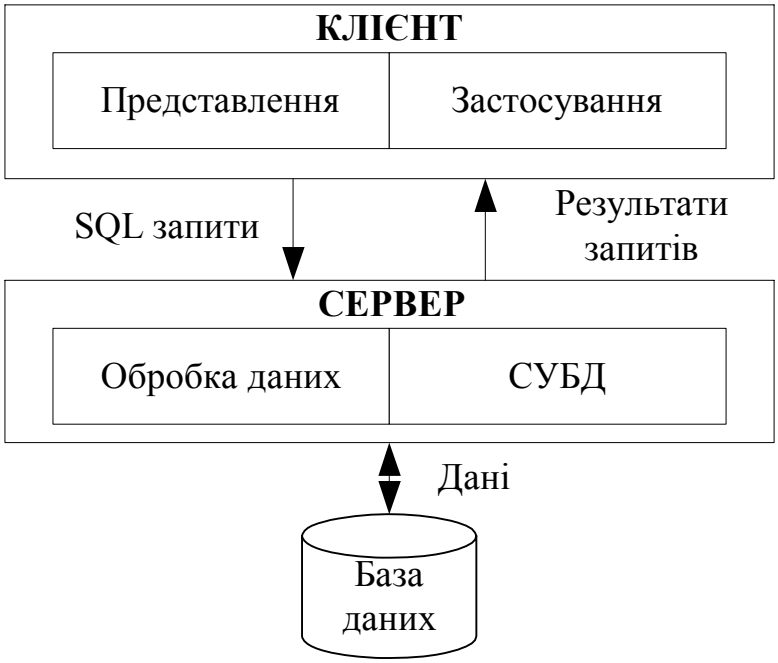


Рис. 10.5. Модель віддаленого доступу

Модель сервера бази даних (рис. 10.6) є подальшим розвитком моделі віддаленого доступу. Ця модель розширена механізмами процедур, що зберігаються і механізмами тригерів, які створюються на розширенні мови SQL.

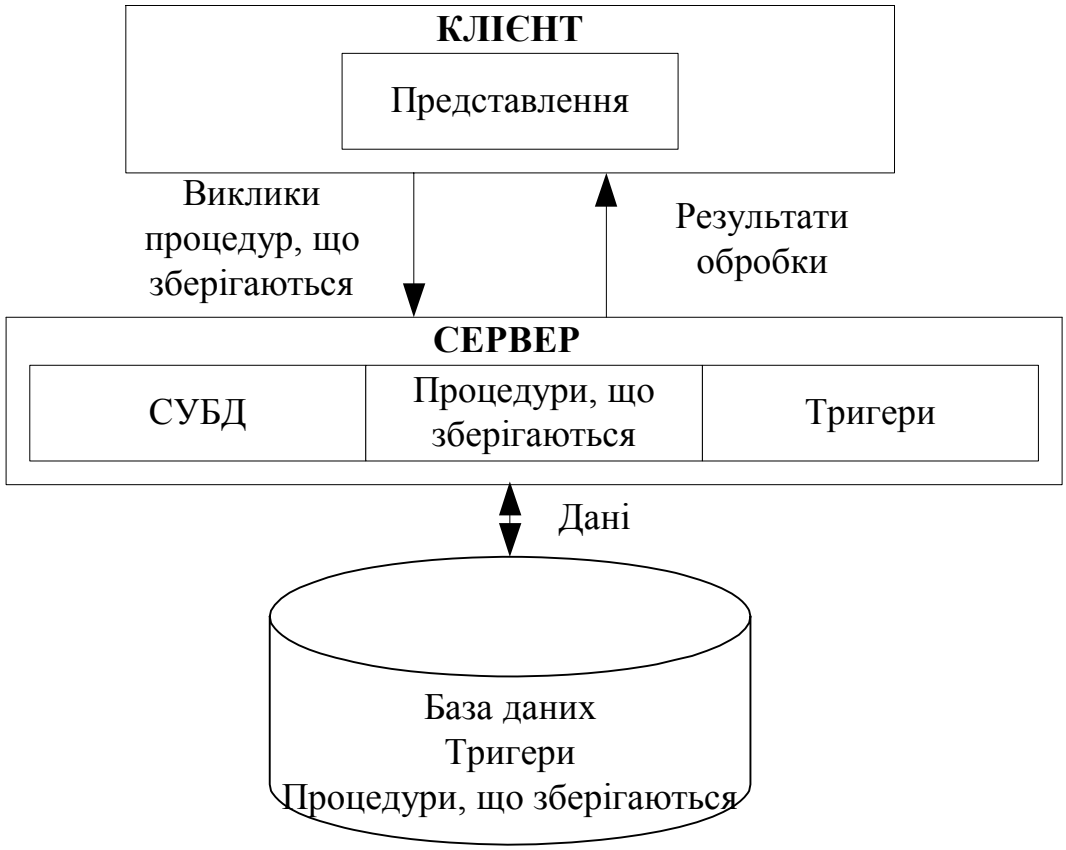


Рис. 10.6. Модель сервера бази даних

Процедури, що зберігаються є засобом програмування SQL-сервера і являють собою підпрограми, які можуть викликатися або застосуваннями користувачів, або тригерами. *Тригери* являють собою особливий тип процедури, що зберігається, яка самостійно реагує на виникнення певної події в БД. Тригер може активізуватися після операцій додавання, модифікації і вилучення. *Застосування тригерів дозволяє зробити сервер активним.* Це означає, що сам сервер може бути ініціатором обробки даних в БД.

Застосування процедур, що зберігаються і тригерів на сервері дозволяє їх використовувати різним клієнтам, що суттєво зменшує дублювання алгоритмів обробки даних. Недоліком цієї моделі в першу чергу є велике завантаження сервера.

Подальшим розвитком моделі сервера бази даних є трирівнева архітектура (рис.10.7). Ця архітектура ще називається *моделлю із сервером застосувань*.

Рис. 10.7. Модель сервера застосувань

Сервер застосувань – в триланковій архітектурі "клієнт-сервер" компонент прикладної системи, який розташовується між клієнтом і сервером БД і реалізує логіку застосування.

У цій моделі клієнт виконує тільки представлення інформації, сервер БД – виконує функції управління інформаційними ресурсами БД, забезпечує функції створення і ведення БД, підтримує цілісність БД. Сервер застосувань виконує загальні функції для клієнтів, забезпечує обмін повідомленнями, підтримує запити, зберігає і виконує найбільш загальні правила бізнес-логіки.

Перевагою моделі із сервером застосувань є гнучкість і універсальність внаслідок розділення функцій на три незалежні складові. Головним недоліком є більш високі витрати ресурсів комп'ютера на обмін інформацією між компонентами застосування у порівнянні з дворівневими моделями.

10.4. Проектування багатокористувацьких баз даних

Багатокористувацька БД передбачає, що у визначенні вимог до БД задіяні декілька користувачів. Залежно від того, як враховуються вимоги користувачів, розрізняють централізований підхід й інтеграцію представлень користувачів. *Централізований підхід* передбачає, що вимоги до кожного представлення користувача об'єднуються в загальний набір вимог (рис. 10.8).

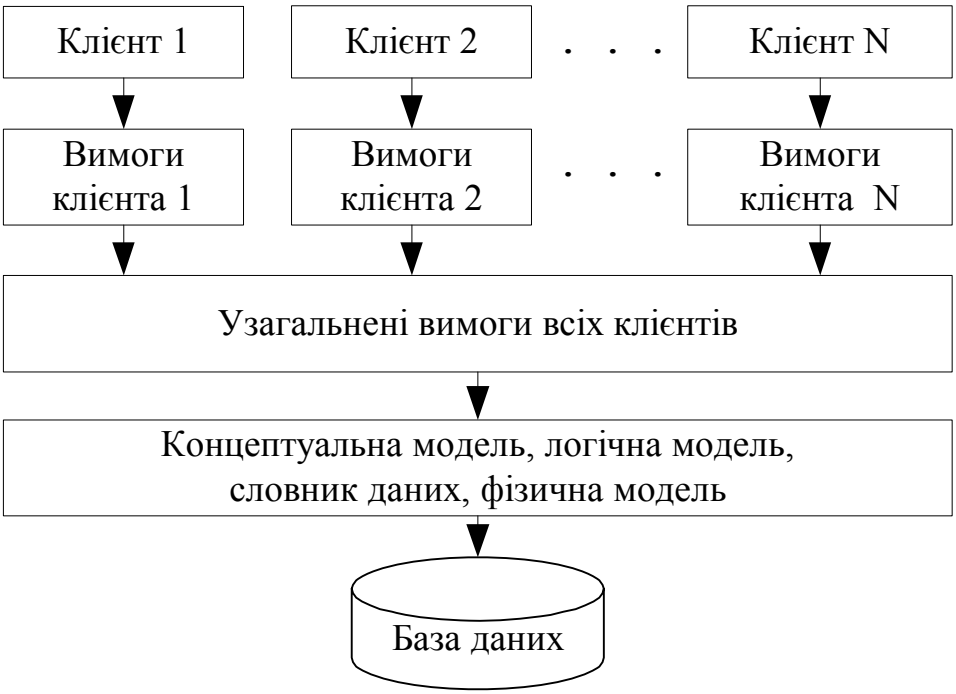


Рис. 10.8. Централізований підхід до проектування багатокористувацької бази даних

На етапі проектування БД створюється глобальна модель даних, яка відповідає загальному представленню. Глобальна модель також перевіряється, як і локальні моделі. Коректність

глобальної моделі перевіряється за допомогою правил нормалізації, що дозволяє переконатися в структурній узгодженості, логічній цілісності і мінімальній збитковості прийнятої моделі даних. Модель також перевіряється з метою виявлення можливостей виконання транзакцій, які будуть задаватися користувачами.

Інтеграція представлень користувачів передбачає, що вимоги до кожного представлення користувача застосовуються для створення окремої моделі даних, яка відповідає цьому представленню користувача. У подальшому, на етапі проектування БД, моделі даних, що отримані об'єднуються в єдину глобальну модель даних (рис. 10.9).

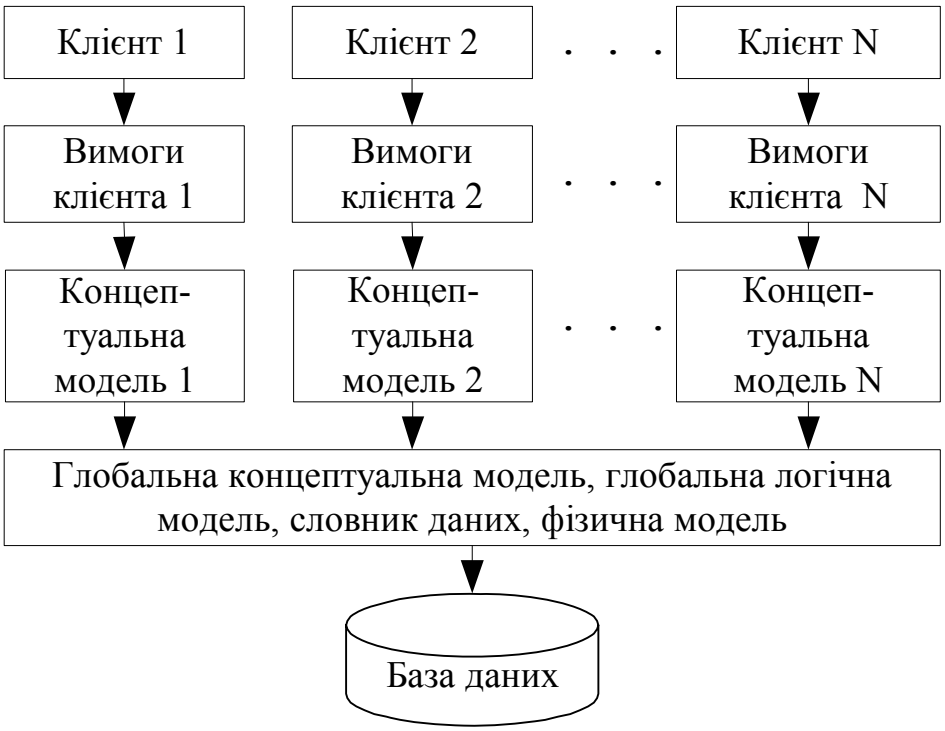


Рис. 10.9. Інтеграція представлень користувачів при проектуванні багатокористувацької бази даних

Після завершення об'єднання локальних моделей може виникнути необхідність перевірити вірність отриманої глобальної моделі, як у відношенні правил нормалізації, так і у відношенні можливості виконання транзакцій, передбачених специфікаціями всіх представлень користувачів. Це викликано

тим, що між окремими моделями може існувати несумісність або взаємне перекриття. Простий підхід до об'єднання декількох моделей передбачає, що спочатку об'єднуються дві моделі для отримання нової моделі, а потім до них послідовно додаються інші локальні моделі.

10.5. Проектування розподілених баз даних

На рис. 10.10 показана архітектура розподіленої СУБД.

Існують такі схеми розміщення даних в системі:

- централізоване;
- фрагментоване;
- з повною реплікацією;
- з вибірковою реплікацією.

Централізоване розміщення передбачає, що на одному з вузлів створюється і зберігається єдина БД. Доступ до цієї БД мають всі користувачі мережі.

Фрагментоване розміщення передбачає, що БД ділиться на неперетинні фрагменти, кожен з яких розміщується в одному з вузлів системи.

Розміщення з *повною реплікацією* передбачає, що повна копія БД розміщується на кожному вузлі системи.

Розміщення з *вибірковою реплікацією* являє собою комбінацію методів фрагментування, реплікації й централізації. Одні масиви даних поділяються на фрагменти, інші реплікуються, останні зберігаються централізовано.

Реплікація це процес генерації і відтворювання декількох копій даних, які розміщуються на одному або декількох вузлах. Використання реплікацій дозволяє досягнути багатьох переваг (продуктивність, надійність зберігання даних, відновлення даних і т.ін.).

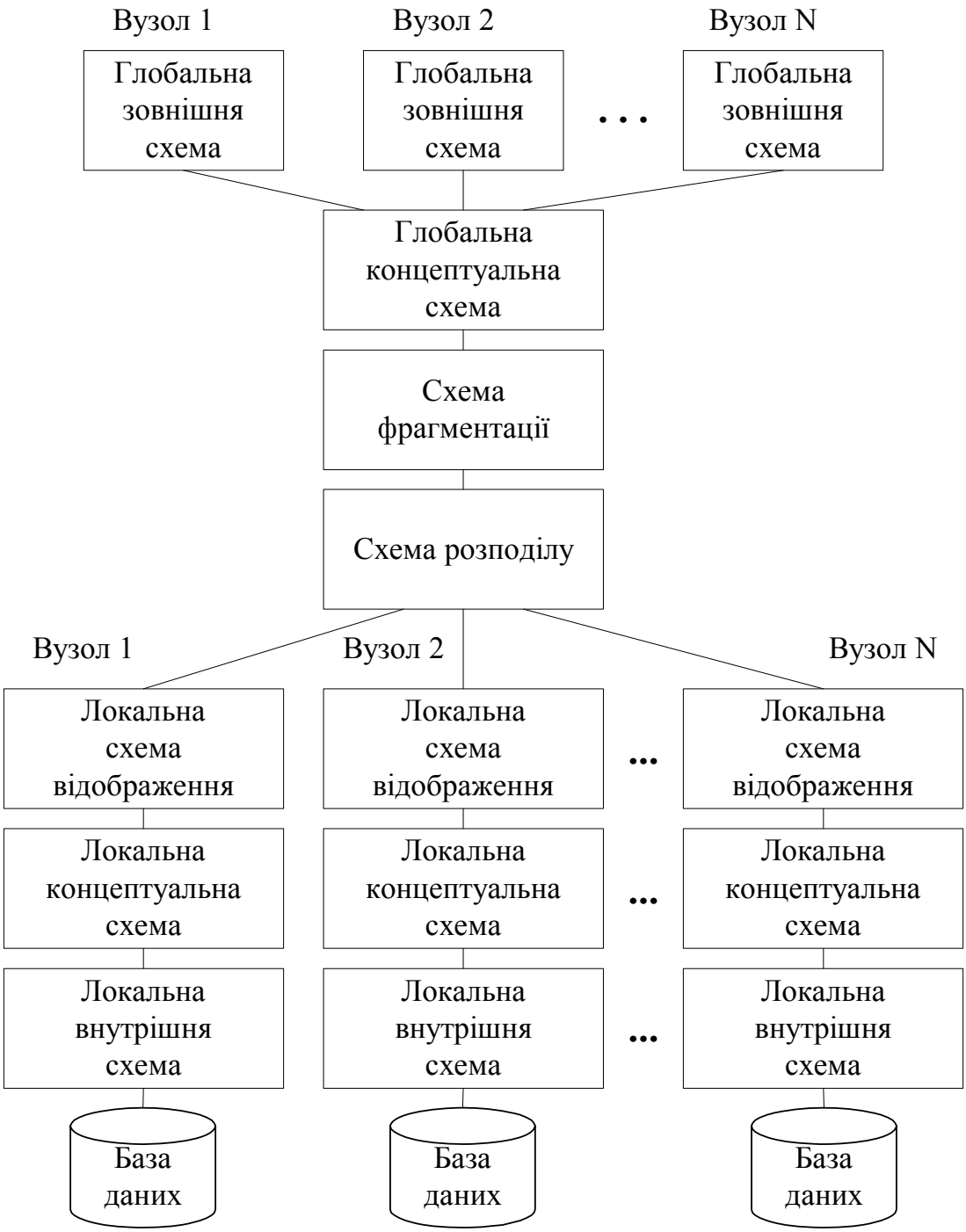


Рис. 10.10. Архітектура розподіленої СУБД

Оновлення даних, що реплікуються, може виконуватися синхронно й асинхронно. Синхронне оновлення передбачає, що всі копії реплікованих даних оновлюються одночасно з оновленням вихідної копії. Асинхронна реплікація передбачає,

що оновлення реплікованих даних виконується із затримкою відносно оновлення вихідних даних.

Система підтримує фрагментацію, якщо дане відношення може бути поділене на частини або фрагменти при організації його фізичного зберігання. Фрагментація бажана для підвищення ефективності системи. В цьому випадку дані можуть зберігатися в тому місці, де вони найчастіше використовуються. Це дозволяє досягнути локалізації більшості операцій і зменшити трафік мережі. Крім того, виключається зберігання даних, які не використовуються локальними застосуваннями, що сприяє підвищенню захисту в системі. Для коректності фрагментації необхідно виконати правила:

- *повнота* – це означає, що кожен елемент даних з вихідного відношення, повинен бути присутнім щонайменше в одному фрагменті;
- *підновленість* – це означає, що вихідне відношення може бути відновлено з його фрагментів за допомогою реляційної алгебри;
- *неперетинність* – це означає, що один елемент не повинен бути присутнім в двох і більше фрагментах.

Фрагментація може бути вертикальна (по атрибутах) і горизонтальна (по кортежах).

На рис.10.11 показана послідовність проектування розподілених БД. Відмінність проектування від централізованих БД полягає в застосуванні фрагментації відношень, реплікації фрагментів і розподілі фрагментів по вузлах мережі. Визначення і розміщення фрагментів повинно виконуватися з урахуванням особливостей використання БД. Під цим розуміється виконання аналізу транзакцій, необхідна продуктивність системи, підтримка цілісності даних і т.ін.

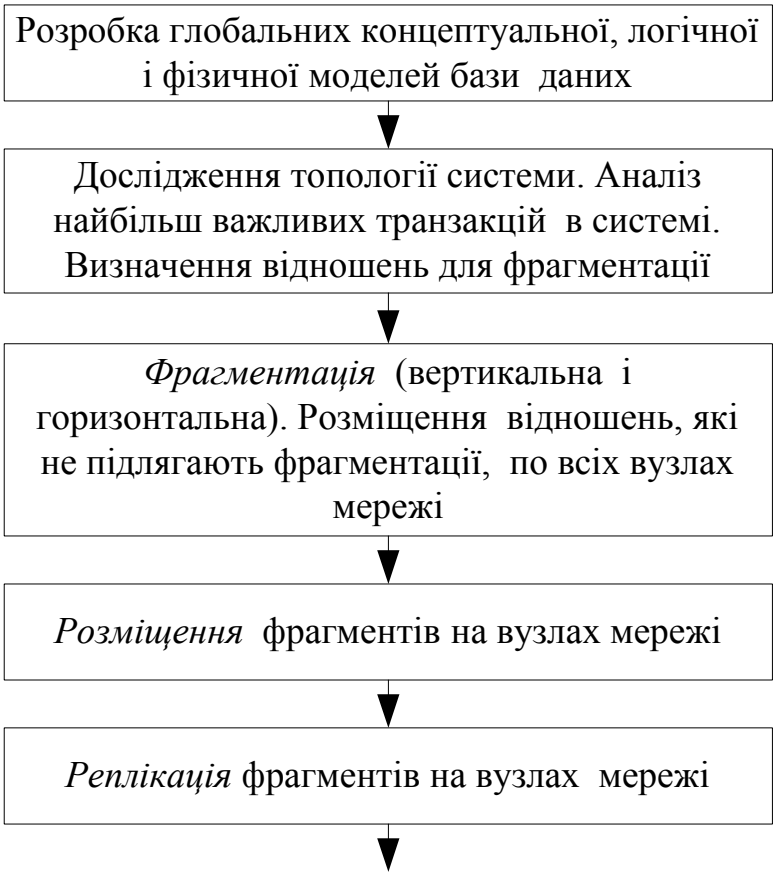


Рис. 10.11. Етапи проектування розподілених баз даних

Згідно з правилами Дейта розподілена СУБД повинна відповідати таким вимогам:

- розподілена система повинна виглядати з точки зору користувача, як звичайна нерозподілена система;
- вузли в розподіленій системі повинні бути незалежні або автономні;
- в системі не повинно бути жодного вузла, без якого система не може функціонувати;
- повинна виконуватись умова незалежності від розташування і користувач може отримувати доступ до БД з будь-якого вузла;
- незалежність від фрагментації – це означає, що користувач може отримати доступ до даних незалежно від засобу їх фрагментації;

– незалежність від реплікації – це означає, що користувач не має засобів доступу до конкретної копії даних і не займається питаннями оновлення всіх копій в БД;
– підтримка обробки розподілених запитів;
– підтримка управління розподіленими транзакціями;
– апаратна, програмна, мережна незалежність, а також незалежність від типу СУБД;
– забезпечення безперервного функціонування.

10.6. Стандартні інтерфейси доступу до серверів баз даних

Для доступу до БД застосовують такі інтерфейси:
– інтерфейс прикладного програмування (API, Application Programming Interface);
– інтерфейс рівня викликів (*SQL/CLI, SQL/Call Level* Interface);
– відкритий інтерфейс для підключення до БД (*ODBC*, Open Database Connectivity);
– інтерфейс прикладного програмування для мови Java (JDBC, Java Database Connectivity);
– об'єктно-орієнтований інтерфейс *OLE DB* (Object Linking and Embedding for DataBase);
– високорівневий об'єктно-орієнтований інтерфейс *ADO* (ActiveX Data Objects);
– високорівневий об'єктно-орієнтований інтерфейс *ADO.NET* (ActiveX Data Objects.NET) для доступу в платформі *.NET*.

Інтерфейс прикладного програмування – сукупність програмних засобів, які забезпечують прикладній програмі на традиційній мові програмування доступ до систем БД, який підтримується у середовищі конкретної СУБД. Реалізація API являє собою інтерфейс рівнів викликів. Вона включає бібліотеку функцій або процедур, які забезпечують зв'язок прикладної програми і СУБД.

Інтерфейс рівня викликів CLI – стандартизований інтерфейс прикладного програмування в *SQL*-серверах БД.

Стандарт ODBC – це інтерфейс, за допомогою якого прикладні програми можуть звертатися до *SQL*-баз даних і обробляти їх незалежним від СУБД засобом. Застосування може звертатися до БД, які підтримують різні СУБД, без необхідності його змінювати і перекомпільовувати (рис. 10.12).

Ошибка! Объект не может быть создан из кодов полей редактирования.

Рис. 10.12. Використання стандарту *ODBC* для доступу до даних

Стандарт JDBC – це інтерфейс, за допомогою якого прикладні програми на мові Java можуть звертатися до *SQL*-баз даних і обробляти їх незалежним від СУБД засобом. Цей стандарт є основою для створення інструментальних засобів і інтерфейсів більш високого рівня.

Стандарт OLE DB – це об'єктно-орієнтований інтерфейс, який дозволяє застосуванням сумісно використовувати і управляти наборами даних як об'єктами. *OLE DB* являє собою набір спеціалізованих об'єктів *COM* (*Component Object Model*), які включають в себе стандартні функції обробки даних і спеціалізовані функції конкретних джерел даних і інтерфейсів, які забезпечують передачу даних між об'єктами. Технологія *OLE DB* забезпечує доступ на низькому рівні до будь-яких джерел даних.

Стандарт ADO – це технологія, яка забезпечує механізми взаємодії об'єктів з даними і застосуваннями. *ADO* також виконує роль інтерфейса прикладного рівня з механізмами *OLE DB* (рис. 10.13).

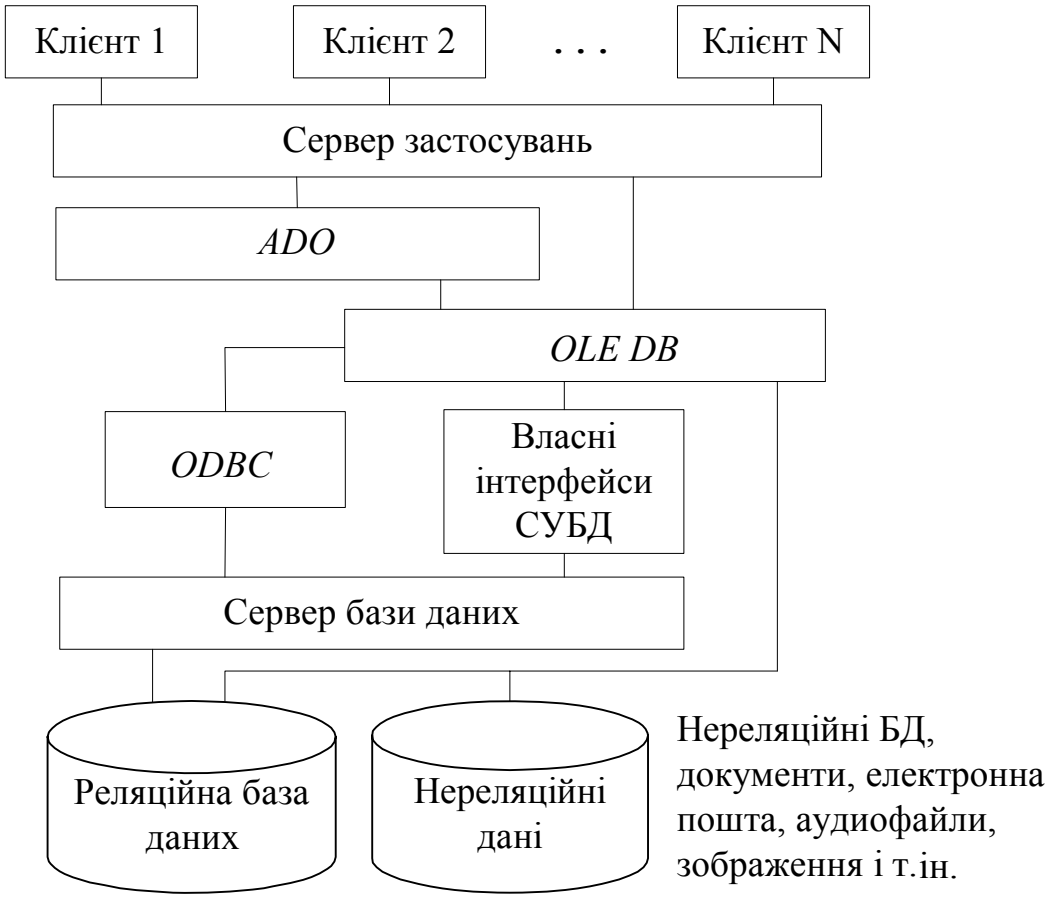


Рис. 10.13. Використання стандартів *ODBC*, *OLE DB*, *ADO* для доступу до даних

Стандарт *ADO.NET* – це нова версія технології *ADO*. *ADO.NET* являє собою набір класів, який використовується для доступу до джерел даних в платформі *.NET*. *ADO.NET* використовує в якості стандарта мову *XML* для передачі даних.

Контрольні запитання

1. Дати визначення термінам "розподілена БД" і "розподілена СУБД", перелічити їх основні властивості.
2. У чому полягає різниця між розподіленою СУБД і розподіленою обробкою?
3. Які головні функції виділяють в багатокористувацьких СУБД?

4. Назвати переваги і недоліки моделі файлового сервера БД.
5. Назвати переваги і недоліки моделі сервера БД.
6. Назвати переваги і недоліки моделі віддаленого доступу БД.
7. Порівняти дворівневі моделі багатокористувацьких СУБД.
8. Що таке тригер і процедура, що зберігається?
9. Дати характеристику трирівневій архітектурі.
10. Які функції виконує сервер застосувань?
11. Дати визначення транзакції та її основним властивостям.
12. Що таке серіалізація транзакцій?
13. Дати визначення поняттям: блокування, тупик.
14. Які особливості проектування багатокористувацьких БД.
15. У чому полягає фрагментація БД?
16. Що таке реплікація БД?
17. Перелічити головні етапи проектування розподілених БД.
18. Які проблеми виникають при паралельній обробці транзакцій?
19. Що таке інтерфейс і які його задачі?
20. Який зв'язок між *ODBC*, *OLE DB* і *ADO*?
21. Яке призначення *ODBC*?
22. Яке призначення *OLE DB*?
23. Яке призначення *ADO*?

Глава 11. ОБ'ЄКТНО-ОРІЄНТОВАНІ БАЗИ ДАНИХ

11.1. Основні поняття об'єктно-орієнтованих систем

Об'єктно-орієнтована база даних – база даних, яка створюється і використовується в середовищі СУБД, яке засновано на принципах об'єктно-орієнтованого підходу і підтримує об'єктну модель даних. Об'єктна база даних являє

собою сукупність взаємозв'язаних об'єктів, які відповідають певній схемі.

Об'єкт являє собою концептуальну модель реального світу, в яку вбудовані представлення даних (атрибути) і їх поведінка (методи). Кожен об'єкт має *унікальний ідентифікатор OID* (Object Identifier, ідентифікатор об'єкта), який не залежить від його атрибутів. Атрибути входять в склад об'єкта і кожен атрибут може посилатися на інший об'єкт. Стан об'єкта визначається набором значень, які об'єкт має в даний момент часу. Методи реалізують поведінку об'єкта. Методи викликають за допомогою повідомлень. Реалізація методів і представлення даних інкапсульовані, тобто приховані від зовнішніх джерел.

Схожі об'єкти групуються в *класи*. Клас являє собою колекцію об'єктів зі спільною структурою і поведінкою. Кожен об'єкт являє собою екземпляр класа або екземпляр об'єкта. Класи організовуються в *ієрархію класів*. Об'єкт наслідує атрибути і методи всіх своїх суперкласів.

Об'єктна модель даних характеризується рядом властивостей.

1. Основними компонентами є об'єкти і літерали. *Об'єкт* – це екземпляр сутності. Він має унікальний ідентифікатор. *Літерал* – конкретне значення. Він не має ідентифікатора.

2. Об'єкт має *властивості*, в тому числі атрибути і зв'язки з іншими об'єктами. Множина поточних значень всіх властивостей об'єкта визначає його *стан*.

3. Об'єкти і літерали мають *типи*. Кожен тип має власний домен. Типи можуть володіти поведінкою. Всі об'єкти одного типу володіють однаковою поведінкою і мають домени, що притаманні цьому типу.

4. Дії, які може виконувати об'єкт називаються *операціями*.

5. База даних зберігає об'єкти і заснована на схемі даних, що визначається мовою визначення даних. БД містить екземпляри типів, що визначаються схемою.

В табл. 11.1 наведені співвідношення термінів реляційної БД і об'єктно-орієнтованої БД.

Таблиця 11.1

Порівняння термінів реляційної й об'єктно-орієнтованої моделей

Реляційна база даних	Об'єктно-орієнтована база даних
Відношення	Клас
Кортеж	Екземпляр класу
Атрибут	Атрибут класу
Ієрархія відношень	Ієрархія класів
Підлегле відношення (відношення "нащадок")	Підклас
Головне відношення (відношення "батьківське")	Суперклас
Правила перетворення даних	Методи

Однією із суттєвих відмінностей об'єктних БД від реляційних є можливість створення і використання нових типів даних (абстрактні типи даних).

11.2. Проектування об'єктно-орієнтованих баз даних

Проектування об'єктно-орієнтованих баз даних (ООБД) має багато спільних рис з проектуванням реляційних БД і в цілому можна застосовувати методологію концептуального і логічного моделювання, яка застосовується для створення традиційних БД. Відмінності полягають у такому.

Процес проектування реляційних БД в основному направлений на ідентифікацію елементів даних, а не на визначення операцій з даними. Значна частина обмежень на дані і перетворення даних зазвичай розглядається на більш

пізніх етапах проектування БД і реалізується на програмному коді застосувань. *Операції не є частиною моделі БД*.

При проектуванні ООБД визначаються як дані, так і процедури їх обробки. На етапі реалізації обов'язково повинні використовуватися об'єктно-орієнтовані мови. При проектуванні ООБД дані і процедури вважаються єдиною сутністю і об'єкти розглядаються як окремі модулі.

Нормалізація зберігає своє значення і дозволяє зменшити збитковість даних. Для ідентифікації кожного об'єкта використовується *унікальний ідентифікатор об'єкта*. Кожен атрибут об'єкта залежить від ідентифікатора. Також для ідентифікації атрибутів об'єкта може вводитись первинний ключ. Схема БД містить опис структури даних об'єкта, обмеження, а також правила поведінки об'єкта.

В об'єктно-орієнтованих моделях є зв'язки двох типів:

- посилання між класами;
- наслідування в ієрархії класів.

Реляційні моделі використовують зв'язки на основі значень. Це означає, що зв'язки між сутностями встановлюються через спільні значення в одному або декількох атрибутах різних сутностей. На відміну від цього в об'єктно-орієнтованих моделях даних використовується підхід заснований на ідентифікаторах (*OID*), тому *зв'язки не залежать від стану об'єкта*.

В об'єктно-орієнтованих моделях даних створюється схема, в якій зв'язки складають частину структури БД. Об'єктно-орієнтовані БД є *навігаційними*: доступ до даних виконується за допомогою зв'язків, які зберігаються всередині самих даних. Об'єктно-орієнтовані моделі даних також передбачають доступ, орієнтований на множинність значень. Навігаційний доступ реалізується за допомогою ідентифікаторів *OID*.

Асоціативний доступ, орієнтований на множинність об’єктів, в об'єктно-орієнтованих моделях даних реалізується за

допомогою явно визначених методів. При проектуванні операції для маніпуляції екземплярами об'єктів в цьому випадку реалізуються в схемі БД.

Процес об'єктно-орієнтованого проектування є ітеративним і послідовним. На рис. 11.1 показана послідовність об'єктно-орієнтованого проектування.

Модель, що отримується в результаті проектування може мати зв'язки "один до одного", "один до багатьох", "багато до багатьох", а також рекурсивні зв'язки, що пов'язано з більшою потужністю об'єктно-орієнтованої моделі. В об'єктно-орієнтованій моделі даних зв'язки між об'єктами представляються атрибутами, які реалізуються об'єктними ідентифікаторами *OID*.

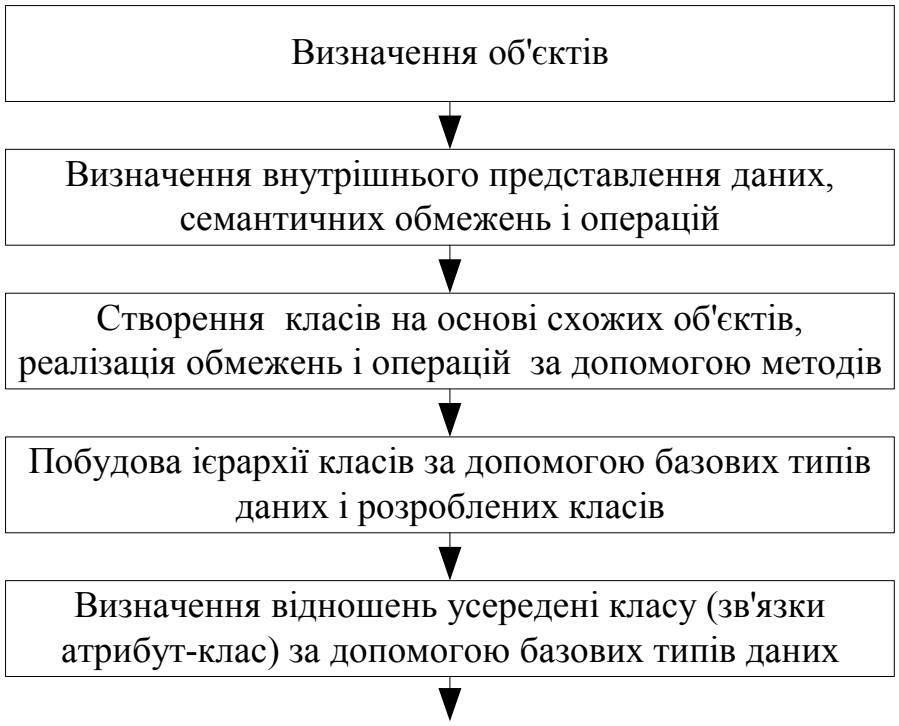


Рис. 11.1. Етапи проектування об'єктно-орієнтованої бази даних

Приклад. Зв'язок між об'єктами типу 1:1 представляється за рахунок додавання посилкових атрибутів у обидва об'єкти (рис. 11.2). Додаткові атрибути посилаються на ідентифікатор об'єкта (*OID*).

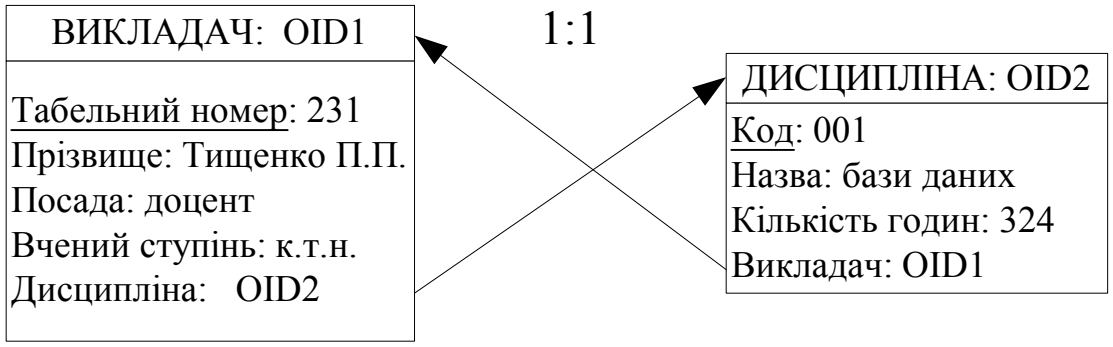


Рис. 11.2. Зв'язок "один до одного"

Зв'язок між об'єктами типу 1:M представляється за рахунок додавання посилкового атрибуту в перший об'єкт, який посилається на другий об'єкт, і атрибуту, який містить набір покажчиків на перший об'єкт, в другий об'єкт (рис. 11.3).

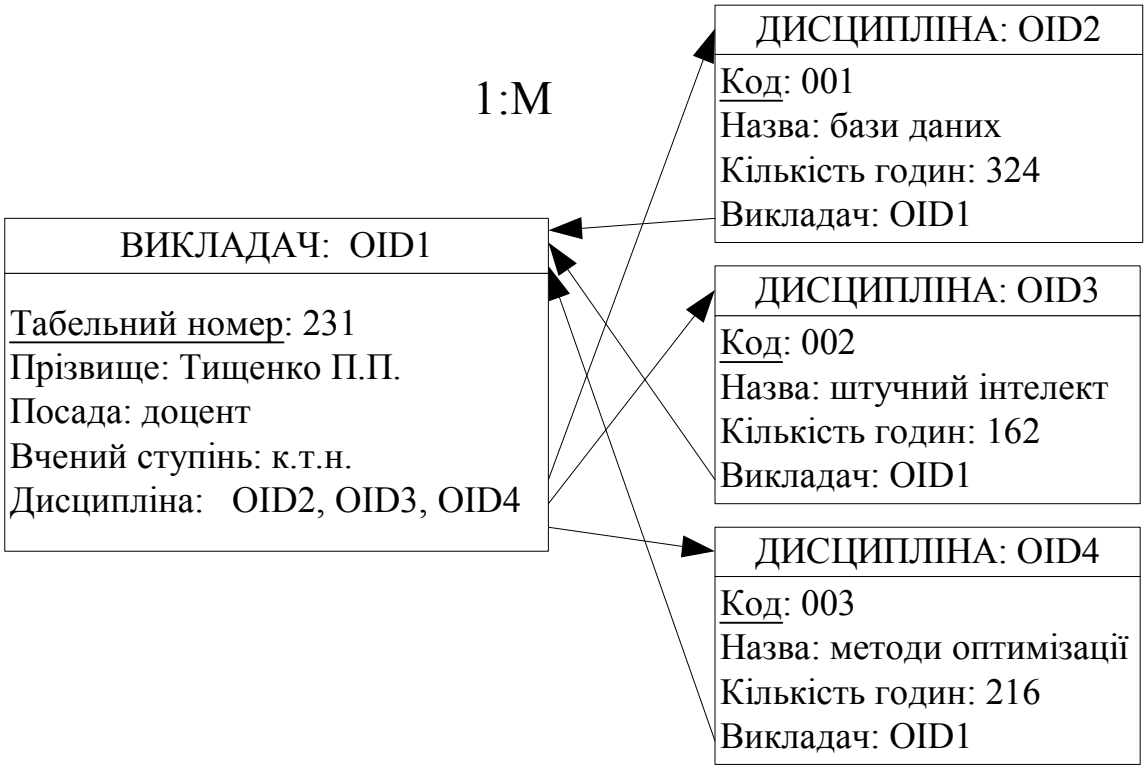


Рис. 11.3. Зв'язок "один до багатьох"

Зв'язок між об'єктами типу N:M представляється за рахунок додавання посилкового атрибуту, який містить набір покажчиків, в кожен об'єкт (рис. 11.4).

Об'єктно-орієнтовані СУБД являють собою результат комбінування об'єктно-орієнтованих можливостей (поліморфізм, наслідування, інкапсуляція) з можливостями БД (цілісність, безпека, управління транзакціями і паралельним виконанням, резервне копіювання, відновлення, маніпулювання даними).

Об'єктно-орієнтована СУБД повинна відповідати таким вимогам:

- підтримка складних об'єктів;
- підтримка унікальної ідентифікації об'єктів (OID);
- підтримка класів або типів;
- підтримка наслідування;
- інкапсуляція об'єктів;
- управління БД;
- забезпечення паралельної роботи декількох користувачів;
- можливість відновлення після перебоїв;
- підтримка запитів до БД на мові високого рівня;
- запис і зчитування даних із зовнішніх пристроїв пам'яті.

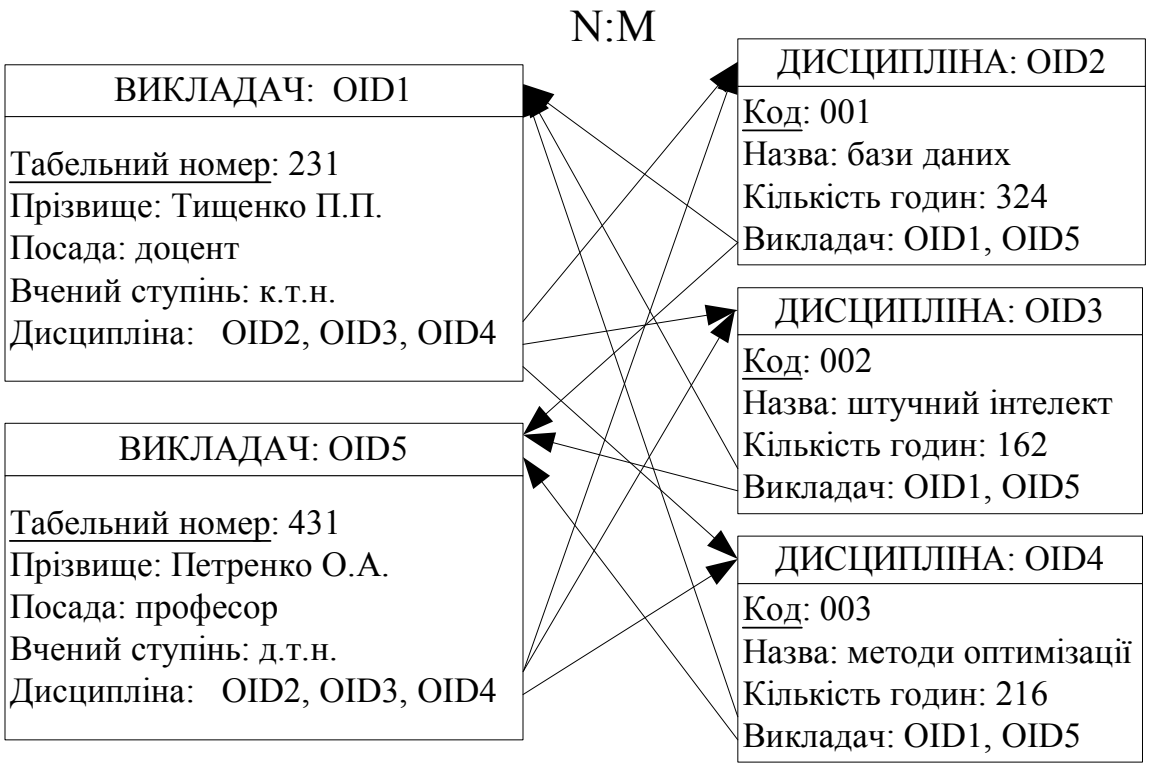


Рис. 11.4. Зв'язок "багато до багатьох"

Питаннями розвитку об'єктно-орієнтованих систем займається *OMG* (Object Management Group) – робоча група по розвитку стандартів об'єктного програмування. Цією групою розроблена мова *UML* (Unified Modeling Language), яка використовується для моделювання компонентів БД. Групою *OMG* також були створені стандарти об'єктів *OMA* (Object Management Architecture), які допускають взаємодію об'єктів на різних платформах і системах.

До переваг об'єктно-орієнтованих баз даних можна віднести такі:

- можливість включення детальної семантичної інформації в БД;
- можливість розширення базових типів даних, підтримка складних вкладених структур, визначення довільних структур (мультимедіа і т.ін.);
- контроль версій;

– можливість багаторазового використання класів, прискорення розробки застосувань за рахунок наслідування.

До недоліків об'єктно-орієнтованих баз даних належить:

– відсутність достатньо пророблені теоретичної бази у об'єктної моделі;
– складність структури;
– навігаційне управління даними, складність додавання атрибутів і методів;
– відсутність стандартної мови запитів.

11.3. Об'єктно-реляційні бази даних

Об'єктно-реляційна БД – база даних, в якій підтримується як реляційне, так і об'єктне представлення. В основі цих баз лежить розширення реляційної моделі. До характеристик цієї моделі можна віднести:

– визначення користувачем абстрактних типів даних;
– підтримку наслідування;
– можливість створювати більш складні у порівнянні з реляційною БД моделі;
– застосування унікальних ідентифікаторів, що генеруються системою.

Об'єктно-реляційні БД мають такі особливості:

– підтримують структурні типи атрибутів (множини. мультимножини і т.ін.), тобто компонент кортежу одного відношення здатний, в свою чергу, зберігати інше відношення;
– дозволяють визначати спеціальні операції, які виконуються над даним деякого типу, який оголошений користувачем;
– кортежі виконують функції об'єктів, тому у багатьох випадках вони мають унікальні ідентифікатори, які

дозволяють відрізняти один кортеж від іншого при рівних значеннях у всіх компонентах;

– дозволяється використовувати посилання на кортежі відношення.

Контрольні запитання

1. Дати визначення поняттям: *об'єкт*, *стан об'єкта*, *ідентифікатор об'єкта*, *метод*, *повідомлення*.
2. У чому полягає різниця між реляційною моделлю і об'єктною моделлю предметної області?
3. Які основні властивості об'єктної моделі?
4. Дати визначення поняттям: *клас*, *суперклас*, *підклас*, *ієрархія класів*.
5. Перелічити переваги і недоліки ООБД.
6. У чому полягає різниця між об'єктно-орієнтованими СУБД і реляційними СУБД?
7. Описати властивості об'єктно-реляційної моделі. Які її особливості у порівнянні з об'єктно-орієнтованою?
8. Перелічити етапи проектування ООБД.
9. Що є спільного і в чому різниця між проектуванням реляційних БД і ООБД?
10. Як реалізується зв'язок 1:1, 1:M, N:M в об'єктно-орієнтованих моделях?

Глава 12. СХОВИЩЕ ДАНИХ

12.1. Організація сховищ даних

Сховище даних (*Data Warehouse*, *DW*) – система, що підтримує несуперечливу інтегровану предметно-орієнтовану сукупність історичних даних організації з метою підтримки

прийняття стратегічних рішень. Сховище даних представляє також різнобічні інструментальні засоби для аналізу даних.

Концепція сховищ даних − це концепція підготовки даних для подальшого аналізу. Інформаційні сховища призначені для систем підтримки прийняття рішень. Сховища даних розробляються з урахуванням специфіки предметної області, а не застосувань, які обробляють дані. Дані у сховищі повинні бути інтегровані, зведені до єдиного синтаксичного і семантичного вигляду, перевірені на цілісність і несуперечливість.

В основі концепції сховищ даних лежить ідея розподілу на дві групи даних, що використовуються: для оперативної обробки (*OLTP*) і для рішення задач аналізу (*OLAP*).

OLTP (*On-Line Transaction Processing*) − системи оперативной обробки транзакцій, які призначені для підтримки поточної діяльності різного роду організацій.

OLAP (*On-Line Transaction Processing*) − системи оперативної аналітичної обробки, які призначені для підтримки прийняття рішень і орієнтовані головним чином на нерегламентовані запити. Термін *OLAP* дозволяє описувати технологію обробки даних, в якій застосовується багатомірне представлення агрегованих даних для забезпечення швидкого доступу до даних для поглибленого аналізу.

Порівняльний аналіз *OLTP* і *OLAP* систем наведено в табл. 12.1.

Архітектура сучасних сховищ даних базується або на використанні багатомірної моделі БД (Multidimension *OLAP*, *MOLAP*), або на реляційній моделі БД (Relational *OLAP*, *ROLAP*).

Таблиця 12.1

Порівняльний аналіз OLTP і OLAP систем

Характеристика	*OLTP*-система	*OLAP*-система

Ступінь деталізації даних, що зберігаються	Зберігання тільки деталізованих даних	Зберігання як деталізованих, так і узагальнених даних
Управління даними	Управління даними в будь-який час	Періодичне додавання даних
Допущення збитковості даних	Забезпечується максимальна нормалізація даних	Допускається контрольована денормалізація даних
Характер запитів до даних	Доступ до даних по заздалегідь складеним запитам	Запити до даних можуть бути довільні і заздалегідь не оформлені
Частота оновлення	Висока частота, маленькими порціями	Мала частота, великими порціями
Вік даних	Поточні (до одного року)	Історичні (за декілька років) і прогнозні

Складність створення сховищ даних викликала необхідність розробки і організації підмножин даних сховища, які називаються кіосками даних.

Кіоск даних (*вітрина даних*) – спрощений варіант сховища даних, який містить тільки тематично об'єднані дані. Кіоск даних максимально наближений до кінцевого користувача і містить дані орієнтовані на нього.

Одна з найважливіших частин сучасних аналітичних систем – це засоби інтелектуального аналізу даних. Виконання більшості аналітичних запитів користувачів потребує складної статистичної обробки, застосування штучного інтелекту.

Data Mining – дослідження і знаходження комп'ютером (засобами штучного інтелекту) в даних прихованих

закономірностей, які не були раніше відомі, нетривіальні, практично корисні, доступні для інтерпретації людиною.

На рис. 12.1 показана логічна схема аналітичної системи зі сховищем даних.

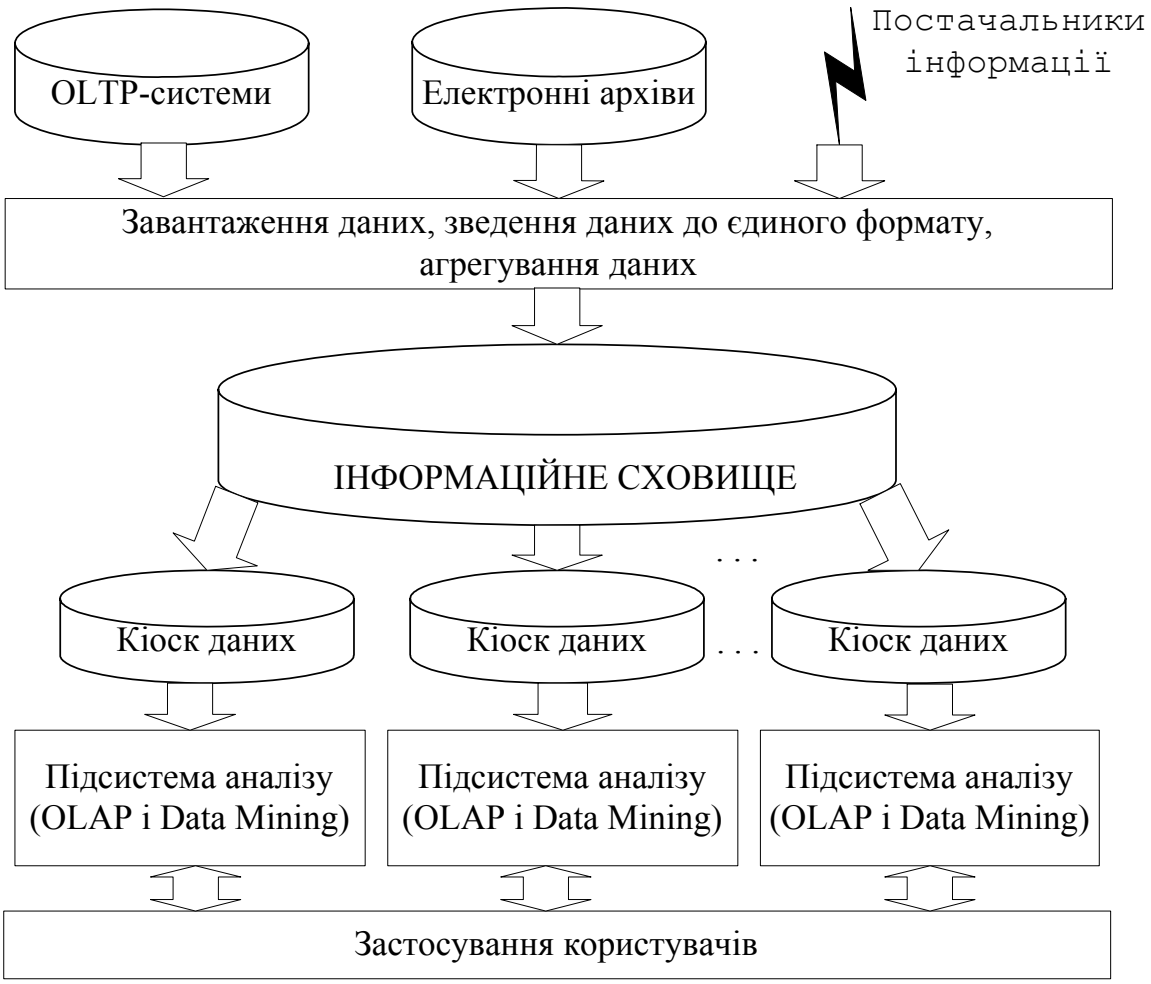


Рис. 12.1. Схема аналітичної системи зі сховищем даних

12.2. Багатомірна модель сховища

Багатомірна модель передбачає, що дані зберігаються не у вигляді плоских таблиць, як в реляційній БД, а у вигляді *гіперкубів* – впорядкованих багатомірних масивів. Багатомірне

представлення даних тут реалізується фізично. Багатомірні СУБД забезпечують більш швидкий у порівнянні з реляційними системами пошук і читання даних. В цьому випадку немає потреби у багаторазовому з'єднанні таблиць. Такий підхід вимагає більше пам'яті для зберігання даних, при його використанні важко модифікувати структуру даних.

У багатомірній моделі розглядаються такі операції маніпулювання даними:

- *переріз*, який передбачає формування підмножини гіперкуба, в якому значення одного або більшої кількості вимірів є фіксованим;
- *обертання*, при якому змінюється порядок представлення вимірів;
- *згорнення*, передбачає заміну одного з вимірів іншим більш високого рівня ієрархії;
- *деталізація* – це операція зворотна до згорнення і забезпечує перехід від узагальнених даних до деталізованих.

Багатомірна СУБД краще за інші системи виконує складні *нерегламентовані запити*.

12.3. Проектування сховищ даних

При створенні сховища даних однією з основних задач є визначення оптимальної структури зберігання даних з точки зору забезпечення прийнятного часу відповіді на аналітичні запити і потрібного об'єму пам'яті.

Всі дані в сховищі даних поділяються на категорії:

- детальні дані;
- агреговані дані;
- метадані.

Детальні дані – дані, які переносяться безпосередньо від оперативних джерел інформації (*OLTP*). Вони відповідають елементарним подіям, що фіксуються в звичайних БД. Всі дані поділяються на виміри і факти. *Вимірами* називаються набори

даних, які необхідні для опису подій (студенти, факультети і т.ін.). Вимір є аналогом домену в реляційній моделі. Виміри грають роль індексів для ідентифікації конкретних значень в комірках гіперкуба. *Фактами* називаються дані, які відображають сутність події (результати екзамену, кількість студентів і т.ін.). Непотрібні детальні дані можуть зберігатися в архівах у стислому вигляді.

Агреговані дані – дані, які отримують агрегуванням детальних даних по певних вимірах. Частина агрегованих даних безпосередньо зберігається в сховищі даних, а не обчислюється при виконанні запитів.

Метадані – це високорівневі засоби відображення інформаційної моделі. Метадані містять таку інформацію: опис структури даних сховища, структури даних, які імпортуються з різних джерел, відомості про періодичність імпортування, методах завантаження і узагальнення даних, засобах доступу і правилах представлення інформації, оцінки витрат часу на отримання відповіді на запит. Метадані знаходяться в *репозиторії* метаданих.

Послідовність проектування сховища даних показана на рис. 12.2.

Розмірності встановлюють контекст для пошуку відповідей на питання, що стосуються фактів в таблиці фактів. Вдало підібрані розмірності дозволяють зробити магазин даних зрозумілим і легким у використанні. Одна і та ж розмірність в різних магазинах даних повинна бути однаковою, або бути підмножиною іншої розмірності. Всі факти повинні бути визначені на відповідному рівні деталізації.

Існують інші підходи до створення сховища даних. Один з найбільш поширених передбачає декомпозицію проекту сховищ даних на магазини даних з подальшою інтеграцією інформації.

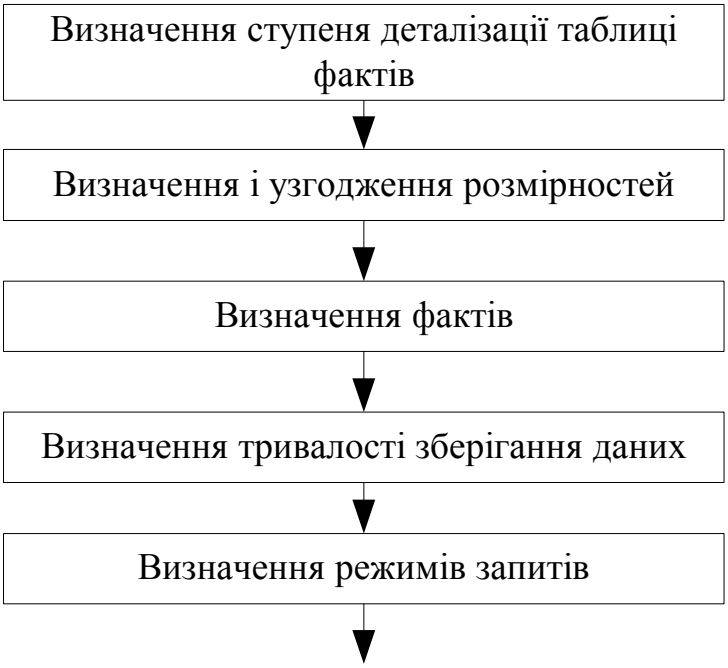


Рис. 12.2. Послідовність проектування сховища даних

При моделюванні сховищ даних використовуються концепції ER-моделювання з деякими обмеженнями. Кожна модель складається з таблиці зі складовим ключем, яка називається *таблицею фактів*, і набору невеликих таблиць, які називаються *таблицями розмірностей*. У таблиці фактів розміщуються дані, які найбільш інтенсивно використовуються для аналізу. Запис фактологічної таблиці відповідає комірці гіперкуба. У довідковій таблиці перелічені можливі значення одного з вимірів гіперкуба. Кожен вимір описується своєю власною таблицею.

Кожна таблиця розмірності має простий первинний ключ, який точно відповідає одному з компонентів складового ключа в таблиці фактів. Тобто первинний ключ таблиці фактів складається з декількох зовнішніх ключів. Така централізована структура називається схемою "зірка".

Приклад. Розглянемо організацію сховища даних для вищих навчальних закладів України. За вимірювання візьмемо такі величини:

– параметри, що характеризують діяльність ВНЗ (кількість студентів, конкурс і т.ін.);

– опис ВНЗ (назва, факультети, спеціальності і т.ін.);

– момент часу (рік, квартал, місяць і т.ін.).

Кількість можливих параметрів, ВНЗ, а також моментів часу, що розглядаються, кінцева, тому всі значення можливо представити у вигляді гіперкуба. У цьому гіперкубі кожне значення знаходиться в окремій комірці. На рис. 12.3 наведена спрощена схема структури сховища даних.

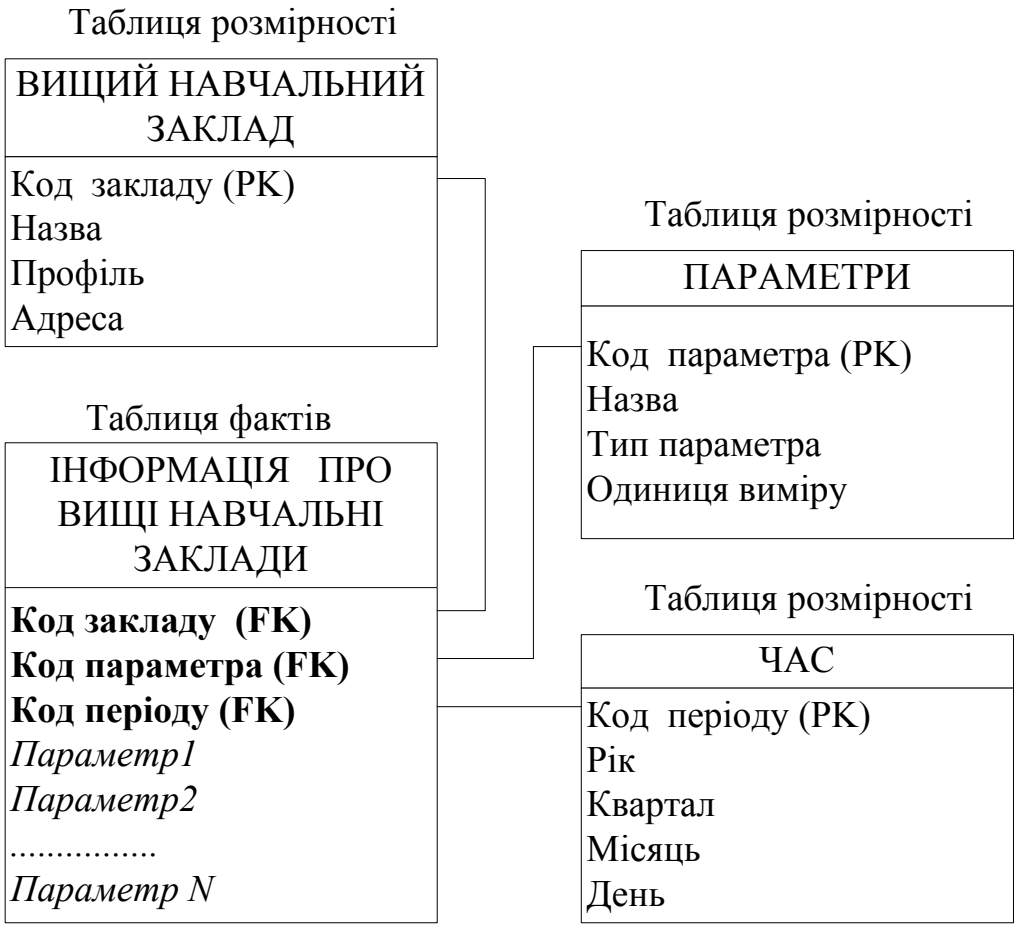


Рис. 12.3. Приклад бази даних з радіально зв'язаними таблицями

Приклади запитів до сховища даних: "Визначити середню успішність студентів в технічних університетах", "Як змінився конкурс студентів на економічні спеціальності за останні п'ять років?"

Якщо БД включає велику кількість вимірів, то можна використовувати схему "сніжинка". В цій схемі атрибути

таблиць розмірності можуть бути деталізовані у додаткових довідкових таблицях.

Контрольні запитання

1. Порівняти задачі, які виконують *OLTP*-системи і *OLAP*-системи.
2. Що таке сховище даних і які його основні характеристики?
3. Що таке багатомірний куб?
4. Що собою являють ієрархії атрибутів і рівні агрегації в схемі "зірка"?
5. Які проблеми виникають при реалізації сховища даних?
6. Які головні етапи проектування сховища даних?
7. Що таке кіоск даних?
8. Які головні операції маніпулювання даними виконуються в сховищі даних?
9. Які дані зберігаються в сховищі даних?
10. В яких випадках доцільно застосовувати сховища даних, а в яких недоцільно?

Частина 4. ЕКСПЛУАТАЦІЯ БАЗ ДАНИХ

Глава 13. ЗАСТОСУВАННЯ БАЗ ДАНИХ

13.1. Адміністрування базами даних

Адміністрування даними передбачає виконання функцій адміністратора даних. Адміністратор даних відповідає за достовірність та повноту даних, що знаходяться в БД, їх узгодженість, а також виконання регламенту робіт по актуалізації БД.

Адміністрування бази даних передбачає виконання функцій адміністратора БД та інших адміністративних функцій, які забезпечують життєдіяльність системи бази даних (рис. 13.1).

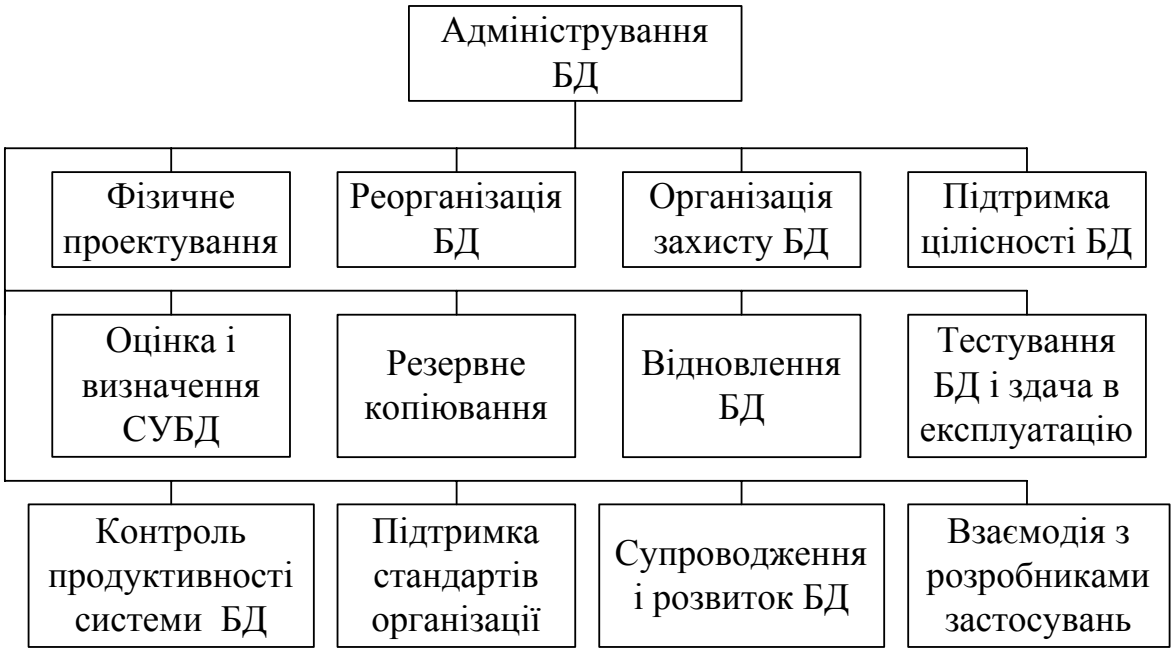


Рис. 13.1. Задачі адміністрування бази даних

Адміністратор БД відповідає за забезпечення необхідного рівня продуктивності системи. Ці задачі вирішуються шляхом використання ефективних методів доступу, раціональною стратегією розміщення даних на носіях і оптимальною збитковістю даних. Адміністратор БД вирішує задачі пов'язані з вибором розміщення файлів на диску, визначенням необхідного об'єму дискової пам'яті, розподілом інформації на диску.

Створення нормалізованих відношень є важливою складовою частиною проектування БД, але крім позитивних результатів, нормалізація може призводити до зниження продуктивності роботи системи. Виконання вимог нормалізації породжує велику кількість таблиць, збільшення таблиць призводить до збільшення кількості операцій обміну даними з диском і збільшенню часу на обробку логічних зв'язків. При певних режимах роботи з БД можливе деяке зниження рівня нормалізації і за рахунок цього підвищення швидкості обробки

даних. Суперечність між ефективністю проекта, інформаційними потребами користувачів і швидкістю обробки інформації вирішується шляхом пошуку компромісу, який, як правило, полягає в деякій *денормалізації бази даних*.

Адміністратор БД відповідає також за збір і обробку статистики функціонування системи, забезпечення ефективного використання ресурсів, за надійність функціонування системи, оцінку необхідності переналагодження середовища зберігання БД та її виконання, відновлення стану БД при порушенні логічної і фізичної цілісності.

Реорганізація бази даних – діяльність адміністратора БД, яка забезпечує збір сміття, необхідні зміни для розміщення даних і, можливо їх структури, згідно з проведеною модифікацією схеми зберігання. Реорганізація БД дозволяє підвищити продуктивність системи БД і/або більш ефективно використовувати ресурси простору пам'яті середовища зберігання.

Резервне копіювання (*backup*) виконується для запобігання можливого руйнування БД і, у разі потреби, відновлення даних. Виконується копіювання спеціальними утілітами. Резервуються не тільки дані, але і службова інформація (журнали транзакцій, словники і т.ін.). Резервна копія може бути точною копією БД, або архівною копією. Резервне копіювання може виконуватись під час роботи з БД або в інший час.

Тестування являє собою процес виконання прикладних програм з метою пошуку помилок.

Експлуатація і супроводження полягають в промисловому використанні створеної системи, яке постійно супроводжується перевірками її поточних покажчиків функціонування, а також необхідною підтримкою.

Стандарти організації це правила, які використовуються в процесі розробки і експлуатації системи: принципи структурування прикладних програм, погодження про найменування, які використовують програмісти і т.ін.

Задачі *визначення СУБД* полягають у визначенні таких питань: модель БД, ємкість БД, продуктивність, мобільність і стандартизація, необхідне обладнання, вартість, підтримка розробки застосувань, засоби адміністрування БД, безпека і цілісність, резервне копіювання і відновлення і т.ін.

13.2. Відновлення баз даних

Відновлення бази даних – усунення порушень логічної або фізичної цілісності БД з метою забезпечення її подальшого використання.

Цілісність бази даних логічна – властивість стану БД, яка характеризується відсутністю порушень всіх обмежень цілісності, які явно визначені в логічній схемі.

Цілісність бази даних фізична – властивість стану БД, яка характеризується відсутністю порушень специфікацій схеми зберігання, а також фізичних руйнувань даних на носіях інформації.

Причини руйнування БД, головні функції і методи відновлення БД показані на рис. 13.2. Головною одиницею відновлення в системах з БД є транзакції. Транзакція у разі успішного завершення виконує операцію Commit, а у разі невдачі – операцію Rollback. При виконанні роботи з БД СУБД використовує буфер. *Буфер БД* – це область основної пам'яті для тимчасового зберігання даних, яка призначена для прискорення операцій з диском.

Рис. 13.2. Засоби для організації відновлення бази даних

У відновленні БД ключову роль грає журнал, в якому зберігаються зміни внесені в БД, а також стан кожної транзакції.

Журнал – функціональний компонент СУБД, який забезпечує реєстрацію в процесі функціонування системи БД відомостей про виконання транзакцій, про працюючих користувачів, про застосування що використовуються, про доступ до різних структур даних і т.ін.

Журнал забезпечує фіксацію всіх дій транзакцій: старт транзакції, зміну значень елементів БД, результатів або припинення роботи. Інформація журналу і зміни, які вносяться в елементи даних повинні виконуватися синхронно, але конкретний спосіб синхронізації залежить від обраного режиму протоколювання. Існують два основних методи відновлення БД: з відкладеним оновленням і негайним оновленням.

Відновлення з використанням *відкладеного оновлення* передбачає, що оновлення не заносяться в БД до тих пір, поки транзакція не видасть команду фіксації зроблених змін. Якщо виконання транзакції було закінчено до досягнення цієї точки, то ніяких змін в БД виконано не буде, тому і не потрібна їх відміна (рис. 13.3).

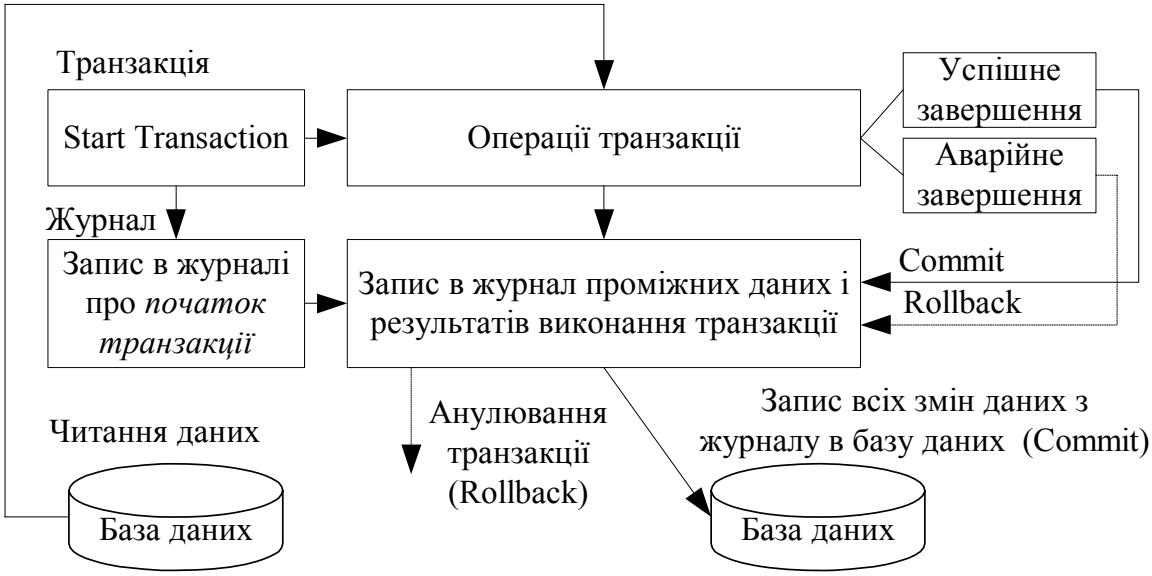


Рис.13.3. Схема відновлення з використанням відкладеного оновлення

Відновлення з використанням *негайного оновлення* передбачає, що всі зміни вносяться в БД одразу після їх виконання в транзакції, ще до досягнення моменту фіксації зроблених змін. Якщо виконання транзакції було закінчено до досягнення цієї точки (аварійно), то для відміни внесених змін у БД використовується журнал. Відміна операцій транзакцій виконується у зворотному порядку (рис. 13.4).

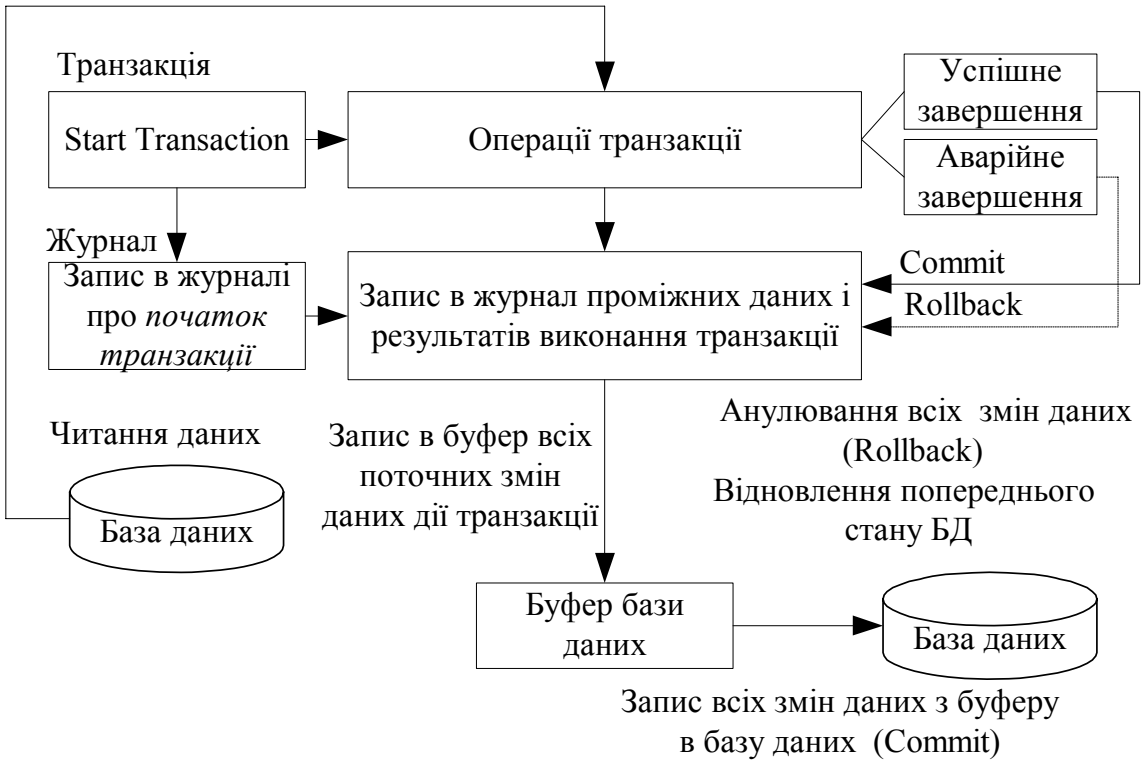


Рис. 13.4. Схема відновлення з використанням негайного оновлення

Контрольна точка – сукупність даних, яка зв'язана з деякою транзакцією в певний момент часу, і зберігається для відновлення (у разі необхідності) стану БД і транзакції, що виконується на цей момент, з метою забезпечення можливості повторного її використання починаючи з цього моменту. Створення контрольних точок під час виконання транзакцій дозволяє виконувати при необхідності її відкат не на початок, а до останньої за часом контрольної точки, завдяки чому суттєво підвищується продуктивність системи.

Засоби ведення журналу дозволяють захистити систему від наслідків втрати даних в оперативній пам'яті. Для відновлення БД від руйнування дискових пристроїв застосовується *архівування*. Архів являє собою повну або часткову копію БД, яка зберігається у безпечному місці.

Відновлення це процес, який складається з таких етапів:

– створення пустої БД із заданими параметрами (розміром сторінки, режимом запису і т.ін.);

– додавання метаданих (таблиць, різних обмежень і перевірок і т.ін.);
– наповнення даними з файлу резервної копії;
– створення необхідних індексів.

Контрольні запитання

1. Які основні задачі адміністрування базами даних?
2. У чому полягає задача реорганізації бази даних?
3. Як можна підвищити продуктивніть роботи системи бази даних?
4. Які головні методи відновлення бази даних?
5. Порівняти методи відновлення бази даних з відкладеним оновленням і з негайним оновленням.
6. Які головні переваги застосування контрольних точок?
7. Які головні функції журналу бази даних?

Глава 14. ЗАХИСТ ІНФОРМАЦІЇ В БАЗАХ ДАНИХ

Захист даних – попередження несанкціонованого або випадкового доступу до даних, їх зміни або руйнування даних з боку користувачів; попередження змін або руйнування даних при збоях апаратних і програмних засобів і при помилках в роботі співробітників групи експлуатації.

Наслідками порушення захисту є матеріальні збитки, зниження продуктивності системи, отримання таємних відомостей, руйнування БД. Головними причинами порушення захисту є несанкціонований доступ, некоректне використання ресурсів, перебої в роботі системи. В БД застосовуються засоби захисту наведені на рис. 14.1.

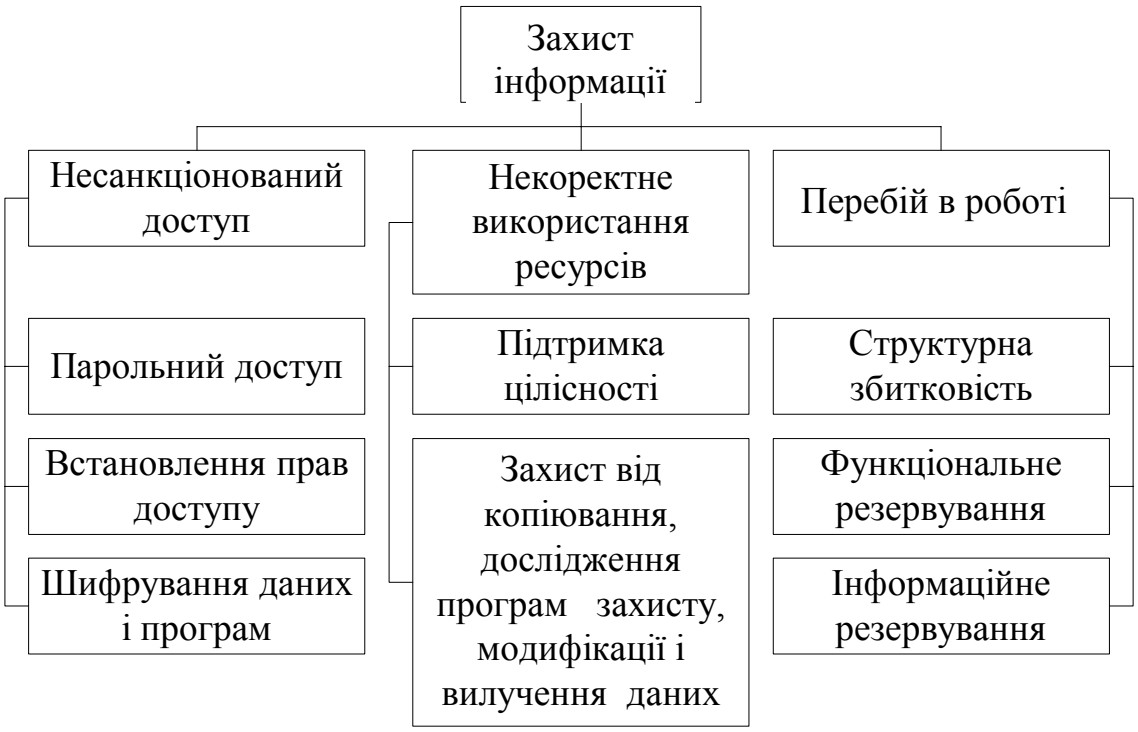


Рис. 14.1. Засоби захисту інформації в базах даних

Парольний захист передбачає встановлення пароля, який являє собою ряд літер, визначених користувачем при вході в систему з метою *ідентифікації* системними механізмами управління доступом. Після ідентифікації виконується аутентифікація. *Аутентифікація* – процедура, яка встановлює автентичність ідентифікатора, який висувається користувачем, програмою, процесом, системою при вході в систему, при запиті деякої послуги або при доступі до ресурсу. Процес аутентифікації це обмін між користувачем і системою інформацією, яка відома тільки системі і користувачу.

Управління доступом – захист інформаційних ресурсів в системі БД від несанкціонованого доступу. Мета полягає в забезпеченні представлення доступу до ресурсів тільки користувачам, програмам, процесам, системам, які мають відповідні права доступу.

Як тільки користувач отримає право доступу до СУБД, йому автоматично представляються різні привілеї, пов'язані з його ідентифікатором. Ці привілеї можуть включати дозвіл на доступ до певних таблиць, представлень, індексів, а також на

певні операції над цими об'єктами: перегляд (SELECT), додавання (INSERT), оновлення (UPDATE), вилучення (DELETE) (рис. 14.2).

Рис. 14.2. Рівні доступу до бази даних

У сучасних СУБД підтримуються два найбільш загальних підходи до забезпечення безпеки даних: вибірковий підхід і обов'язковий підхід.

Вибірковий підхід передбачає, що кожен користувач має різні права при роботі з даними об'єктами. Різні користувачі можуть мати різні права доступу до одного й того ж об'єкта. Такий підхід це однорівнева модель безпеки (табл. 14.1).

Обов'язковий підхід передбачає, що кожному об'єкту даних надається деякий кваліфікаційний рівень, а кожен користувач має деякий рівень допуску. При такому підході правом доступу до певного об'єкта даних мають тільки

користувачі з відповідним рівнем доступу. Такий підхід являє собою багаторівневу модель безпеки (рис. 14.3).

Приклад. Одноріvнева модель безпеки.

Таблиця 14.1

Одноріvнева модель безпеки

	Таблиця 1	Таблиця 1	. . .	Таблиця N
Користувач 1	Read, Update	–	…	Read, Write, Delete
Користувач 2	Read	Read, Write, Insert	…	Update, Insert
…	…	…	…	…
Користувач N	Read, Insert	Update	…	–

Багаторівнева модель безпеки

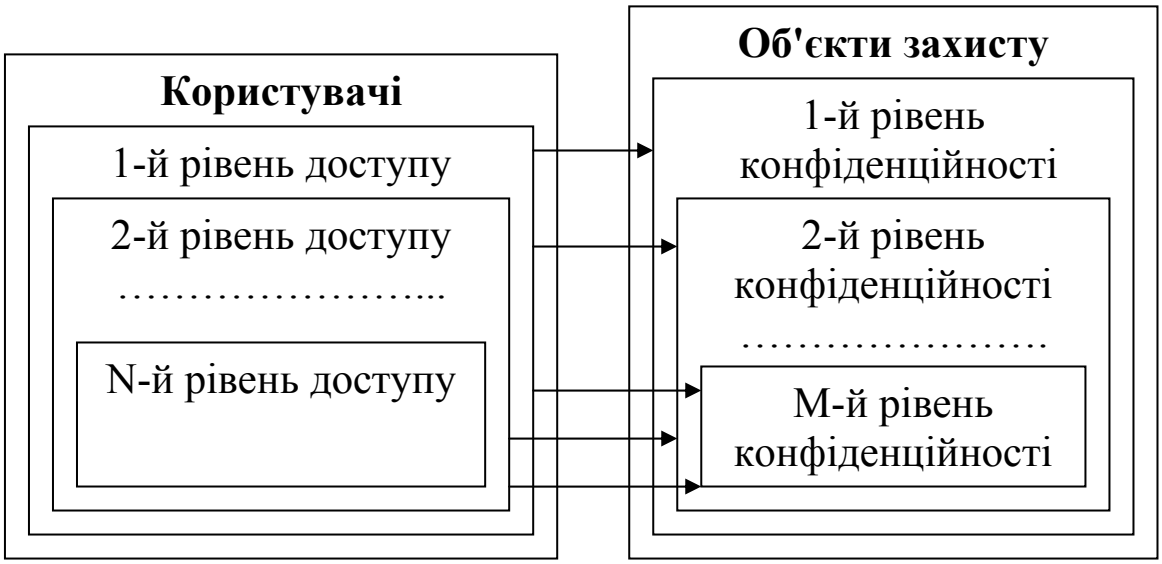


Рис. 14.3. Організація багаторівневої моделі безпеки

Шифрування даних – метод забезпечення таємності даних шляхом генерації нового їх представлення, яке допускає однозначне відновлення вихідного представлення.

Система шифрування включає в себе ключ шифрування, алгоритм шифрування, ключ дешифрування і алгоритм дешифрування.

Захист від перебоїв і відказів роботи апаратно-програмного забезпечення є однією з необхідних умов нормальної роботи системи. Головне навантаження по забезпеченню захисту системи від перебоїв і відказів припадає на апаратно-програмні компоненти. *Структурна збитковість* передбачає резервування апаратних компонентів на різних рівнях: дублювання серверів, процесорів, накопичувачів на магнітних дисках (RAID – масив недорогих дискових накопичувачів зі збитковістю), використання джерел безперебійного живлення. *Функціональне резервування* означає організацію обчислювального процесу, при якій функції обробки даних виконуються декількома елементами системи. *Інформаційне резервування* реалізується шляхом періодичного копіювання і архівування даних.

Під *некоректним використанням ресурсів* розуміють некоректні зміни в БД (порушення посилкової цілісності, введення невірних даних і т.ін.), випадковий доступ прикладних програм до чужих розділів основної пам'яті і т.ін. Вирішуються ці проблеми шляхом підтримки цілісності БД, використанням захисту від копіювання, дослідження програм, модифікації і вилучення даних. Тут застосовуються як прикладні програми, так і засоби операційної системи.

Контрольні запитання

1. Яке призначення і область застосування поняття "захист бази даних"?
2. Дати пояснення термінам: *ідентифікація*, *аутентифікація*, *шифрування*.
3. Що таке одноріvнева модель безпеки?
4. Що таке багаторівнева модель безпеки?
5. Що таке управління доступом?
6. Охарактеризувати методи захисту баз даних.

ГОЛОВНІ ТЕНДЕНЦІЇ РОЗВИТКУ БАЗ ДАНИХ

Сучасний ринок СУБД за рік приносить більше 10 мільярдів доларів прибутків від продажу продуктів і послуг і постійно збільшується. Технології баз даних безперервно розвиваються, а ринок СУБД насичується і консолідується.

Реляційна технологія складає ядро корпоративних систем обробки даних, а реляційні бази даних використовуються практично у всіх великих компаніях, організаціях, установах.

До сегментів ринку, що найбільш швидко розвиваються за останній час в галузі баз даних належать напрями:

- інтеграції неоднорідних інформаційних ресурсів; інтеграція структурованих і неструктурованих даних;
- побудова сховищ даних;
- OLAP-системи, які призначені для комплексного аналізу даних і знаходження прихованих тенденцій розвитку;
- мобільні БД, які призначені для роботи на портативних комп'ютерах;
- вбудовані БД, які є інтегрованою частиною застосування;
- мікробази даних для різних пристроїв, таких як кредитні картки, електронні записні книжки і т.ін.;
- резидентні БД, які використовуються у надпродуктивних OLTP-системах.

До напрямів, які постійно знаходяться у полі зору розробників баз даних належать:

- удосконалення архітектури систем баз даних, розробка нових архітектурних рішень систем баз даних;
- організації середовища зберігання і методів доступу;
- розвиток технології і техніки моделювання даними;
- методи оптимізації запитів;
- питання експлуатація баз даних (безпека даних, відновлення даних, підтримка цілісності даних);

– розвиток технологій розробки застосувань.

Одним з найбільш важливих напрямів розвитку баз даних є *зближення технологій баз даних, WEB-технологій і технологій текстових систем*. Середовище WEB забезпечує доступ до великих обсягів інформації в Інтернеті, однак крім накопичення слабкоструктурованої інформації необхідно забезпечувати ефективне управління цією різноманітною інформацією. Розробка платформи XML забезпечує нові можливості для ефективного доступу до інформаційних ресурсів цього середовища, а також для створення нових технологій інтеграції ресурсів.

Поєднання технологій баз даних і WEB-технологій дозволяє вирішувати такі задачі:

– організація взаємозв'язку СУБД, які працюють на різних платформах;
– використання в Інтернеті інформації з існуючих локальних мережних баз даних (публікація баз даних);
– застосування СУБД для впорядкування і каталогізації інформації в Інтернеті;
– застосування в інформаційних систем в Інтернеті на основі багаторівневої архітектури баз даних;
– застосування мови *SQL* для пошуку необхідної інформації;
– використання засобів СУБД для забезпечення безпеки даних, управління транзакціями при доступі до інформації в Інтернеті;
– стандартизація користувацького інтерфейсу на основі застосування оглядачів WEB, використання оглядача WEB в якості дешевої клієнтської програми для доступу до бази даних.

Наступним напрямом в розвитку СУБД є розвиток і подальше вдосконалення управлінням *об'єктно-орієнтованими базами даних*. Сучасні виробники СУБД або будують свої системи виключно на основі об'єктно-орієнтованої моделі, або

включають в свої програмні продукти об'єктно-реляційні можливості.

Одним з важливих класів інформаційних систем є *геоінформаційні системи*. Ці системи мають справу з геоданими, тобто даними про *просторово-розподілені об'єкти*, процеси, події. Ці системи широко застосовують в картографії, економіці, екології, управлінні ресурсами оточуючого середовища, транспорті і т.ін. Основу інформаційних ресурсів геоінформаційних систем складають *просторові дані*. Розрізняють моделі геоданих векторні і растрові, а також застосовуються традиційні фактографічні дані. Подальший розвиток цих систем полягає у вдосконаленні моделей бази даних (геореляційні і геооб'єктні моделі), створенні мультимедійних ресурсів геоданих і т.ін.

Ще одним із перспективних напрямів розвитку баз даних є розширення можливостей СУБД для роботи з темпоральними даними, зображеннями і мультимедійними даними.

Темпоральні системи дозволяють автоматично зберігати множину об'єктів даних, які залежать від часу; крім того вони забезпечують мовні засоби для знаходження даних, які зв'язані з часом. Для цих систем застосовуються спеціальні часові моделі даних.

Мультимедійні інформаційні системи дозволяють зберігати текст, аудіо-, відеозображення. Використання таких баз даних вимагає підтримки багатомодельної системи, розширення функцій управління звичайної СУБД.

Одним з перспективних напрямів розвитку баз даних є напрям, який пов'язаний з об'єднанням технологій експертних систем і баз даних і розвитком *дедуктивних баз даних*. В цих базах даних нова інформація отримується на основі використання логічних правил виводу і побудови ланцюгів виводу для отримання відповідей на запити.

СПИСОК ЛІТЕРАТУРИ

1. *Гайдамакин Н.А.* Автоматизированные информационные системы, базы и банки данных. Вводный курс.–М.: Гелиос АРВ, 2002. – 368 с.
2. *Гайна Г.А.* Організація баз даних і знань. Мови баз даних: Конспект лекцій.–К.:КНУБА, 2002. – 64 с.
3. *Гайна Г.А., Попович Н.Л.* Організація баз даних і знань. Організація реляційних баз даних: Конспект лекцій.–К.:КНУБА, 2000. – 76 с.
4. *Гарсиа-Молина Г., Ульман Д., Уидом Д.* Системы баз данных.–М.: Издательский дом "Вильямс", 2003. – 1088 с.
5. *Григорьев Ю.А., Ревунков Г.И.* Банки данных.–М.: Изд-во МГТУ им. Н.Э.Баумана, 2002. – 320 с.
6. *Гроff Дж., Вайнберг П.* Энциклопедия SQL.–СПб.: Питер, 2003. – 896 с.
7. *Дейт К.Дж.* Введение в системы баз данных.–К.: Диалектика, 1998. – 784 с.
8. *Диго С.М.* Проектирование и использование баз данных.–М.: Финансы и статистика, 1995. – 208 с.
9. *Карпова Т.С.* Базы данных: модели, разработка, реализация.– СПб.: Питер, 2001. – 304 с.
10. *Когаловский М.Р.* Энциклопедия технологий баз данных.– М.: Финансы и статистика, 2002. – 800 с.
11. *Конноли Т., Бегг К.* Базы данных. Проектирование, реализация и сопровождение. Теория и практика. – М.: Издательский дом "Вильямс", 2003. – 1440 с.
12. *Кренке Д.* Теория и практика построения баз данных.– СПб.: Питер, 2003. – 800 с.
13. *Малыхина М.П.* Базы данных: основы, проектирование, использование. – СПб.: БХВ-Петербург, 2004. – 512 с.
14. *Роб П., Коронел К.* Системы баз данных: проектирование, реализация и управление. – СПб.: БХВ-Петербург, 2004. – 1040 с.

15. *Хомоненко А.Д., Цыганков В.М., Мальцев М.Г.* Базы данных.– СПб.: Корона принт, 2004. – 736 с.

СЛОВНИК ТЕРМІНІВ

Адміністратор бази даних (database administrator) – особа або група осіб, які відповідають за вироблення вимог до бази даних, за її проектування, створення, супроводження та забезпечення необхідного рівня продуктивності системи.

Адміністратор даних (data steward) – особа або група осіб, які відповідають за достовірність і повноту даних, які містяться в базі даних, її узгодженість, а також за дотримання регламенту щодо актуалізації бази даних.

Архітектура системи (System Architecture) – представлення про сукупність функціональних компонентів системи та їх взаємозв'язках.

База даних (Database) – організована згідно з певними правилами і збереженої у пам'яті комп'ютера сукупність даних, яка характеризує актуальний стан деякої предметної області і використовується для задоволення інформаційних потреб користувачів.

Буфер (buffer) – область оперативної пам'яті, яка використовується для тимчасового розміщення оброблюваних даних, які передбачається використовувати при зверненні до СУБД, або планується записати в БД після обробки, і яка призначена для прискорення обміну між зовнішньою і оперативною пам'яттю.

Дані (Data) **–** представлення фактів про предметну область системи баз даних в формі, яка допускає їх зберігання і обробку на комп'ютері.

Дані неструктуровані (Unstructured Data) **–** дані, які не описуються засобами будь-якої типизованої моделі даних, для них не існує схеми, що описує їх структуру і багатьох інших властивостей.

Дані структуровані (Structured Data) **–** дані, які організовані згідно з функціональними можливостями будь-якої типизованої моделі даних, яка має регулярну статичну структуру, яка чітко відповідає заданій схемі.

Домен відношення (Domain of relation) – множина всіх можливих значень певного атрибуту відношення.

Журнал (Log) **–** функціональний компонент СУБД, який забезпечує реєстрацію в процесі функціонування системи бази даних відомостей про виконання транзакцій, про працюючих користувачів, про застосування, що виконуються і т.ін.

Запис (Record**)** **–** структура даних, яка являє собою сукупність взаємозв'язаних елементів даних різних типів.

Запит (Query) – повідомлення кінцевого користувача або застосування, що направляється до СУБД і активізує в системі бази даних дії, які забезпечують вибірку, вставку, вилучення або оновлення вказаних у ньому даних.

Застосування (Application) – програма або комплекс програм, які забезпечують автоматизацію обробки інформації для прикладної задачі.

Застосування бази даних (Database Application) – програма або комплекс програм, які використовують базу даних і забезпечують автоматизацію обробки інформації з деякої предметної області.

Інтеграція даних (Data Integration) **–** формування єдиного інформаційного ресурсу на основі різних, можливо неоднорідних джерел даних, які базуються на різних технологіях, і забезпечення його використання в деяких застосуваннях.

Індекс (index) – допоміжна структура даних в середовищі зберігання бази даних, яка призначена для підвищення продуктивності виконання операцій пошуку даних, що задовольняють заданому критерію, забезпечення обробки даних відповідно до порядку значень ключа або обчислення

різних статистичних характеристик, які залежать від значень ключів.

Інтерфейс прикладного програмування (Application Programming Interface) – сукупність програмних засобів, які забезпечують прикладній програмі на деякій традиційній мові програмування доступ до систем баз даних, які підтримуються в середовищі конкретної СУБД.

Інформаційна система (Information System) – система, яка реалізує автоматизоване збирання, обробку і маніпулювання даними і включає технічні засоби обробки даних, програмне забезпечення і обслуговуючий персонал.

Інформаційна технологія (Information Technology) – комплекс методів, підходів, стандартів й інструментальних засобів, які використовуються для створення, підтримки і застосування комп'ютерних систем будь-якого класу в деякому середовищі функціонування.

Ключ первинний (Primary Key) – атрибут або підмножина атрибутів, які унікальним чином визначають кортеж даного відношення.

Ключ зовнішній (Foreign Key) – сукупність атрибутів відношення, значення яких є одночасно значеннями первинного ключа іншого відношення.

Метадані (Meta Data) – дані про дані, які описують їх склад і структуру, формат представлення, методи доступу і необхідні для цього повноваження користувачів, місце зберігання, їх семантику, володаря і т.ін.

Мова визначення даних (Data Definition Language) – мова декларативного типу, яка призначена для опису структури бази даних, обмежень цілісності, а іноді для специфікації тригерів, процедур, що зберігаються і т.ін.

Мова маніпулювання даними (Data Manipulation Language) – мова декларативного типу, яка призначена для виконання основних операцій по роботі з базою даних: введення, модифікація і вибір даних за запитами.

Модель даних (Data model) – сукупність правил структурування даних в базах даних, допустимих операцій над ними і обмежень цілісності, яким вони повинні задовольняти.

Модель предметної області (Enterprise Model) – опис структури предметної області разом із сукупністю зв'язаних з нею обмежень цілісності.

Процедура, що зберігається (Storage Procedure) – програма або процедура обробки даних, що зберігається і виконується на комп'ютері-сервері.

Сервер (Server) – програма, яка представляє деякі послуги по запитах інших програм, що називаються клієнтами.

Сервер бази даних (Database Server) – програма, яка виконує функції управління і захисту бази даних; якщо виклик функцій виконується на мові *SQL*, то його називають *SQL*-сервером.

Словник даних (Data Dictionary) – програмна система призначена для збереження інформації про структури даних, взаємозв'язках файлів бази даних, типах даних і форматах їх представлення, належності користувачам, кодах захисту, засобах ідентифікації і т.ін.

Система управління базами даних (Data Base Management System) – комплекс мовних і програмних засобів, призначений для створення, ведення і сумісного використання баз даних, забезпечення логічної і фізичної цілісності даних, надійного і ефективного використання ресурсів, представлення санкціонованого доступу для застосувань і кінцевих користувачів, а також підтримки функцій адміністрування бази даних.

Схема бази даних (Database Schema) – опис бази даних засобами мови опису даних будь-якого архітектурного рівня СКБД.

Схема відношення (Relation Schema) – список імен атрибутів відношення.

Сховище даних (Data Warehouse) – система, що містить несуперечливу інтегровану предметно-орієнтовану сукупність історичних даних з метою аналіза даних і прийняття рішень.

Транзакція (Transaction) – сукупність операцій маніпулювання даними в системі баз даних, яка переводить базу даних з одного цілісного стану в інший.

Тригер (Trigger) – різновид процедури, що зберігається, яка автоматично викликається при виникненні певних подій в базі даних.

Файл плоский (Flat File) – файл, записи якого не можуть містити яких-небудь агрегати даних – елементів даних або груп, що повторюються, а також покажчики на будь-які інші записи цього файлу.

Файл-сервер (File Server) – комп'ютер, що забезпечує зберігання файлів і представлення санкціонованого доступу до них користувачів і застосувань в мережі.

Цілісність бази даних (Database Integrity) – властивість бази даних, яка означає, що вона містить повну, несуперечливу і адекватно відображаючу предметну область інформацію.

CASE-технологія (Computer Aided System Engineering і Computer Aided Software Engineering) – методологія проектування інформаційних систем й інструментальні засоби, які дозволяють наочно моделювати предметну область, аналізувати її модель на всіх етапах розробки і супроводження інформаційної системи і розробляти застосування.

CORBA (Common Object Request Broker Architecture – архітектура брокера загальних об'єктних запитів) – стандарт технології для інформаційних систем з розподіленою обробкою групи *OMG (Object Management Group)*.

CGI (Common Gateway Interface – інтерфейс загальний шлюзовий) – стандарт, що визначає взаємодію між *WWW*-сервером і модулями розширення, які можуть застосовуватися

для виконання додаткових функцій, які не підтримуються сервером.

DCOM (Distributed Component Object Model – розподілена компонентна об'єктна модель) – стандарт технології для інформаційних систем з розподіленою обробкою фірми *Microsoft*.

ODBC (Open Database Connectivity – інтерфейс відкритий для підключення до баз даних) – інтерфейс прикладного програмування у вигляді бібліотек функцій, що викликаються з різних програмних середовищ і дозволяють застосуванням уніфікованим чином звертатися на мові *SQL* до баз даних різних форматів.

OLAP (On-Line Analitical Processing – аналітична обробка інформації) – технологія інтерактивної аналітичної обробки даних в системах баз даних, яка призначена для підтримки прийняття рішень і орієнтована головним чином на нерегламентовані інтерактивні запити.

OLE DB (Object Linking and Embedding Database – зв'язування і вбудовування об'єктів баз даних) – стандартний інтерфейс, що являє собою універсальну технологію доступу до будь-яких джерел даних через інтерфейс *COM* (*Common Object Model*).

OLTP (On-Line Transaction Processing – оперативна обробка транзакцій) – клас застосувань систем баз даних, які призначені для підтримки поточної діяльності різного рода організацій.

SQL (Structured Query Language – структурована мова запитів) – мова баз даних, яка являє собою стандартизований засіб опису запитів до баз даних.

***XML (eXtensible Markup Language – розширювана мова розмітки*) –** мова розмітки для документів WEB, яка дозволяє визначати структуру документа і типи даних, які в ньому зберігаються.

Додаток 1

Системи позначень в ER-моделях

Модель П. Чена запропонована П.Ченом 1976 р. застосовує в якості основних функціональних блоків для моделювання сутності і зв'язки. Всі сучасні ER-діаграми базуються на моделі Чена.

Модель "пташина лапка" (Crow's Foot Model) розроблена К.В.Бахманом розглядає сумісно інформацію про зв'язність і потужність зв'язку в єдиному наборі символів. Ця модель відображає лише три значення потужності 0, 1 і N.

Модель IDEF1X розроблена для армії США і широко використовується у державних установах США, фінансових і промислових корпораціях. В цій моделі використовується менше символів ніж в інших моделях, це зменшує деталізацію типів і розширює можливості відображення зв'язків.

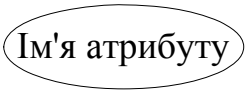
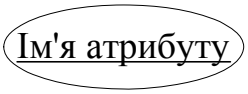
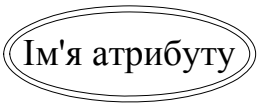
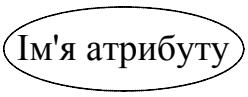
Модель Баркера є подальшим розвитком моделі Чена і дозволяє на верхньому рівні інтегрувати запропоновані Ченом засоби опису моделей. Всі зв'язки в цій моделі є бінарними і дозволяють визначати кількість об'єктів, що беруть участь у зв'язку, визначати ступінь обов'язковості зв'язків. Екземпляри сутностей нащадків можуть існувати тільки при існуванні батьківської сутності. Атрибути можуть бути обов'язковими і необов'язковими. Модель даних підтримує класи і підкласи.

Уніфікована мова моделювання UML (Unified Modeling Language) була створена на основі методів об'єктно-орієнтованого аналізу і проектування, які були розроблені у 1980-1990-х рр. У мові UML використовується ряд діаграм. До основних належать діаграми класів. В даний час мова UML наближається до критеріїв стандартної мови моделювання.

Система позначень, що застосовується в представлених у додатку ER-діаграмах, має довідковий характер й ілюструє можливості моделювання даних в різних методиках проектування баз даних.

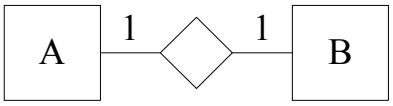
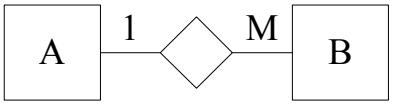
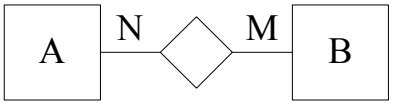
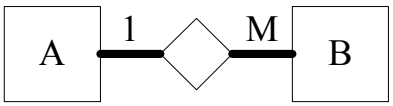
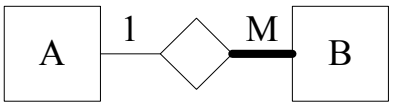
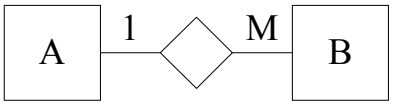
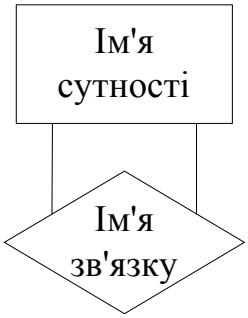
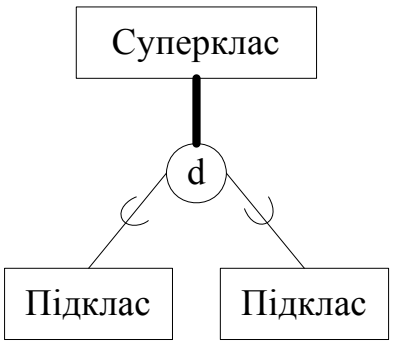
1. Модель Чена

Таблиця Д.1

Графічне представлення	Зміст
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

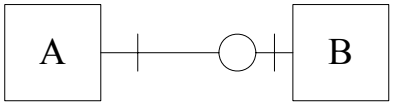
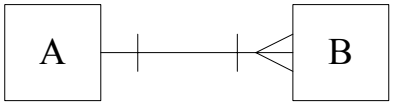
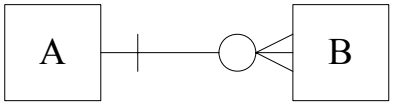
1. Модель Чена

Продовження табл. Д.1

Графічне представлення	Зміст
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 неперетинається , літера o означає, що зв'язок може перетинатися. Подвійна лінія означає обов'язкову участь, а одинарна лінія означає необов'язкову участь

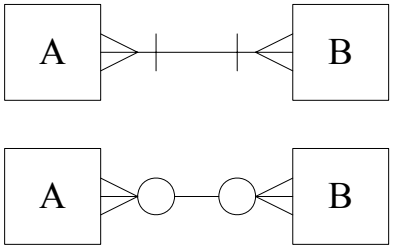
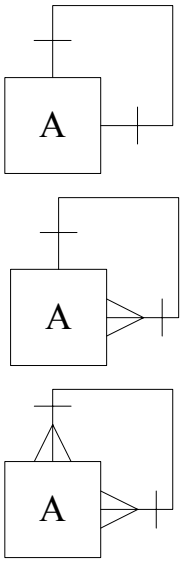
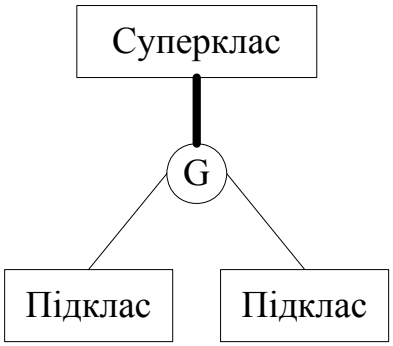
2. Модель "пташина лапка"

Таблиця Д.2

Графічне представлення	Зміст
Сутності 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
Ім'я сутності Ім'я атрибуту 1 Ім'я атрибуту 2 ..... Ім'я атрибут N	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

2. Модель "пташина лапка"

Продовження табл. Д.2

Графічне представлення	Зміст
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 неперетинається , літера Gs означає, що зв'язок може перетинатися. Подвійна лінія означає обов'язкову участь, а одинарна лінія означає необов'язкову участь

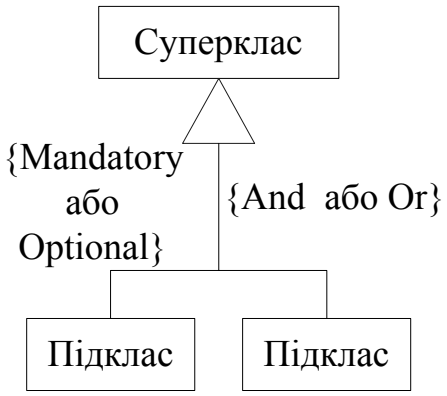
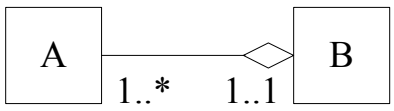
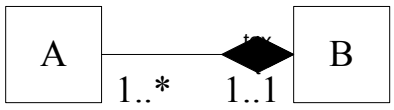
3. Модель UML

Таблиця Д.3

Графічне представлення	Зміст
<u>Сутності</u> 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
<u>Зв'язок</u> 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

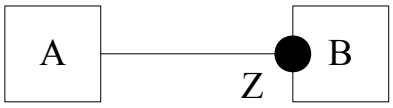
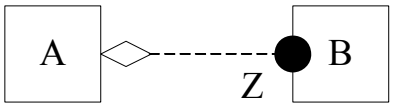
3. Модель UML

Продовження табл. Д.3

Графічне представлення	Зміст
Розширена модель сутність-зв'язок **Суперклас/підклас** 	Уточнення/узагальнення. *Or* означає, що зв'язок неперетинається , а *And* означає, що зв'язок може перетинатися. *Mandatory* означає обов'язкову участь, а *Optional* означає необов'язкову участь
Агрегування 	Агрегування. На сутність "ціле" вказує пустий ромб
Композиція 	Композиція. На сутність "ціле" вказує заповнений ромб

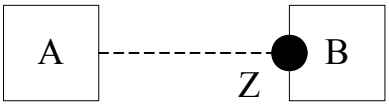
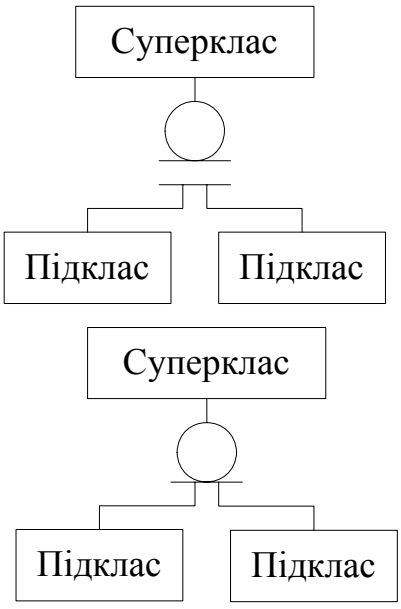
4. Модель IDEF1X

Таблиця Д.4

Графічне представлення	Зміст
Сутності Ім'я сутності Ім'я атрибута Ім'я атрибута 1 Ім'я атрибута 2 ..... Ім'я атрибут N Ім'я сутності Ім'я атрибута Ім'я атрибута 1 Ім'я атрибута 2 ..... Ім'я атрибут 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

4. Модель IDEF1X

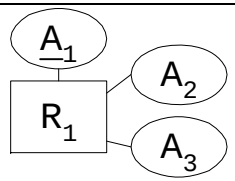
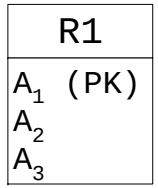
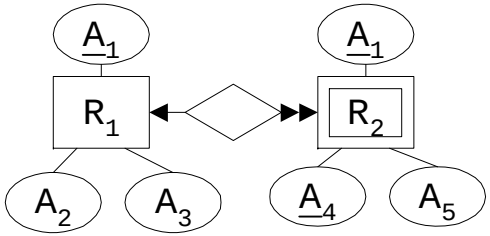
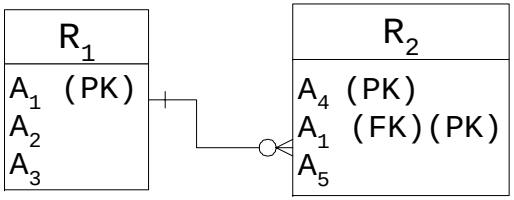
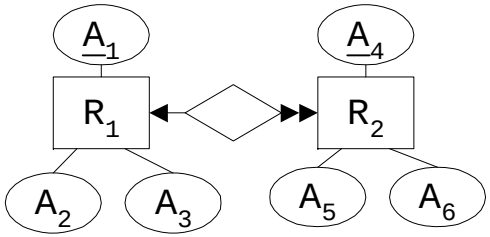
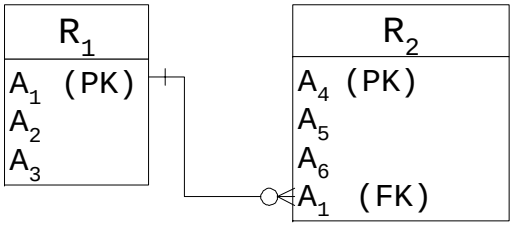
Продовження табл. Д.4

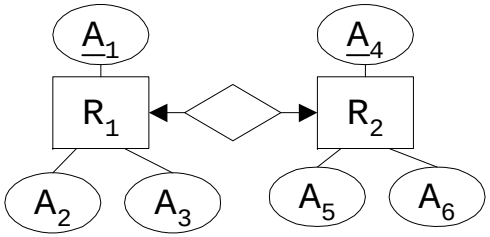
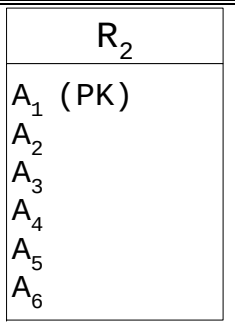
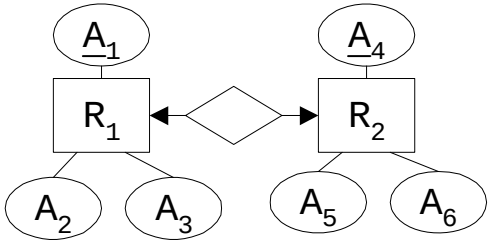
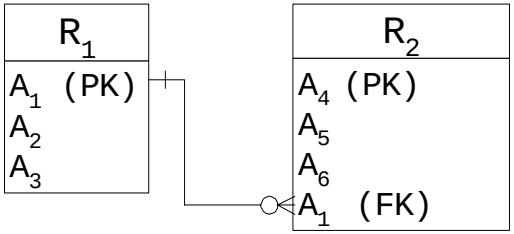
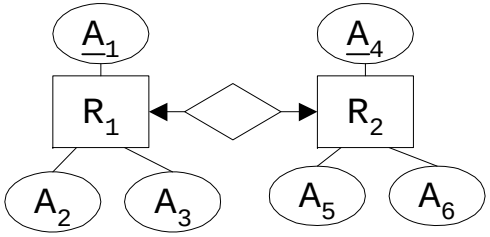
Графічне представлення	Зміст
Зв'язки	
	Зв'язок "один до одного". Запис з A може бути зв'язаний тільки з одним записом з B. Запис з B може бути зв'язаний тільки з одним записом з A, записи в A приймають значення NOT NULL
	Зв'язок "один до багатьох". Запис з A повинен бути зв'язаним з одним або декількома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

Додаток 2

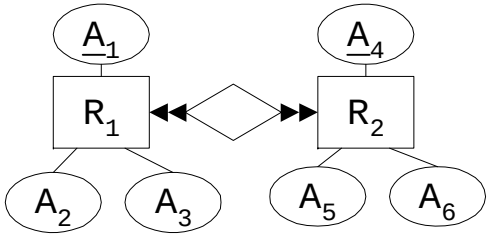
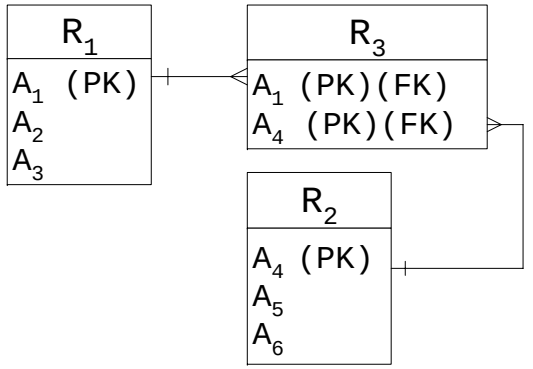
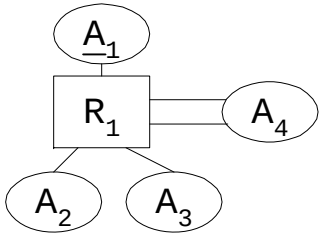
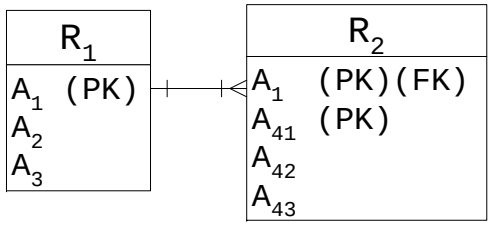
Правила перетворення сутностей, атрибутів і зв'язків у відношення

Таблиця Д.5

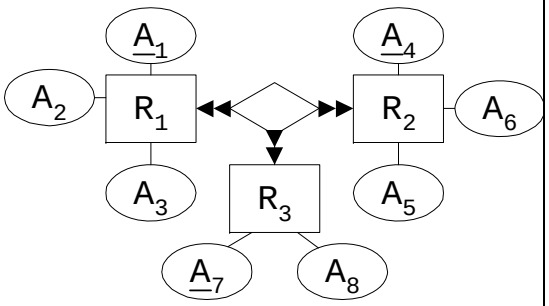
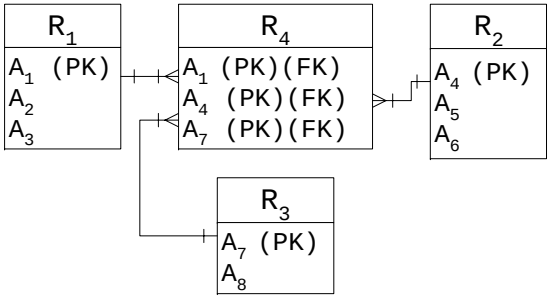
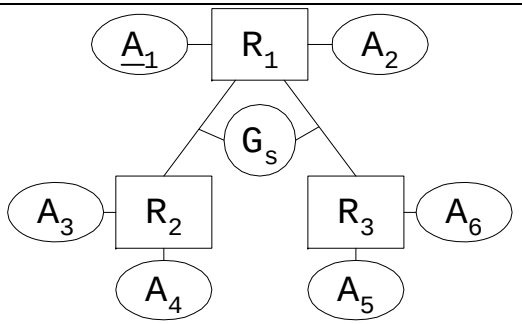
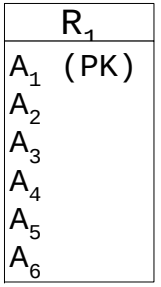
Сутність /зв'язок	Правило перетворення	Приклад	
		ER-модель	**Логічна модель**
Сильна сутність	Створення відношення, яке включає всі прості атрибути		
Слабка сутність	Створення відношення, яке включає всі прості атрибути. Після перетворення зв'язку з сутністю-володарем необхідно визначити первинний ключ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

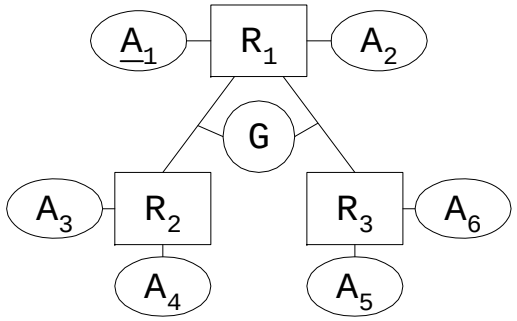
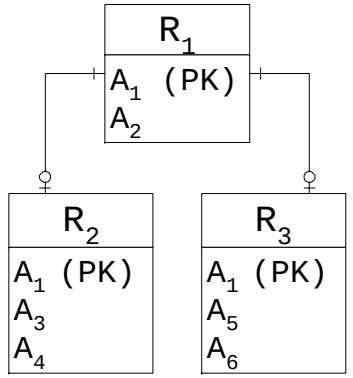
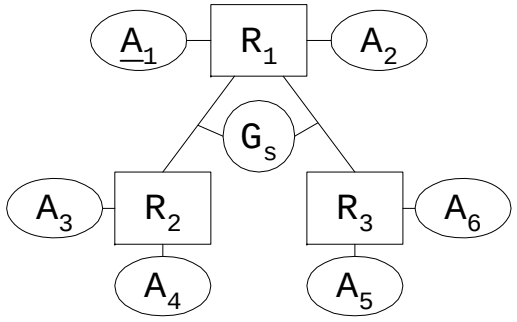
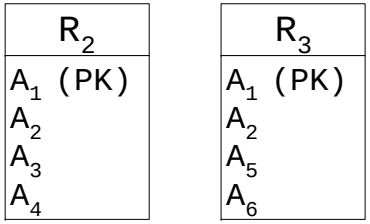
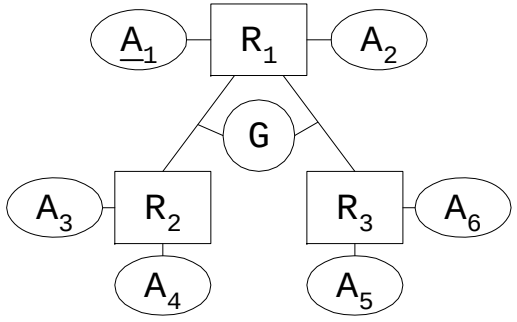
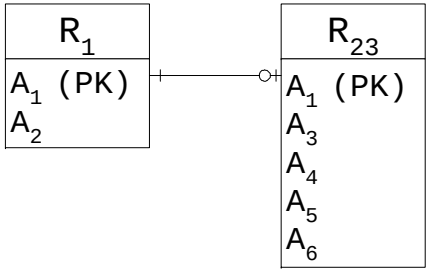
Зв'язок 1:1	Обов'язкова участь обох боків зв'язку. Сутності об'єднуються в одне відношення		
	Обов'язкова участь одного боку зв'язку. Передача первинного ключа на "обов'язковий" бік для використання у якості зовнішнього ключа у відношенні, що представляє "необов'язковий" бік		
	Необов'язкова участь обох боків зв'язку. У разі відсутності додаткової інформації вибір буде довільним.		Можливі три варіанти логчної моделі (з двома відношеннями і з трьома відношеннями)

Продовження дод. 2

Зв'язок N:M	Створення відношення, яке представляє зв'язок, і включення всіх атрибутів зв'язку. Передача в нове відношення копії первинного ключа з кожної сутності-володаря для використання в якості зовнішнього ключа		
Багато-значний атрибут	Створення відношення, яке представляє багатозначний атрибут, і передає в нове відношення копії первинного ключа із сутності-володаря для використання в якості зовнішнього ключа		

Продовження дод. 2

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

Зв'язок клас–підклас	Необов'язкова участь зв'язку суперклас/підклас (Optional), не дозволяється перетин підкласів (Or)		
	Обов'язкова участь зв'язку суперклас/підклас (Mandatory), не дозволяється перетин підкласів (Or)		
	Необов'язкова участь зв'язку суперклас/підклас (Optional), дозволяється перетин підкласів (And)		

ЗМІСТ

ДЛЯ НОТАТОК

Навчальне видання

ГАЙНА Георгій Анатолійович

ОСНОВИ ПРОЕКТУВАННЯ БАЗ ДАНИХ

Навчальний посібник

Редагування та коректура *Н.Б. Науменко*

Комп'ютерна верстка *Т.І. Кукарєвої*

Підписано до друку 2005. Формат 60x84$_{1/16}$
Папір офсетний. Гарнітура Таймс. Друк на різографі.
Ум. друк. арк. 11,86. Обл.-вид. арк. 12,75. Ум. фарбовідб. 103.
Тираж прим. Вид. № 13/І-05. Зам. №

КНУБА, Повітрофлотський проспект, 31, Київ-680, 03680

Віддруковано в редакційно-видавничому відділі
Київського національного університету будівництва і архітектури

Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.