МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
"Харківський авіаційний інститут"

І. В. Шевченко, Ю. С. Манжос

ПРОЕКТУВАННЯ БАЗ ДАНИХ

Навчальний посібник до лабораторного практикуму

Харків "ХАІ" 2018

УДК 004.65.001.63 (076.5)
Ш37

Рецензенти: канд. техн. наук, доц. О. В. Щербаков,
канд. техн. наук Т. В. Філімончук

Шевченко, І. В.
Ш37 Проектування баз даних [Текст] : навч. посіб. до лаб. практикуму / І. В. Шевченко, Ю. С. Манжос. – Харків : Нац. аерокосм. ун-т ім. М. Є. Жуковського "Харків. авіац. ін-т", 2018. – 92 с.

ISBN 978-966-662-611-3

Описано підхід до проектування реляційних баз даних (БД), який ґрунтується на побудові концептуальної моделі предметної області з наступним перетворенням її на реляційну схему (логічну і фізичну моделі даних) за спеціальною методикою. Подано велику кількість прикладів, що полегшують розуміння теоретичного матеріалу.

Наведено лабораторний практикум з проектування БД і варіанти індивідуального завдання з розроблення програмного забезпечення з використанням БД.

Для студентів усіх форм навчання напряму підготовки 121 "Інженерія програмного забезпечення", а також для всіх тих, хто бажає вивчити курс "Бази даних".

Іл. 38. Табл. 5. Бібліогр.: 15 назв

УДК 004.65.001.63 (076.5)

ISBN 978-966-662-611-3

Вступ. ІСТОРІЯ РОЗВИТКУ МОДЕЛЕЙ БАЗ ДАНИХ

У класичній теорії баз даних модель даних − формальна теорія подання й оброблення даних у системі управління базами даних (СУБД), яка складається, принаймні, з трьох аспектів:

1) аспекту структури − методи опису типів і логічних структур даних у базі даних;

2) аспекту маніпуляції − методи маніпулювання даними;

3) аспекту цілісності − методи опису й підтримки цілісності бази даних.

Аспект структури визначає, що логічно являє собою база даних, аспект цілісності − засоби описів коректних станів бази даних, аспект маніпуляції − способи переходу між станами бази даних (тобто способи модифікації даних) і способи отримання даних з бази даних.

Модель даних − це абстрактне, самодостатнє, логічне визначення об'єктів, операторів та інших елементів, які в сукупності становлять абстрактну машину доступу до даних, з якою взаємодіє користувач. Ці об'єкти дають змогу моделювати структуру даних, а оператори − поведінку даних.

Кожна БД і СУБД будується на основі певної явної або неявної моделі даних. Усі СУБД, побудовані на одній і тій самій моделі даних, відносять до одного типу. Наприклад, основою реляційних СУБД є реляційна модель даних, мережних СУБД − мережна модель даних, ієрархічних СУБД − ієрархічна модель даних і т. д.

У літературі [1−3] термін "модель даних" іноді використовується як "модель бази даних" ("схема бази даних"). Однак така підміна понять є неправильною. Модель даних − це теорія або інструмент моделювання, тоді як модель бази даних (схема бази даних) є результатом моделювання. Співвідношення між цими поняттями є аналогічним співвідношенню між мовою програмування й конкретною програмою, написаною цією мовою.

Наведемо короткий історичний огляд найбільш поширених моделей БД: *ієрархічної, мережної, реляційної та об'єктно-орієнтованої.*

Ієрархічна модель являє собою деревоподібну структуру, тобто кожний запис пов'язаний тільки з одним записом, що знаходиться на більш високому рівні.

Така система добре ілюструється системою класифікації рослин і тварин. Прикладом може також бути структура зберігання інформації на дисках ПК. Головне поняття такої моделі − рівень. Кількість рівнів і їх склад залежать від прийнятої при створенні БД класифікації. Доступ до будь-якого з цих записів здійснюється шляхом проходу по суворо визначеному ланцюжку вузлів. При такій структурі легко здійснювати пошук потрібних даних, але якщо спочатку опис є неповним або не передбачено будь-який критерій пошуку, то він стає неможливим. Для досить простих завдань така

система є ефективною, але вона майже непридатна для використання в складних системах з оперативним обробленням запитів.

Розглянемо приклад ієрархічної моделі даних підприємства з роботи [4]. Підприємство складається з відділів, в яких працюють робітники. У кожному відділі можуть працювати кілька робітників, але робітник не може працювати більш ніж в одному відділі.

Тому для інформаційної системи управління персоналом необхідно створити групове відношення, що складається з батьківського запису ВІДДІЛ (назва_відділу, кількість_робітників) і дочірнього запису РОБІТНИК (прізвище, посада, оклад). Це відношення показано на рис. В.1, а.

Для автоматизації обліку контрактів із замовниками слід створити ще одну ієрархічну структуру: замовник → контракти з ним → робітники, які беруть участь у роботі над контрактом. Це дерево буде містити записи ЗАМОВНИК (ім'я_замовника, адреса), КОНТРАКТ (номер, дата, сума), ВИКОНАВЕЦЬ (прізвище, посада, назва_відділу) (рис. В.1, б).

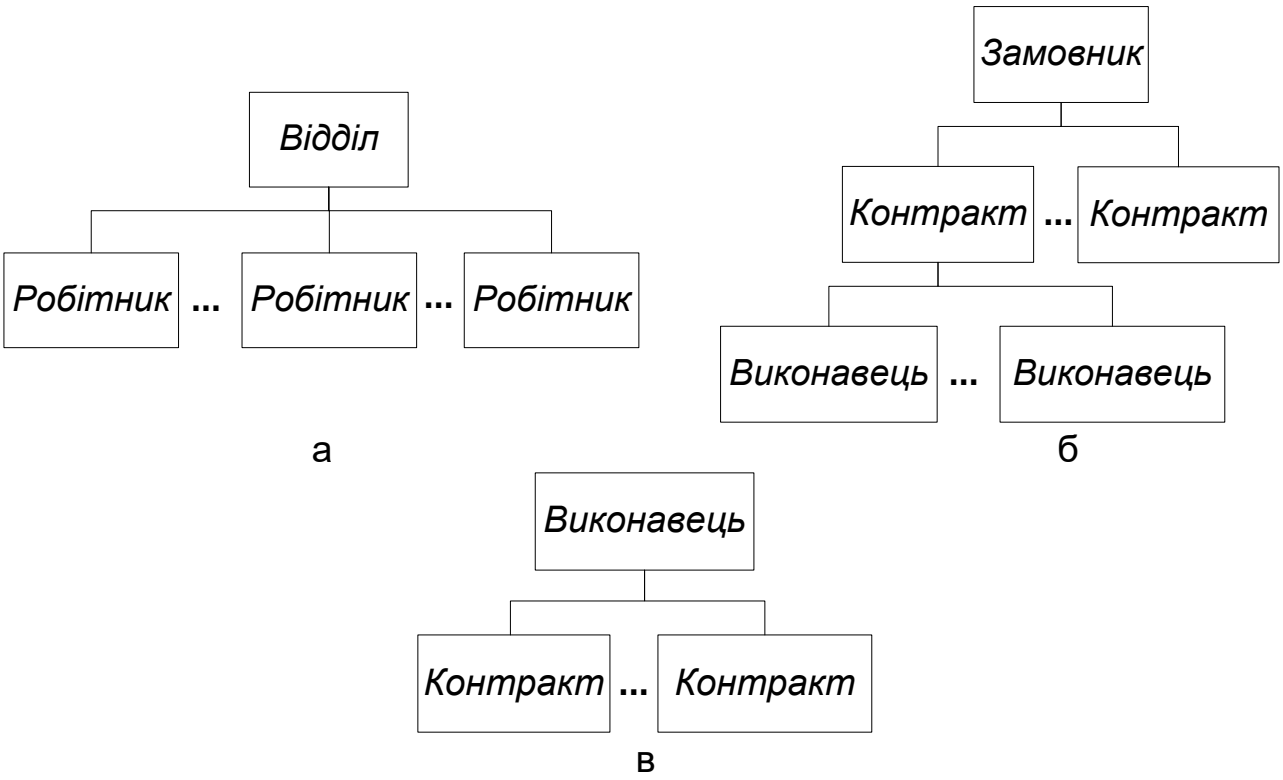


Рис. В.1. Ієрархічна модель інформаційної системи управління персоналом підприємства

Із цього прикладу видно недоліки ієрархічних баз даних. Частково дублюється інформація між записами РОБІТНИК і ВИКОНАВЕЦЬ (такі записи називають парними), причому в ієрархічній моделі даних не передбачено підтримку відповідності між парними записами.

Ієрархічна модель реалізує відношення між батьківським і дочірнім записами за схемою 1:N, тобто одному батьківському запису може відповідати будь-яка кількість дочірніх. Припустимо тепер, що виконавець

може брати участь більш ніж в одному контракті (тобто виникає зв'язок типу M:N). У цьому випадку в базу даних необхідно ввести ще одне групове відношення, в якому ВИКОНАВЕЦЬ буде батьківським записом, а КОНТРАКТ − дочірнім (рис. В.1, в). Таким чином, знову дублюється інформація.

Однією з найбільш популярних ієрархічних СУБД була Information Management System (IMS) компанії IBM, яка з'явилася в 1968 році.

Мережна модель повинна була усунути деякі з недоліків ієрархічних моделей. Якщо структура даних виявлялася складнішою, ніж звичайна ієрархія, простота структури ієрархічної бази даних ставала її недоліком. Наприклад, у базі даних для зберігання замовлень одне замовлення могло брати участь у трьох різних відносинах предок/нащадок, що зв'язують замовлення з клієнтом, який розмістив його, зі службовцем, який прийняв його, і з замовленим товаром. Такі структури даних не відповідали суворій ієрархії.

У зв'язку з цим було розроблено нову мережну модель даних, в якій один запис міг брати участь у декількох відносинах предок/нащадок, як показано на рис. В.2. У мережній моделі такі відносини називалися множинами.

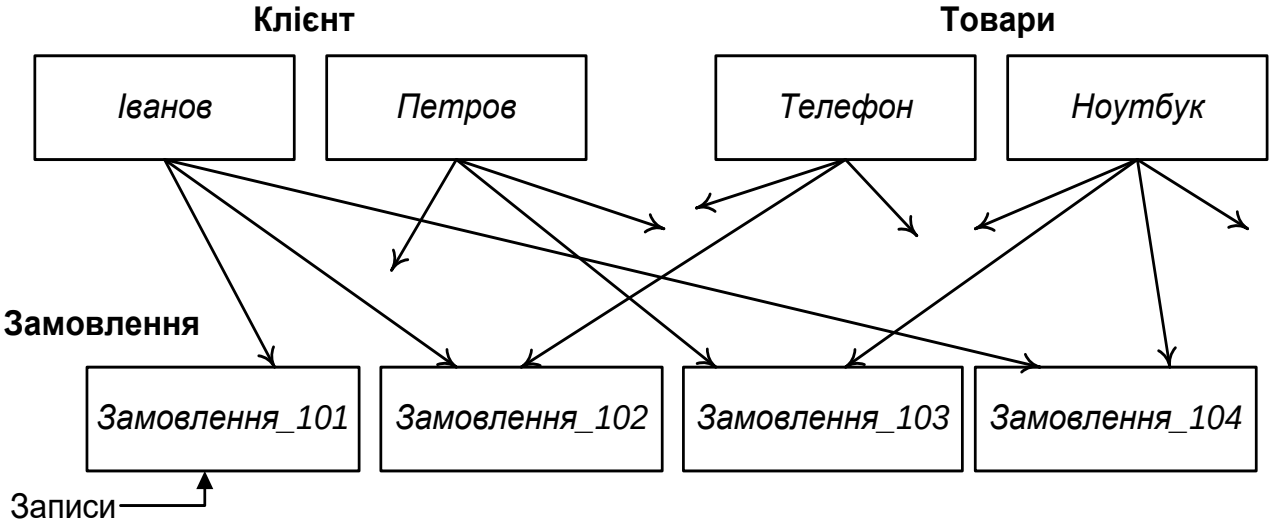


Рис. В.2. Мережна база даних, яка містить інформацію про замовлення

У 1971 році на конференції з мов систем даних було опубліковано офіційний стандарт мережних баз даних, відомий як модель CODASYL. Компанія IBM не стала розробляти власну мережну СУБД і замість цього продовжувала нарощувати можливості IMS. Але у 70-х роках незалежні виробники програмного забезпечення реалізували мережну модель у таких продуктах, як IDMS компанії Cullinet, Total компанії Cincom і СУБД Adabas, які набули великої популярності.

Однак мережні бази даних, як і ієрархічні, були дуже жорсткими. Набори відносин і структуру записів доводилося визначати заздалегідь. Така модель давала змогу прискорити доступ до даних, але зміна структури бази потребувала значних зусиль і часу, а іноді необхідно було перебудовувати всю базу даних.

Як ієрархічна, так і мережна бази даних були інструментами програмістів. Щоб отримати відповідь на запитання типу "Який товар найбільш часто замовляє компанія А?", програмісту доводилося писати програму для навігації по базі даних. Реалізація призначених для користувача запитів часто затягувалася на тижні й навіть місяці.

Реляційна модель повинна була усунути недоліки ієрархічної й мережної моделей. Ця модель була створена Едгаром Коддом у 1970 році й викликала загальний інтерес. Реляційна модель була спробою спростити структуру бази даних. У ній не було явних покажчиків на предків і нащадків, а всі дані подавалися у вигляді простих таблиць, розбитих на рядки й стовпці. На рис. В.3 показано реляційну базу даних, яка містить інформацію про замовлення.

Таблиця **"КЛІЄНТИ"**

Код_клієнта	***Прізвище_ім'я***	***Місто***	***Телефон***
1	Іванов Ілля	Харків	+380501111111
2	Петров Микита	Київ	+380962222222
3	Попов Сергій	Харків	+380633333333
4	Смірнова Ірина	Дніпро	+380954444444

Таблиця **"ТОВАРИ"**

Код_товару	***Назва***	***Ціна***	***Кількість***
1562	Samsung Galaxy J7	4999	15
0267	Nokia 5 Dual Sim	5999	17
0312	Samsung Galaxy J1	2099	20

Таблиця **"ЗАМОВЛЕННЯ"**

Код_замовл	***Дата***	***Код_клієнта***	***Код_товару***	***Кількість***
101	3.10.2017	2	0312	1
102	5.10.2017	1	1562	1
103	5.10.2017	3	0312	2

Рис. В.3. Реляційна база даних, яка містить інформацію щодо замовлень в Інтернет-магазині

Е. Кодд у 1985 році написав статтю, де сформулював 12 правил, яким має задовольняти будь-яка база даних, що претендує на звання

реляційної. Відтоді дванадцять правил Кодда вважаються визначенням реляційної СУБД. Однак можна сформулювати й більш просте визначення.

Реляційною називається база даних, в якій усі доступні користувачеві дані організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями.

Наведене визначення не залишає місця вбудованим покажчикам, які є в наявності в ієрархічних і мережних СУБД. Незважаючи на це, реляційна СУБД також здатна реалізувати відносини предок/нащадок, проте ці відносини подані виключно значеннями даних, які містяться в таблицях.

В **об'єктно-орієнтованій моделі** дані й методи, що їх обробляють, об'єднуються в структури, які називаються об'єктами. Типи об'єктів називаються класами. Розглянемо особливості об'єктно-орієнтованої моделі даних [5].

Складні об'єкти будуються з простіших за допомогою *конструкторів.* Найпростішими об'єктами є числа, символи, символьні рядки довільної довжини, булеві змінні тощо. Будь-який конструктор має бути застосовним до будь-якого об'єкта (наприклад, повинна надаватися можливість побудови множини з масивів або масиву з множин). Маніпулювання складними об'єктами забезпечується відповідними операціями, які часто поширюються на всі компоненти таких об'єктів. Прикладом може бути вибір чи видалення складного об'єкта або створення його копії. Існує можливість визначати додаткові операції над складними об'єктами.

Кожний об'єкт є унікальним, тобто забезпечується унікальна *ідентифікація об'єктів* (для мов програмування унікальними ідентифікаторами можуть бути адреси пам'яті, за якими зберігаються об'єкти). Кожний об'єкт має свій *стан* – це поточне значення, яке приписане об'єкту. Об'єкт може мати єдиний стан протягом свого життєвого циклу або переходити з одного стану в інший. Оскільки об'єкти мають властивість *інкапсуляції* (буде розглянуто нижче), то стан об'єкта є *абстракцією*, яка визначається лише через його поведінку (методи). *Унікальність об'єкта* не залежить від його стану.

У моделі з об'єктами, що ідентифікуються, два об'єкти можуть спільно використовувати компоненти (зокрема інші об'єкти). Отже, схематичним відображенням складного об'єкта є граф, а у системі без ідентифікованості об'єктів – це дерево.

Поведінка об'єкта – це сукупність операцій (методів), які він надає. Лише через ці операції розкривається семантика об'єкта.

Класи об'єктів – спосіб реалізації множини об'єктів, встановлення їхньої структури, поведінки й інтерфейсу, тобто спосіб запам'ятовування інформації про їхні стани. Проте власне стан має запам'ятовувати сам об'єкт. Однією з основних властивостей класу, а отже, і його об'єктів є інкапсуляція.

Інкапсуляція потребує, щоб дані й програмні коди для маніпулювання даними були прихованими. З цієї точки зору об’єкт складається з інтерфейсної й реалізаційної частин. *Інтерфейсна частина* є специфікацією набору операцій, допустимих над об’єктом. Лише ця частина об’єкта є видимою для методів інших об’єктів. *Реалізаційна частина* складається з даних, що описують стан об’єкта, і процедур, які реалізують операції над об’єктом. Інкапсуляція специфікується на рівні оголошення класу.

Успадкування є механізмом створення нових класів з використанням даних і методів інших класів. Це дає можливість деякі властивості, спільні для багатьох класів, описувати в базовому класі.

Принцип поліморфізму є розширенням принципу успадкування й дає змогу перевизначити методи в успадкованих класах.

Основна перевага об’єктно-орієнтованої моделі даних – відсутність проблеми невідповідності моделі даних у програмі й БД.

Однак застосування СУБД, які базуються на об’єктно-орієнтованій моделі даних, обмежується відсутністю загальної моделі даних. Навіть за наявності досить великої кількості продуктів, що реалізують об'єктно-орієнтовану модель даних, залишається багато невирішених питань. Проте цей напрямок є перспективним і прогресивним.

NoSQL модель даних. Сьогодні термін "NoSQL" не має загальновизнаного визначення і, скоріше, характеризує вектор розвитку інформаційних технологій зберігання й оброблення величезних обсягів даних, спрямований у бік від реляційних баз даних – "Not Only SQL".

Структура даних у NoSQL-базах є нерегламентованою, наприклад, в окремому рядку або документі можна додати довільне поле без попередньої декларативної зміни структури всієї таблиці. Позитивний ефект, що одержується через відсутність структурованої схеми даних, – ефективна робота з розрідженими даними. Неструктурована схема даних має і недоліки: відсутність всіляких обмежень з боку бази даних, а також додаткові складності в розумінні й контролі структури даних при паралельній роботі з базою різних проектів.

На відміну від реляційної моделі, яка зберігає логічну бізнес-сутність програмної системи в різних фізичних таблицях для забезпечення нормалізації, NoSQL-бази даних оперують з цими сутностями як з цілісними об'єктами (агрегатами). При цьому демонструється головне правило проектування структури даних у NoSQL-базах – підпорядкування вимогам програми й максимальна оптимізація під найбільш часті запити.

1. ОСНОВНІ ПОНЯТТЯ РЕЛЯЦІЙНОЇ МОДЕЛІ БАЗ ДАНИХ

Реляційний підхід до організації баз даних був закладений наприкінці 60-х років Едгаром Коддом. В останні десятиліття цей підхід є найбільш поширеним.

Перевагами реляційного підходу прийнято вважати такі властивості:

- реляційний підхід ґрунтується на невеликій кількості інтуїтивно зрозумілих абстракцій, на основі яких є можливим просте моделювання найбільш поширених предметних областей;
- ці абстракції можуть бути точно й формально визначеними;
- теоретичним базисом реляційного підходу до організації баз даних є простий і потужний математичний апарат теорії множин і математичної логіки;
- реляційний підхід забезпечує можливість ненавігаційного маніпулювання даними без необхідності знання конкретної фізичної організації баз даних у зовнішній пам'яті.

Комп'ютерний світ далеко не відразу визнав реляційні системи. У 70-ті роки минулого століття, коли вже було отримано майже всі основні теоретичні результати і навіть існували перші прототипи реляційних СУБД, деякі авторитетні фахівці заперечували можливість добитися ефективної реалізації таких систем. Однак переваги реляційного підходу й розвиток методів і алгоритмів організації й управління базами даних привели до того, що до кінця 80-х років реляційні системи зайняли на світовому ринку СУБД домінуючі позиції.

Розглянемо реляційний підхід до подання даних докладніше [1–4].

Наприкінці 60-х років з'явилися роботи, в яких обговорювалися можливості застосування різних табличних даталогічних моделей даних, тобто можливості використання звичних і природних способів подання даних. Найбільш значною з них була стаття співробітника фірми IBM доктора Е. Кодда, де, ймовірно, уперше було застосовано термін "реляційна модель даних". Будучи математиком за освітою, Е. Кодд запропонував використовувати для оброблення даних апарат теорії множин (об'єднання, перетинання, різниця, декартовий добуток). Він показав, що будь-яке подання даних зводиться до сукупності двовимірних таблиць особливого виду, відомого в математиці як відношення (relation).

1.1. Таблиці

У реляційній базі даних інформація організована у вигляді таблиць, розділених на рядки і стовпці, на перетині яких містяться значення даних. Кожна таблиця має унікальне ім'я, яке описує її зміст. Більш наочно структуру таблиці ілюструє рис. 1.1, на якому зображено таблицю СТУДЕНТ. Кожний горизонтальний рядок цієї таблиці містить дані про окрему фізичну сутність – одного студента.

Таблиця **"Студент"**

Номер залікової книжки	***Прізвище***	***Ім'я***	***По батькові***	***Дата народження***	***Номер групи***	***Середній бал***
101/17	Іванов	Іван	Іванович	1.03.2000	131-1	85
102/17	Петров	Петро	Петрович	15.01.1999	131-2	80
103/17	Олексіїв	Олексій	Олексійович	25.02.2000	131-1	75
...	...	...	...	...	...	...

Рис. 1.1. Структура реляційної таблиці

Кожен вертикальний стовпець таблиці СТУДЕНТ містить один елемент даних для кожного студента: номер залікової книжки, прізвище, ім'я, по батькові, дату народження, номер групи, середній бал.

На перетині кожного рядка з кожним стовпцем таблиці міститься тільки одне значення даних. Наприклад, розглянемо рядок, в якому наведено дані про студента Петрова: у стовпці НОМЕР ГРУПИ міститься значення "131-2", у стовпці СЕРЕДНІЙ БАЛ того ж рядка – значення 80, яке є загальним середнім балом екзаменаційної успішності цього студента за 100-бальною системою оцінювання.

Усі значення, які містяться в одному й тому ж стовпці, є даними одного типу. Наприклад, у стовпцях НОМЕР ЗАЛІКОВОЇ КНИЖКИ, ПРІЗВИЩЕ, ІМ'Я, ПО БАТЬКОВІ містяться тільки рядки, у стовпці ДАТА НАРОДЖЕННЯ – дати, в стовпці НОМЕР ГРУПИ – рядки, а в стовпці СЕРЕДНІЙ БАЛ – цілі числа.

Кожний стовпець в таблиці має своє ім'я, яке зазвичай є заголовком стовпця. Усі стовпці в одній таблиці повинні мати унікальні імена, однак дозволяється присвоювати однакові імена стовпцям, розташованим у різних таблицях. На практиці такі імена стовпців, як НАЗВА, АДРЕСА, КІЛЬКІСТЬ, ЦІНА, часто зустрічаються у різних таблицях однієї бази даних.

Стовпці таблиці впорядковано зліва направо, і їх порядок визначається при створенні таблиці. У будь-якій таблиці завжди є як мінімум один стовпець. У стандарті ANSI/ISO не вказується максимально допустима кількість стовпців у таблиці, проте майже у всіх комерційних СУБД ця межа існує і зазвичай становить приблизно 255 стовпців.

На відміну від стовпців рядки таблиці не мають певного порядку. Це означає, що якщо послідовно виконати два однакових запити для відображення вмісту таблиці, то немає гарантії, що обидва рази рядки будуть перераховані в одному й тому ж порядку.

У таблиці може міститися будь-яка кількість рядків. Цілком припустимо існування таблиці з нульовою кількістю рядків. Така таблиця називається порожньою. Порожня таблиця зберігає структуру, визначену її стовпцями (просто в ній не містяться дані). Стандарт ANSI/ISO не накладає обмежень на кількість рядків у таблиці, і в багатьох СУБД розмір

таблиць обмежено лише вільним дисковим простором комп'ютера. В інших СУБД є максимальна межа, однак вона досить висока – близько двох мільярдів рядків, а іноді й більше.

1.2. Первинні ключі

Оскільки рядки в реляційній таблиці не впорядковано, не можна вибрати рядок за його номером у таблиці. У таблиці немає "першого", "останнього" або "тринадцятого" рядка. Тоді яким же чином можна вказати в таблиці на конкретний рядок, наприклад, рядок для студента з прізвищем Петров?

У правильно побудованій реляційній базі даних у кожній таблиці є один або декілька стовпців, значення в яких у всіх рядках різні. Цей стовпець (стовпці) називається **первинним ключем** таблиці. Якщо подивитися на базу даних, показану на рис. 1.2, то первинним ключем таблиці СТУДЕНТ однозначно є стовпець "Номер залікової книжки". Усі значення в цьому стовпці є різними, тому що не має двох студентів з однаковими номерами залікових книжок.

Таблиця **"Студент"**

Первинний ключ

Номер залікової книжки	***Прізвище***	***Ім'я***	***По батькові***	***Дата народження***	***Номер групи***	***Середній бал***
101/17	Іванов	Іван	Іванович	1.03.2000	131-1	85
102/17	Петров	Петро	Петрович	15.01.1999	131-2	80
103/17	Олексіїв	Олексій	Олексійович	25.02.2000	131-1	75

Рис. 1.2. Структура таблиці "Студент" із зазначенням первинного ключа

Розглянемо таблицю ДИСЦИПЛИНА (рис. 1.3), яка містить такі стовпці: ідентифікатор дисципліни, назва дисципліни, номер семестру, вид контролю. Первинним ключем цієї таблиці є стовпець "Ідентифікатор дисципліни", в якому всі значення різні.

Розглянемо ще одну таблицю ЗАЛІКОВА КНИЖКА (рис. 1.4), яка містить такі стовпці: номер залікової книжки, ідентифікатор дисципліни, оцінка, дата складання, прізвище викладача. Через те, що один і той самий студент склав залік або екзамен не з однієї дисципліни (тобто значення в стовпці "Номер залікової книжки" можуть повторюватися), а екзамен або залік з однієї дисципліни склав не один студент (тобто значення в стовпці "Ідентифікатор дисципліни" можуть повторюватися), то первинним ключем цієї таблиці не може бути один стовпець – ні номер залікової книжки, ні ідентифікатор дисципліни. В цій ситуації тільки комбінація цих двох

стовпців може бути первинним ключем таблиці ЗАЛІКОВА КНИЖКА. Такий первинний ключ називається складеним.

Таблиця **"Дисципліна"**

Первинний ключ

Ідентифікатор дисципліни	*Назва дисципліни*	*Номер семестру*	*Вид контролю*
1	Основи програмування	1	Залік
2	Основи програмування	2	Екзамен
3	Філософія	2	Залік
4	Бази даних	3	Екзамен
5	Структури даних	3	Залік

Рис. 1.3. Таблиця "Дисципліна" із зазначенням первинного ключа

Таблиця **"Залікова книжка"**

Первинний ключ (складений)

Номер залікової книжки	*Ідентифікатор дисципліни*	*Оцінка*	*Дата складання*	*Прізвище викладача*
101/17	1	90	5.12.2017	Смірнова Ю. Л.
102/17	1	78	5.12.2017	Смірнова Ю. Л.
103/17	1	93	5.12.2017	Смірнова Ю. Л.
101/17	2	87	9.12.2017	Попов О. І.
102/17	2	75	9.12.2017	Попов О. І.
103/17	3	95	12.12.2017	Кузнєцова О. М.
101/17	4	91	15.12.2017	Волков І. І.
102/17	4	77	15.12.2017	Волков І. І.
103/17	5	93	17.12.2017	Федоров О. В.

Рис. 1.4 Таблиця "Залікова книжка" із складеним первинним ключем

Первинний ключ для кожного рядка таблиці є унікальним, тому в таблиці з первинним ключем немає двох абсолютно однакових рядків. Таблиця, в якій всі рядки відрізняються один від одного, у математичних термінах називається відношенням. Саме цьому терміну реляційні бази даних і зобов'язані своєю назвою, оскільки їх основою є відношення (таблиці з рядками, що відрізняються один від одного).

Хоча первинні ключі є важливою частиною реляційної моделі даних, у перших реляційних СУБД не було забезпечено явним чином їх підтримку. Зазвичай проектувальники бази даних самі стежили за тим, щоб всі таблиці мали первинні ключі, однак у самих СУБД не було можливості визначити для таблиці первинний ключ. І тільки в СУБД DB2 Version 2, що з'явилася в квітні 1988 року, компанія IBM реалізувала підтримку первинних ключів. Після цього подібну підтримку було додано в стандарт ANSI/ISO.

1.3. Відношення предок/нащадок

Одна з відмінностей реляційної моделі від перших моделей подання даних – відсутність явних покажчиків, які використовуються для реалізації відношень предок/нащадок в ієрархічній моделі даних. Однак цілком очевидно, що відношення предок/нащадок існують і в реляційних базах даних.

У таблиці ЗАЛІКОВА КНИЖКА кожному значенню із стовпця "Номер залікової книжки" можна поставити у відповідність значення із стовпця "Номер залікової книжки" таблиці СТУДЕНТ (рис. 1.5). Іншими словами, доменом цього стовпця (множиною значень, які можуть в ньому зберігатися) є безліч номерів залікових книжок, що містяться в стовпці "Номер залікової книжки" таблиці СТУДЕНТ. Кажуть, що між рядками таблиці ЗАЛІКОВА КНИЖКА і таблиці СТУДЕНТ існує відношення.

Чи не призводить відсутність явних покажчиків у реляційній моделі до втрати інформації? Для того, щоб знайти всі оцінки студента Петрова, необхідно знайти номер його залікової книжки в таблиці СТУДЕНТ і відібрати всі рядки для цієї залікової книжки із таблиці ЗАЛІКОВА КНИЖКА.

Відношення предок/нащадок, що існує між таблицями СТУДЕНТ і ЗАЛІКОВА КНИЖКА, у реляційній моделі не втрачено; просто воно реалізовано у вигляді однакових значень даних, що зберігаються в двох таблицях, а не у вигляді явного покажчика. Усі відношення, що існують між таблицями реляційної бази даних, реалізуються так само.

Таблиця **"Студент"**

Номер залікової книжки	***Прізвище***	***Ім'я***	***По батькові***	***Дата народження***	***Номер групи***	***Середній бал***
101/17	Іванов	Іван	Іванович	1.03.2000	131-1	85
102/17	Петров	Петро	Петрович	15.01.1999	131-2	80
103/17	Олексіїв	Олексій	Олексійович	25.02.2000	131-1	75

Таблиця **"Залікова книжка"**

Номер залікової книжки	***Ідентифікатор дисципліни***	***Оцінка***	***Дата складання***	***Прізвище викладача***
101/17	1	90	5.12.2017	Смірнова Ю.Л.
102/17	1	78	5.12.2017	Смірнова Ю.Л.
103/17	1	93	5.12.2017	Смірнова Ю.Л.
101/17	2	87	9.12.2017	Попов О.І.
102/17	2	75	9.12.2017	Попов О.І.
103/17	3	95	12.12.2017	Кузнєцова О.М.
101/17	4	91	15.12.2017	Волков І.І.
102/17	4	77	15.12.2017	Волков І.І.
103/17	5	93	17.12.2017	Федоров О.В.

Рис. 1.5. Відношення предок/нащадок у реляційній базі даних

1.4. Зовнішні ключі

Стовпець однієї таблиці, значення в якому збігаються зі значеннями стовпця, що є первинним ключем іншої таблиці, називається **зовнішнім ключем.**

На рис. 1.5 стовпець "Номер залікової книжки" є зовнішнім ключем таблиці ЗАЛІКОВА КНИЖКА. Значення цього стовпця відповідають значенням у стовпці "Номер залікової книжки", який є первинним ключем таблиці СТУДЕНТ.

У сукупності первинний і зовнішній ключі створюють між таблицями, в яких вони містяться, таке ж відношення предок/нащадок, як і в ієрархічній базі даних.

Зовнішній ключ, як і первинний, теж може являти собою комбінацію стовпців. На практиці зовнішній ключ завжди буде складеним (що складається з декількох стовпців), якщо він посилається на складений первинний ключ в іншій таблиці. Очевидно, що кількість стовпців і їх типи даних у первинному й зовнішньому ключах збігаються.

Якщо таблиця зв'язана з декількома іншими таблицями, вона може мати декілька зовнішніх ключів. Насправді, таблиця ЗАЛІКОВА КНИЖКА зв'язана не тільки з таблицею СТУДЕНТ, а й з таблицею ДИСЦИПЛІНА, тому що стовпець "Ідентифікатор дисципліни" в таблиці ЗАЛІКОВА КНИЖКА повинен містити тільки ті дані, які є у стовпці "Ідентифікатор дисципліни" таблиці ДИСЦИПЛІНА (рис. 1.6), тобто між таблицями ЗАЛІКОВА КНИЖКА і ДИСЦИПЛІНА також є відношення предок/нащадок.

Таким чином, таблиця ЗАЛІКОВА КНИЖКА має два зовнішніх ключа (див. рис. 1.6):

– стовпець "Номер залікової книжки", який посилається на таблицю СТУДЕНТ і зв'язує кожну оцінку з відповідним студентом;

– стовпець "Ідентифікатор дисципліни", який посилається на таблицю ДИСЦИПЛІНА і зв'язує кожну оцінку з відповідною дисципліною.

Відношення предок/нащадок, створені за допомогою двох зовнішніх ключів у таблиці ЗАЛІКОВА КНИЖКА, можуть здатися знайомими. І дійсно, це ті ж самі відношення, що і в мережній базі даних, зображеній на рис. В.2. Як показує приклад, реляційна модель даних має всі можливості мережної моделі щодо вираження складних відношень.

Зовнішні ключі є невід'ємною частиною реляційної моделі, оскільки реалізують відношення між таблицями бази даних.

Таблиця **"Студент"**

Номер залікової книжки	***Прізвище***	***Ім'я***	***По батькові***	***Дата народження***	***Номер групи***	***Середній бал***
101/17	Іванов	Іван	Іванович	1.03.2000	131-1	85
102/17	Петров	Петро	Петрович	15.01.1999	131-2	80
103/17	Олексіїв	Олексій	Олексійович	25.02.2000	131-1	75

Таблиця **"Дисципліна"**

Ідентифікатор дисципліни	***Назва дисципліни***	***Номер семестру***	***Вид контролю***
1	Основи програмування	1	Залік
2	Основи програмування	2	Екзамен
3	Філософія	2	Залік
4	Бази даних	3	Екзамен
5	Структури даних	3	Залік

Таблиця **"Залікова книжка"**

Номер залікової книжки	***Ідентифікатор дисципліни***	***Оцінка***	***Дата складання***	***Прізвище викладача***
101/17	1	90	5.12.2017	Смірнова Ю. Л.
102/17	1	78	5.12.2017	Смірнова Ю. Л.
103/17	1	93	5.12.2017	Смірнова Ю. Л.
101/17	2	87	9.12.2017	Попов О. І.
102/17	2	75	9.12.2017	Попов О. І.
103/17	3	95	12.12.2017	Кузнєцова О. М.
101/17	4	91	15.12.2017	Волков І. І.
102/17	4	77	15.12.2017	Волков І. І.
103/17	5	93	17.12.2017	Федоров О. В.

Рис. 1.6. Множинні відносини предок/нащадок у реляційній базі даних

2. ВВЕДЕННЯ В ПРОЕКТУВАННЯ РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ НА ОСНОВІ КОНЦЕПТУАЛЬНОГО МОДЕЛЮВАННЯ

2.1. Процес проектування: від концепції до фізичної моделі даних

Процес проектування БД [6–14] – послідовність переходів від неформального словесного опису інформаційної структури предметної області до формалізованого опису об'єктів предметної області в термінах деякої моделі. У загальному випадку можна виділити такі етапи проектування (рис. 2.1) [8]:

- системний аналіз і словесний опис інформаційних об'єктів предметної області (підготовчий рівень);
- проектування концептуальної (інфологічної) моделі предметної

області – частково формалізований опис об'єктів предметної області в термінах деякої семантичної моделі;

- логічне (або даталогічне) проектування БД, тобто опис БД у термінах прийнятої даталогічної моделі даних;
- фізичне проектування БД, тобто вибір ефективного розміщення БД на зовнішніх носіях для забезпечення найбільш ефективної роботи програми.

Від того, наскільки кваліфіковано спроектовано БД, залежать продуктивність інформаційної системи й повнота забезпечення функціональних потреб користувачів і прикладних програм. Невдало спроектована БД може ускладнити процес розроблення прикладного програмного забезпечення, обумовити необхідність використання більш складної логіки, яка, у свою чергу, збільшить час реакції системи, а в подальшому може призвести до необхідності перепроектування логічної моделі БД. Реструктуризація (або внесення змін) логічної моделі БД – це дуже небажаний процес, оскільки він є причиною необхідності модифікації або навіть перепрограмування окремих завдань.

Рис. 2.1. Етапи проектування БД

Розглянемо поняття моделі даних і класифікацію моделей даних стосовно кожного з трьох рівнів проектування БД.

Модель даних – це деяка абстракція, яка, будучи застосованою до конкретних даних, дає змогу користувачам і розробникам трактувати їх уже як інформацію, тобто відомості, що містять не тільки дані, але й взаємозв'язок між ними.

Відповідно до розглянутих трьох етапів проектування БД (концептуального, логічного, фізичного) приходимо до понять моделі даних для кожного етапу. Класифікацію моделей даних, які застосовуються на тому чи іншому етапі проектування БД, показано на рис. 2.2.

Із наведеної класифікації моделей даних видно, що, з одного боку, на кожному етапі проектування БД необхідно прийняти обґрунтоване рішення відносного того, яку саме модель даних слід вибрати, а з іншого – одна й та ж концептуальна модель може бути перетворена на різні даталогічні моделі; у свою чергу, конкретна даталогічна модель може бути перетворена на різні фізичні моделі даних. Це дає змогу застосовувати різні моделі даних до однієї предметної області.

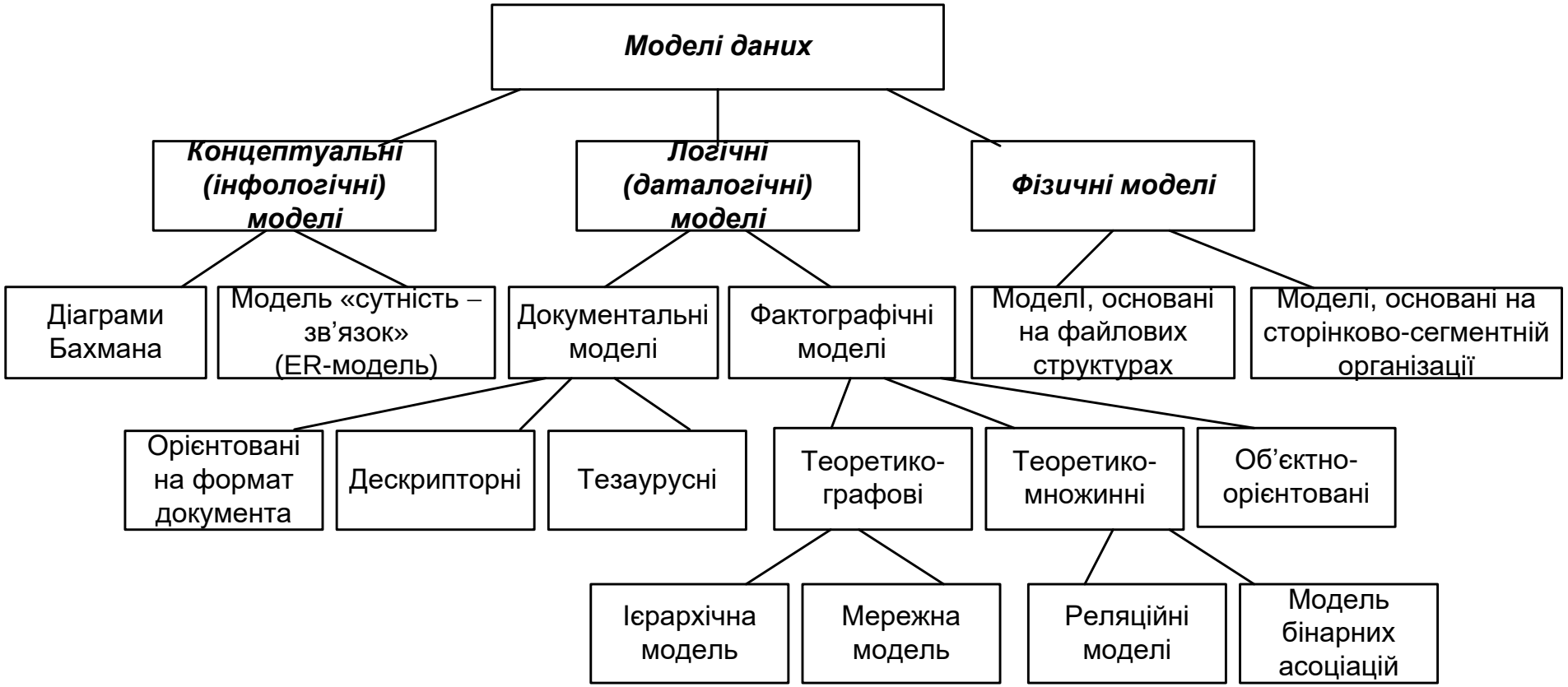


Рис. 2.2. Класифікація моделей даних

2.2. Системний аналіз предметної області

З точки зору проектування БД у межах системного аналізу необхідно здійснити перший етап, тобто провести докладний словесний опис об'єктів предметної області й реальних зв'язків, які існують між описаними об'єктами. Бажано, щоб цей опис давав змогу коректно визначити всі взаємозв'язки між об'єктами предметної області.

Найчастіше на практиці при виконанні аналізу предметної області рекомендується: з одного боку, орієнтуватися на конкретні завдання або функціональні потреби користувачів, а з іншого – ураховувати можливість нарощування нових завдань або навіть нових додатків.

Системний аналіз повинен закінчуватися:

а) докладним описом інформації про об'єкти предметної області, яка потрібна для вирішення конкретних завдань і має зберігатися в БД;

б) формулюванням конкретних завдань, які будуть вирішуватися з використанням цієї БД з коротким описом алгоритмів їх вирішення;

в) описом вихідних документів, які повинні генеруватися в системі;

г) описом вхідних документів, які є підставою для заповнення даними БД.

2.3. Побудова концептуальної (інфологічної) моделі предметної області

2.3.1. Семантичні моделі даних

Навіщо потрібна концептуальна модель і яку користь вона дає проектувальникам?

Концептуальне проектування, перш за все, пов'язане зі спробою подання семантики предметної області в моделі БД. Реляційна модель даних завдяки своїй простоті й лаконічності не дає змоги відобразити семантику, тобто зміст предметної області. Ранні теоретико-графові моделі більшою мірою відображали семантику предметної області. Вони в явному вигляді визначали ієрархічні зв'язки між об'єктами предметної області.

Проблема подання семантики давно цікавила розробників, і в 70-х роках XX ст. було запропоновано кілька моделей даних, названих семантичними моделями. До них можна віднести: семантичну модель даних, запропоновану Хаммером (Hammer) і Мак-Леоном (McLeon) у 1981 році, функціональну модель даних Шипмана (Shipman), також створену у 1981 році, модель "сутність – зв'язок", запропоновану Ченом (Chen) у 1976 році, і багато інших моделей. Усі моделі мали свої переваги й недоліки, але випробування часом витримала лише остання.

Сьогодні саме модель Чена "сутність – зв'язок", або "Entity Relationship", стала фактичним стандартом при інфологичному

моделюванні баз даних. Загальноприйнятою стала скорочена назва – ER-модель. Більшість сучасних CASE-засобів містять інструментальні засоби для опису даних із застосуванням цієї моделі. Крім того, розроблено методи автоматичного перетворення проекту БД з ER-моделі на реляційну, при цьому вона перетворюється на даталогічну модель, яка відповідає конкретній СУБД. Усі CASE-системи мають розвинені засоби документування процесу розроблення БД, автоматичні генератори звітів дають змогу підготувати звіт про поточний стан проекту БД з докладним описом об'єктів БД і їх відношень як у графічному вигляді, так і у вигляді готових стандартних друкованих звітів, що істотно полегшує ведення проекту.

На цей час не існує єдиної загальноприйнятої системи позначень (нотації) для ER-моделі, і різні CASE-системи використовують різні графічні нотації, але, розібравшись в одній, можна легко зрозуміти й інші нотації.

2.3.2. Базові поняття ER-моделі

Модель "сутність – зв'язок", або ER-модель, є найбільш відомим представником класу семантичних (концептуальних, інфологічних) моделей предметної області.

ER-модель – одна з найпростіших візуальних моделей даних, яка дає змогу визначити структуру "великими мазками" у загальних рисах. Цей загальний опис структури називається ER-діаграмою, або онтологією вибраної предметної області (area of interest).

Основні переваги ER-діаграм:

- наочність;
- модель дає змогу проектувати бази даних з великою кількістю об'єктів і атрибутів;
- ER-моделі реалізовані в багатьох системах автоматизованого проектування баз даних (наприклад, ERWin).

Як і будь-яка модель, модель "сутність – зв'язок" має кілька базових понять, з яких будуються вже більш складні об'єкти за заздалегідь визначеними правилами.

Ця модель найбільшою мірою узгоджується з концепцією об'єктно-орієнтованого проектування, яка в цій момент, безсумнівно, є базовою для розроблення складних програмних систем.

Основою ER-моделі є такі базові поняття:

- сутність;
- зв'язок;
- тип зв'язку (або множинність, потужність зв'язку);
- обов'язковість зв'язку (або клас належності зв'язку);
- категоризація сутностей.

Сутність – клас модельованих однотипних об'єктів. Сутність має ім'я, унікальне в межах системи, що моделюється. Оскільки сутність відповідає деякому класу однотипних об'єктів, то передбачається, що в системі існує безліч екземплярів цієї сутності.

Атрибут сутності. Об'єкт, якому відповідає поняття сутності, має свій набір атрибутів-характеристик, що визначають властивості даного представника класу. При цьому набір атрибутів повинен бути таким, щоб можна було розрізняти конкретні екземпляри сутності.

Опишемо сутність РОБІТНИК. Визначимо набір атрибутів для сутності РОБІТНИК, які характеризують екземпляри даної сутності. Екземпляром цієї сутності буде конкретний робітник фірми.

Щоб сформувати набір атрибутів сутності РОБІТНИК, необхідно відповісти на запитання: що характеризує конкретного працівника фірми? – Табельний номер, прізвище, ім'я, по батькові, дата народження, домашня адреса, домашній телефон.

Зазначимо, що набір атрибутів, який однозначно ідентифікує конкретний екземпляр сутності, називають ключовим. Для сутності РОБІТНИК ключовим буде атрибут "табельний номер", оскільки для всіх робітників цієї фірми табельні номери будуть різними.

Зв'язки. Це бінарні асоціації між сутностями, що показують, яким чином сутності співвідносяться або взаємодіють між собою. Зв'язок може існувати між двома різними сутностями або між сутністю і нею самою (рекурсивний зв'язок). Вона показує, як пов'язані екземпляри сутностей між собою. Якщо зв'язок установлюється між двома сутностями, то він визначає взаємозв'язок між екземплярами однієї та іншої сутностей.

Зв'язок між сутностями характеризується певним типом. За типом розрізняють зв'язки:

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

Зв'язок 1:1 означає, що екземпляр однієї сутності зв'язаний тільки з одним екземпляром іншої сутності. Наприклад, між сутностями РОБІТНИК і ПАСПОРТ може бути встановлений зв'язок 1:1, оскільки робітник має один паспорт (наявність закордонного паспорта не береться до уваги), і водночас конкретний паспорт може належати тільки одному робітнику фірми.

Зв'язок 1:N означає, що один екземпляр сутності, розташований ліворуч від зв'язку, може бути зв'язаний з декількома екземплярами сутності, розташованої праворуч від зв'язку. Наприклад, між сутностями РОБІТНИК і ЗАМОВЛЕННЯ може бути встановлений зв'язок 1:N, оскільки один робітник може розмістити багато замовлень клієнтів фірми, але конкретне замовлення може бути оформлене тільки одним робітником фірми.

Зв`язок N:M означає, що один екземпляр першої сутності може бути зв'язаний з декількома екземплярами другої сутності, і навпаки, один екземпляр другої сутності може бути зв'язаний з декількома екземплярами першої сутності. Наприклад, між сутностями РОБІТНИК і ВІДДІЛ може бути встановлений зв'язок N:M, якщо з предметної області випливає, що робітник може працювати в декількох відділах і в кожному відділі може бути багато робітників.

Між двома сутностями може бути задано декілька зв'язків з різним смисловим навантаженнями, при цьому встановлені зв'язки можуть мати різні типи множинності, що визначається смисловим навантаженням зв'язку.

Обов'язковість зв'язку. Зв'язок будь-якого з цих типів може бути обов'язковим, якщо у цьому зв'язку повинен брати участь кожен екземпляр сутності, і необов'язковим, якщо не кожен екземпляр сутності повинен брати участь у даному зв'язку. При цьому зв'язок може бути обов'язковим, з одного боку, і необов'язковим – з іншого.

Розглянемо зв'язок між сутностями РОБІТНИК і ЗАМОВЛЕННЯ. З предметної області відомо, що замовлення обов'язково оформлюється працівником фірми, водночас працівник може не мати розміщених ним замовлень (наприклад, він тільки влаштувався на роботу). Таким чином, між сутностями РОБІТНИК і ЗАМОВЛЕННЯ буде встановлено зв'язок, обов'язковий з боку сутності РОБІТНИК і необов'язковий з боку сутності ЗАМОВЛЕННЯ.

Категоризація сутностей. В ER-моделі допускається принцип категоризації сутностей. Це означає, що, як і в об'єктно-орієнтованих мовах програмування, вводиться поняття підтипу сутності, тобто сутність може бути подана у вигляді двох або більше своїх підтипів – сутностей, кожний з яких може мати загальні атрибути й зв'язки, які визначаються один раз на верхньому рівні й успадковуються на нижньому рівні, і/або особисті атрибути й відносини.

Як підсумок зазначимо, що на етапі переходу до реалізації ER-діаграми у вигляді реальної інформаційної системи або програми, відбувається відображення ER-моделі в більш детальну модель даних реляційної (об'єктної, мережної, логічної або ін.) бази даних, яка називається даталогічною моделлю даних відносно вихідної ER-діаграми.

2.3.3. Нотації для побудови ER-діаграм

На цей момент не існує єдиної загальноприйнятої системи позначень для ER-моделі, і різні CASE-системи використовують різні графічні нотації. Перший варіант моделі "сутність – зв'язок" запропонував Пітер Чен. Надалі багато авторів розробили свої варіанти подібних моделей (нотація Мартіна, нотація IDEF1X, нотація Баркера та ін.). Крім того, різні програмні засоби, що реалізують одну й ту ж нотацію, можуть відрізнятися своїми

можливостями. По суті всі нотації діаграми "сутність – зв'язок" випливають з однієї ідеї – рисунок завжди є наочнішим за текстовий опис. Усі такі діаграми використовують графічне зображення сутностей предметної області, їх властивостей (атрибутів) і взаємозв'язків між сутностями.

Нотація Пітера Чена

Розглянемо зображення основних елементів ER-діаграми в нотації Пітера Чена (табл. 2.1).

Таблиця 2.1

Елементи ER-діаграми в нотації Чена

Елемент діаграми	**Позначення**
ім’я	Незалежна сутність
ім’я	Залежна сутність
ім’я	Батьківська сутність в ієрархічному зв'язку
ім’я	Зв'язок
ім’я	Ідентифікувальний зв'язок
ім’я	Атрибут
ім’я	Первинний ключ
ім’я	Зовнішній ключ
ім’я	Багатозначний атрибут
ім’я	Похідний атрибут

Для зображення сутності використовуються прямокутники, всередині яких указується ім'я сутності, яке виражається іменником. Атрибут сутності зображується у вигляді овалу, який зв'язується лінією з відповідною сутністю. Зв'язки відображаються у вигляді ромбів. Асоційовані сутності

зв'язуються лініями. Якщо зв'язок не є обов'язковим, то лінія – пунктирна. Біля кожної сутності на лінії, що з'єднує її зі зв'язком, цифрами вказується клас належності. На рис. 2.3 наведено приклад ER-діаграми в нотації Пітера Чена.

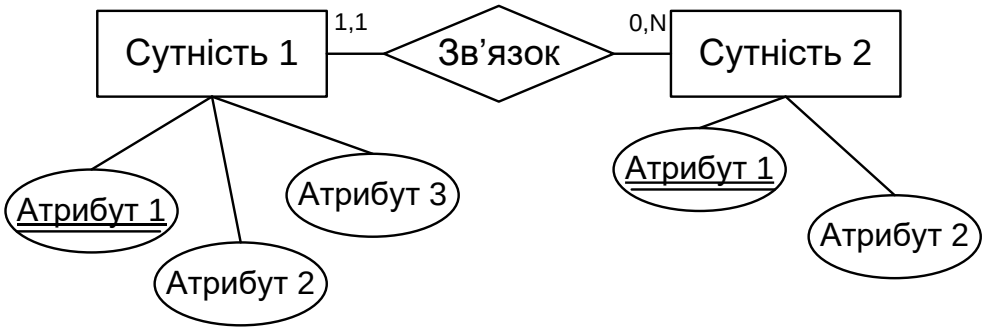


Рис. 2.3. ER-діаграма в нотації Чена

Нотація Мартіна

Відповідно до цієї нотації (табл. 2.2) сутність зображується у вигляді прямокутника, що містить її ім'я, яке виражається іменником. Ім'я сутності має бути унікальним у межах однієї моделі.

Таблиця 2.2

Елементи ER-діаграми в нотації Мартіна

Елемент діаграми	**Позначення**
ім’я	Незалежна сутність
ім’я	Залежна сутність
ім’я	Батьківська сутність в ієрархічному зв'язку
	Кардинальність зв'язку – невстановлена
	Кардинальність зв'язку – 1,1
	Кардинальність зв'язку – 0,1
	Кардинальність зв'язку – M,N
	Кардинальність зв'язку – 0,N
	Кардинальність зв'язку – 1,N

Зв'язок зображується лінією, яка зв'язує дві сутності, що беруть участь у зв'язку. Ступінь кінця зв'язку вказується графічно, множинність зв'язку зображується у вигляді "вилки" на кінці зв'язку. Модальність зв'язку також зображується графічно – необов'язковість зв'язку позначається кружком на кінці зв'язку. Найменування зазвичай виражається одним дієсловом у теперішньому часі ("має", "належить" і т. д.) або дієсловом з пояснювальними словами ("містить" і т. ін.). Найменування може бути одне для всього зв'язку або два – для кожного з кінців зв'язку. У другому випадку назва лівого кінця зв'язку вказується над лінією зв'язку, а правого – під лінією. Кожна з назв розташовується поруч із сутністю, до якої вона належить.

Атрибути сутності записуються всередині прямокутника, що зображує сутність, і виражаються іменником в однині (можливо з уточнюючими словами). Серед атрибутів виділяється ключ сутності. Ключові атрибути підкреслюються.

На рис. 2.4 наведено фрагмент ER-діаграми в нотації Мартіна.

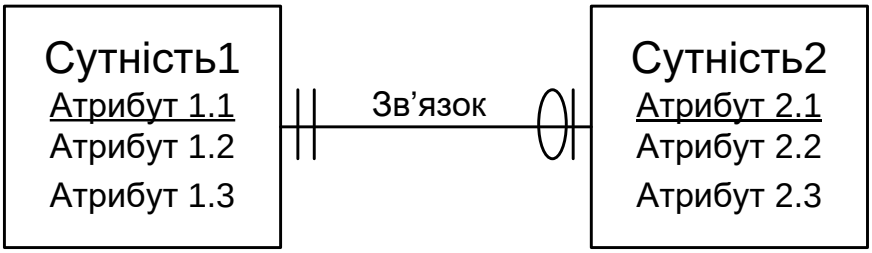


Рис. 2.4. Фрагмент ER-діаграми в нотації Мартіна

Нотація Баркера

Відповідно до цієї нотації (табл. 2.3) сутності позначаються прямокутниками, всередині яких наводиться список атрибутів. Ключові атрибути маркуються символом "#" (решітка). Зв'язки позначаються лініями з іменами, місце з'єднання зв'язку і сутності показує кардинальність зв'язку.

На рис. 2.5 показано фрагмент ER-діаграми в нотації Баркера.

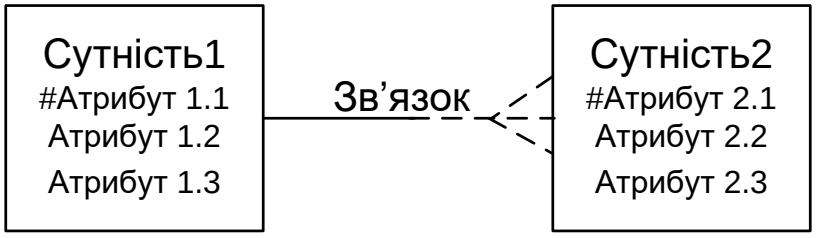


Рис. 2.5. Фрагмент ER-діаграми в нотації Баркера

Таблиця 2.3

Елементи ER-діаграми в нотації Баркера

Елемент діаграми	Позначення
ім’я	Сутність
	Зв'язок 1:1
	Зв'язок 1:N
	Необов'язковий зв'язок
	Обов'язковий зв'язок

Для позначення відношення категоризації вводиться елемент "дуга" (рис. 2.6).

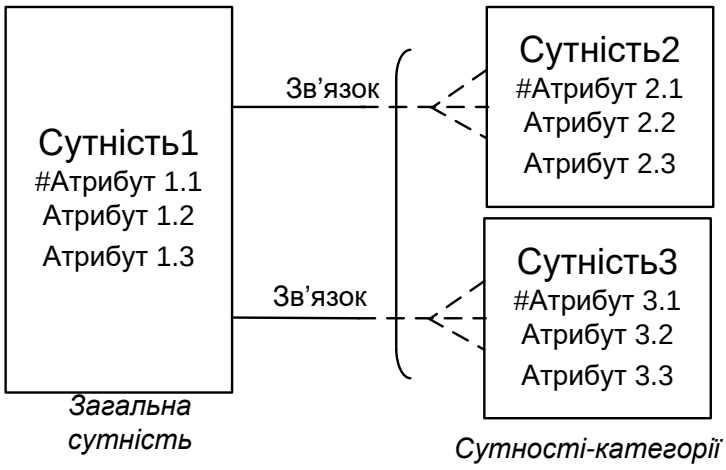


Рис. 2.6. Позначення категоризації в нотації Баркера

Нотація IDEF1X

Позначення конструктивних елементів концептуальної схеми бази даних нотації IDEF1X наведено в табл. 2.4.

Список атрибутів указується всередині прямокутника, що позначає сутність. Атрибути, що становлять ключ сутності, групуються у верхній частині прямокутника і відокремлюються горизонтальною лінією (рис. 2.7).

Крім того, в IDEF1X вводиться поняття відношення категоризації, за змістом еквівалентне ієрархічному зв'язку. Позначення зв'язку повної категоризації (сутності-категорії становлять повну множину нащадків батьківської сутності) зображено на рис. 2.8, а, позначення зв'язку неповної категоризації (сутності-категорії становлять неповну множину нащадків загальної сутності) – на рис. 2.8, б.

Таблиця 2.4

Елементи ER-діаграми в нотації IDEF1X

Елемент діаграми	Позначення
ім’я	Незалежна сутність
ім’я	Залежна сутність
	Ідентифікувальний зв'язок
	Неідентифікувальний зв'язок
Сутність1 — Сутність2	Сутності 1 відповідає 0, 1 або багато (M) сутностей 2
Сутність1 — P — Сутність2	Сутності 1 відповідає 1 або M сутностей 2
Сутність1 — Z — Сутність2	Сутності 1 відповідає 0 або 1 сутностей 2
Сутність1 — 7 — Сутність2	Сутності 1 відповідають рівно N (N = 7) сутностей 2

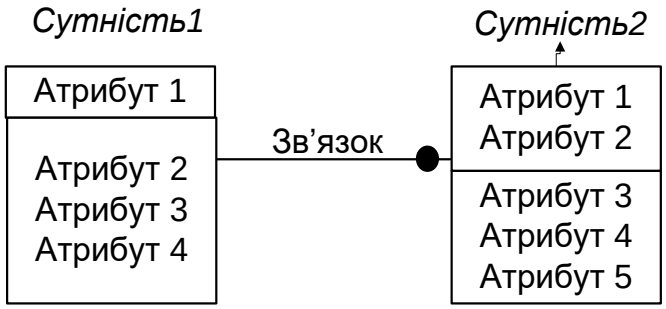


Рис. 2.7. Фрагмент ER-діаграми в нотації IDEF1X

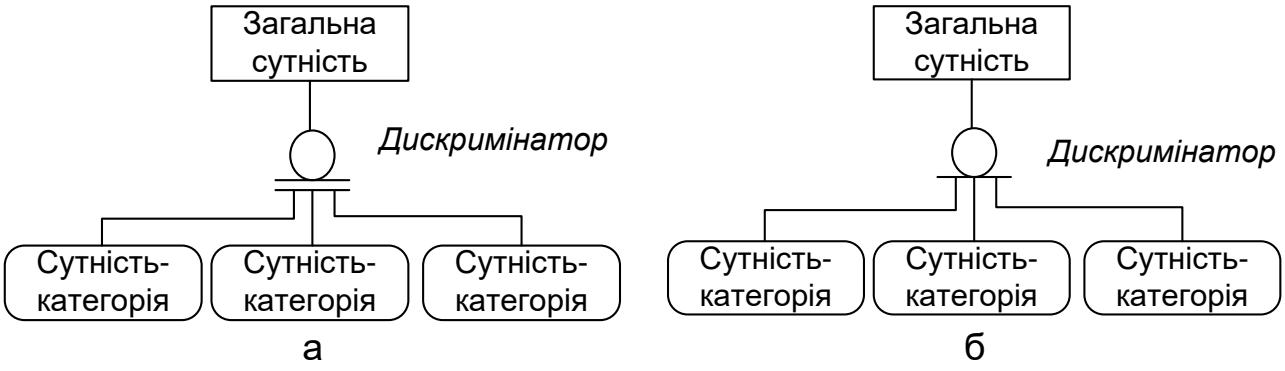


Рис. 2.8. Позначення повної (а) і неповною (б) категоризації в нотації IDEF1X

2.4. Побудова логічної моделі реляційної бази даних

Інфологічна модель використовується на ранніх стадіях розроблення проекту. Якщо розуміти мову умовних позначень, які відповідають категоріям ER-моделі, то її можна легко "читати", отже, вона є доступною для аналізу програмістам-розробникам, які будуть розробляти окремі програми. Ця мова має однозначну інтерпретацію на відміну від деяких речень природної мови, і тому тут не може бути ніякого нерозуміння з боку розробників.

Мова ER-моделі стала умовною загальноприйнятою мовою опису бази даних. Для ER-моделі існує алгоритм однозначного перетворення її на реляційну модель даних, що дало змогу в подальшому розробити безліч інструментальних систем, які підтримують процес розроблення інформаційних систем, що базуються на технології баз даних.

Логічне (даталогічне) проектування бази даних – відображення інфологічної моделі на модель даних, яка використовується в конкретній СУБД, наприклад на реляційну модель даних. Для реляційних СУБД даталогічна модель – набір таблиць зазвичай із зазначенням ключових полів, зв'язків між таблицями. Якщо інфологічну модель побудовано у вигляді ER-діаграми (або інших формалізованих засобів), то даталогічне проектування – це побудова таблиць за певними формалізованими правилами, а також нормалізація цих таблиць. Цей етап може бути значною мірою автоматизованим.

2.4.1. Правила перетворення сутностей і їх атрибутів

Кожній сутності ставиться у відповідність відношення (таблиця) реляційної моделі даних. При цьому імена сутності й відношення можуть бути різними, тому що на імена сутностей можуть не накладатися додаткові синтаксичні обмеження, крім унікальності імені у межах моделі. Ім'я відношення може бути обмежене вимогами конкретної СУБД. Найчастіше ці імена є ідентифікаторами в деякій базовій мові. Вони обмежені по довжині й не повинні містити пропусків і деяких спеціальних символів. Наприклад, сутність може називатися "КНИГА", а відповідне їй відношення може називатися, наприклад, "BOOK".

Кожен атрибут сутності стає атрибутом відповідного відношення. Перейменування атрибутів має відбуватися відповідно до тих же правил, що і перейменування відношення. Для кожного атрибута задаються конкретний допустимий у СУБД тип даних і обов'язковість або необов'язковість цього атрибута (допустимість або недопустимість NULL-значень для нього).

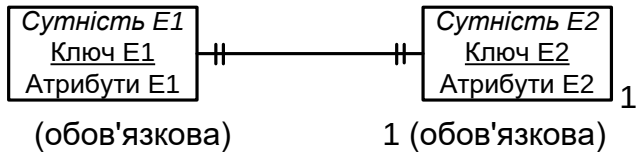
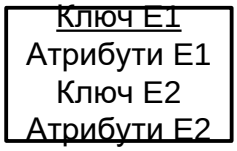
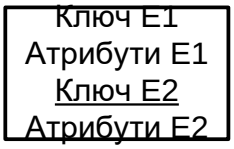
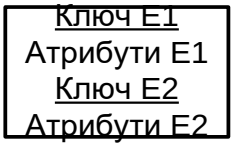
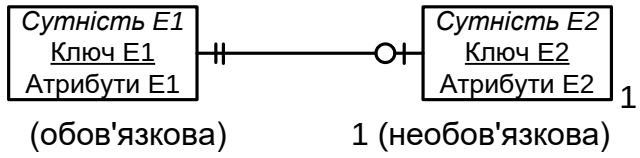
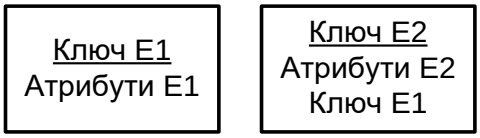
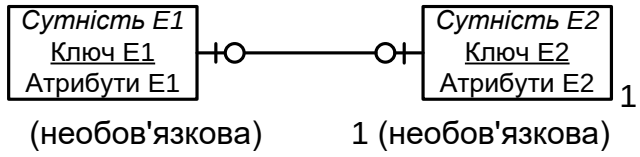
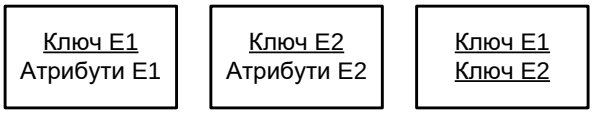
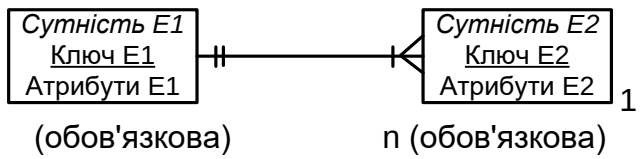
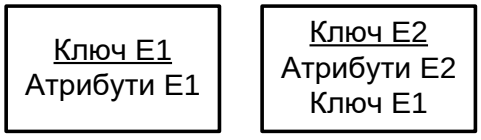
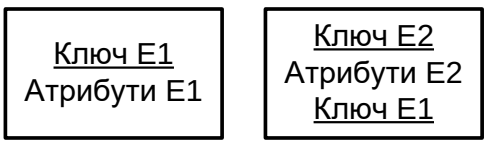
Первинний ключ сутності стає PRIMARY KEY відповідного відношення. Атрибути, що входять у первинний ключ відношення, автоматично отримують властивість обов'язковості (NOT NULL).

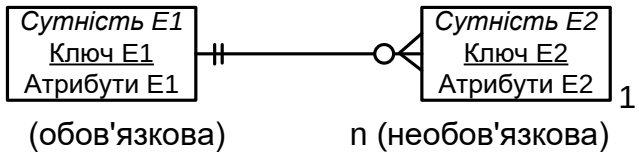
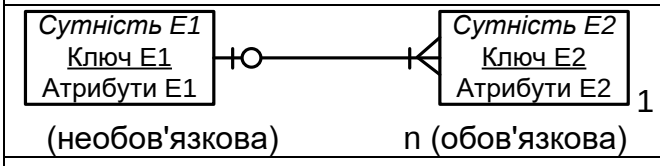
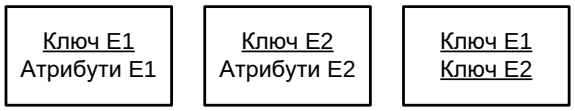
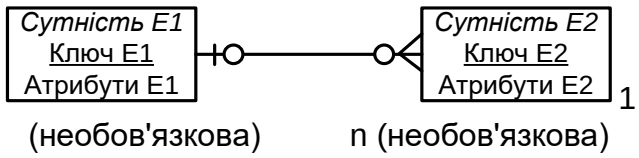
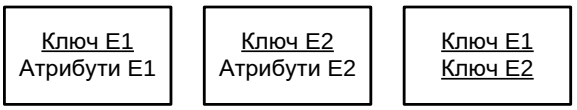
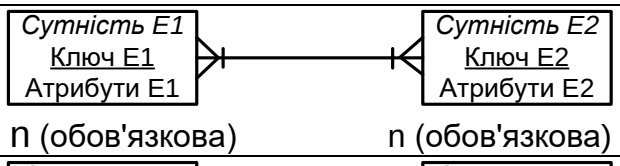
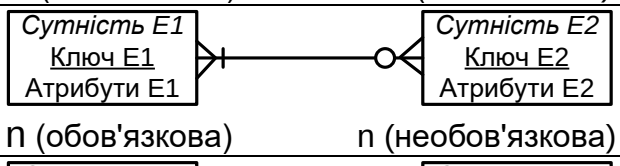
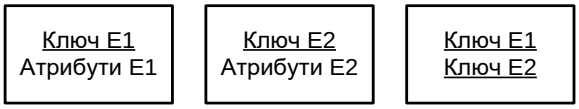
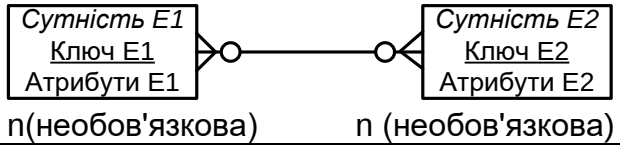
2.4.2. Правила перетворення зв'язків між сутностями

У цьому підрозділі розглянуто правила перетворення зв'язків типу "один до одного", "один до багатьох" та "багато до багатьох" між сутностями (табл. 2.5).

Таблиця 2.5

Перетворення зв'язків

Тип зв'язку	*Результат перетворення*
"Один до одного"	
	  
	
	
"Один до багатьох"	
	 

Тип зв'язку	*Результат перетворення*
	
	
	
"Багато до багатьох"	
	
	
	

2.5. Побудова фізичної моделі реляційної бази даних

Фізичне проектування бази даних – реалізація даталогічної моделі засобами конкретної СУБД, а також вибір рішень, пов'язаних з фізичним середовищем зберігання даних: вибір методів управління дисковою пам'яттю, методів доступу до даних, методів стиснення даних і т. д. Ці задачі вирішуються в основному засобами СУБД і приховані від розробника БД.

Для побудови фізичної моделі реляційної бази даних необхідно згідно з роботою [7]:

- створити об'єкти для зберігання даних;
- урахувати вплив транзакцій.

2.5.1. Забезпечення зберігання даних у БД

Створення фізичної моделі бази даних потребує від розробників досягнення такої мети:

– задоволення потреб у зберіганні даних предметної області у межах реляційної моделі даних, тобто створення базових таблиць для зберігання інформації про всі сутності предметної області;

– задоволення вимог цілісності даних, тобто визначення типів стовпців і накладання обмежень на значення стовпців базових таблиць, які випливають з бізнес-правил предметної області;

– задоволення вимог посилальної цілісності, тобто при прийнятті рішення про підтримку посилальної цілісності вбудованими засобами СУБД накладання обмежень посилальної цілісності на таблиці, які випливають з бізнес-правил посилальної цілісності предметної області;

– часткове задоволення вимог незалежності подання даних для кінцевого користувача від характеру фізичного зберігання даних, тобто побудова першої ітерації зовнішньої схеми.

Таким чином, побудова фізичної моделі даних зводиться до створення таблиць, індексів, розрізів (англ. view) в базі даних, в якій буде зберігатися інформація про сутності предметної області. Виконуючи це завдання, розробник бази даних відображає відношення логічної моделі реляційної бази даних у таблиці й індекси реляційної бази даних. Для цього використовують підмножину команд мови SQL – мову визначення даних DDL (Data Definition Language). Наприклад, для СУБД Oracle ці дії можна виконати в програмі SQL*PLUS.

У межах концепції трирівневої архітектури баз даних ANSI/SPARC побудову фізичної моделі даних називають ще створенням внутрішньої схеми бази даних. Результатом є скрипт для створення таблиць та індексів, що становить початковий прототип фізичної моделі бази даних.

Детальніше розглянемо таку задачу зберігання даних, як забезпечення посилальної цілісності.

Поняття посилальної цілісності

Посилальна цілісність (англ. referential integrity) – необхідна якість реляційної бази даних, що полягає у відсутності в будь-якому її відношенні зовнішніх ключів, які посилаються на кортежі, що не існують.

Зв'язки між даними, збереженими в різних відношеннях, у реляційній БД установлюються шляхом використання зовнішніх ключів. Завдяки наявності зв'язків у реляційній БД можна зберігати факти без надмірного дублювання, тобто в нормалізованому вигляді.

Посилальну цілісність можна визначити таким чином. Дано пару відношень A і B, зв'язаних зовнішнім ключем. Первинний ключ відношення

B – атрибут B.key. Зовнішній ключ відношення A, який посилається на B, – атрибут A.b. Посилальна цілісність для пари відношення A і B має місце тоді, коли виконується така умова: для кожного кортежу відношення A існує відповідний кортеж відношення B, тобто кортеж, у якого B.key = A.b.

База даних має властивість посилальної цілісності, коли в ній для будь-якої пари зв'язаних зовнішнім ключем відношень виконується умова посилальної цілісності.

Якщо наведена вище умова не виконується, то це означає, що в базі даних порушено посилальну цілісність. Така БД не може нормально експлуатуватися, оскільки в ній є розірваними логічні зв'язки між фактами, що залежать один від одного. Безпосереднім результатом порушення посилальної цілісності стає те, що коректним запитом не завжди вдається отримати коректний результат.

Наведемо приклад порушення посилальної цілісності в базі даних (рис. 2.9).

Таблиця "**РОБІТНИК"**

Табельний_номер	Прізвище	Ім'я	По_батькові	Рік_народження	Номер_відділу
10011	Іванов	Іван	Іванович	1975	1
10022	Петров	Петро	Петрович	1977	2
10033	Сідоров	Сідор	Сідорович	1980	3
10044	Ніколаєв	Микола	Миколайович	1978	4
10055	Михайлов	Михайло	Михайлович	1973	5
10066	Сергієв	Сергій	Сергійович	1983	Null

Таблиця "**ВІДДІЛ"**

Номер_відділу	Назва_відділу
1	Цех
2	Плановий відділ
3	Бухгалтерія
4	Охорона

?

Рис. 2.9. Приклад БД з порушенням посилальної цілісності

БД складається з таблиць "РОБІТНИК" і "ВІДДІЛ", які забезпечують зберігання особистої інформації про робітників та інформації про те, в якому відділі працює кожний із робітників. Очевидно, що повноцінна інформація про робітника має бути подана двома зв'язаними записами в обох названих таблицях, що технічно виражається такою умовою: для будь-якого запису таблиці "РОБІТНИК" у таблиці "ВІДДІЛ" повинен існувати відповідний запис, для якого має виконуватися умова РОБІТНИК.Номер_відділу = ВІДДІЛ.Номер_відділу.

Щоб отримати інформацію про всіх працівників із зазначенням назви відділу, в якому вони працюють, з таблиць, в яких дотримується посилальна цілісність, досить застосувати до цих таблиць SQL-запит:

Select *
From Робітник, Відділ
Where Робітник.Номер_відділу = Відділ.Номер_відділу;

Однак у цьому прикладі порушено посилальну цілісність. Запис таблиці "РОБІТНИК" (Табельний_номер = 10055) має в полі "Номер_відділу" так зване "висяче" посилання – значення, яке не відповідає запису в таблиці "ВІДДІЛ". Через це результат наведеного вище запиту не буде містити запис про робітника з табельним номером 10055.

Зазначимо, що в записі, який відповідає робітнику з табельним номером 10066, використовується варіант навмисного (у деяких випадках легального) порушення посилальної цілісності – у полі зовнішнього ключа записано значення NULL. Щоб отримати інформацію про робітника, навіть того, у якого не вказано відділ, в якому він працює, необхідно використати зовнішнє з'єднання, яке в синтаксисі Oracle записується так:

Select *
From Робітник, Відділ
Where Робітник.Номер_відділу = Відділ.Номер_відділу (+);

Причини порушень посилальної цілісності БД

Правильно спроектована й підтримувана база даних не допускає можливості порушення посилальної цілісності. Проте такі порушення можуть з'явитися під час експлуатації бази даних з багатьох причин. Наведемо деякі з них.

Некоректна робота прикладного програмного забезпечення. Зрозуміло, що при помилці в програмі, яка виконує модифікацію бази даних, база може бути модифікована неприпустимим чином, унаслідок чого утворяться "висячі" посилання. Програма може робити помилки таких видів:

1. *Неповний запис об'єктів.* Дані об'єкта розміщуються в записах кількох таблиць, а програма не записує якусь із них.
2. *Некоректне виправлення посилання.* Значення зовнішнього ключа змінюється на таке, якому не відповідає ні один запис у зв'язаній таблиці.
3. *Виправлення первинного ключа без каскадного оновлення.* У таблиці, на яку є посилання, виправляється первинний ключ, але при цьому зовнішні ключі в зв'язаних з нею таблицях залишаються без зміни.

4. *Видалення запису без каскадного оновлення.* З таблиці видаляється запис, на який є посилання із зовнішніх ключів інших таблиць, при цьому в зв'язаних записах зовнішні ключі не змінюються. Унаслідок цього всі записи інших таблиць, які мають посилання на неї, стають некоректними.

Збої в роботі системного програмного забезпечення й обладнання. Навіть коли прикладне програмне забезпечення працює абсолютно правильно, порушення посилальної цілісності є можливим. Наприклад, якщо при додаванні об'єкта в базу потрібно додати кілька зв'язаних записів у декілька таблиць, очевидно, що посилальну цілісність буде порушено в процесі додавання даних (коли частину зв'язаних записів уже додано, а частину – ще ні) і відновлено тільки після завершення операції. Якщо під час виконання операції її буде перервано (через переповнення диска, збій живлення або з якихось інших причин), частину записів буде додано в БД, а частину – ні. Частина доданих записів залишиться з некоректними посиланнями.

Забезпечення посилальної цілісності

Пусті зовнішні ключі

Можлива ситуація, коли зовнішній ключ замість посилання на існуючий запис у таблиці БД містить значення NULL. Такий стан можна трактувати як відсутність якоїсь частини об'єкта. Хоча з точки зору чистої теорії це є неприпустимим, на практиці іноді буває зручно використовувати порожні зовнішні ключі. Щоб коректно працювати з групами зв'язаних таблиць, що допускають порожні зовнішні ключі, слід використовувати специфічну операцію мови SQL – зовнішнє з'єднання (англ. outer join).

Транзакції

Обов'язковою (хоча і недостатньою) умовою збереження посилальної цілісності бази даних є підтримка транзакцій. Якщо програмне забезпечення виконує групу зв'язаних між собою операцій, які окремо можуть призводити до порушення цілісності посилань, то СУБД повинна давати змогу виконувати всі ці групи операцій в одній транзакції, тобто так, щоб при будь-якому збої проводилося автоматичне скасування всіх операцій групи, у тому числі вже повністю завершених.

Посилальна цілісність на тригерах

Можливим є забезпечення посилальної цілісності БД з використанням механізму тригерів. У цьому випадку для будь-якої потенційно небезпечної операції над таблицею створюється тригер, який

проводить необхідні перевірки або навіть змінює дані у зв'язаних таблицях, щоб виключити втрату посилань.

Так, для забезпечення каскадних змін тригер може бути встановлений на операцію зміни запису в таблиці. Якщо виявиться, що при редагуванні змінилося значення ключового поля, тригер повинен здійснити узгоджені зміни в усіх таблицях, зв'язаних з даною, замінивши попереднє значення зовнішніх ключів на нове.

Для виключення втрати посилань від некоректного редагування зовнішнього ключа тригер повинен при кожній зміні відповідного поля перевіряти, чи є в зв'язаній таблиці запис з таким первинним ключем.

Для захисту від видалення запису, на який є посилання, тригер на зв'язаній таблиці повинен перевіряти наявність посилань і залежно від необхідності або забороняти видалення, або обнуляти зовнішні ключі тим чи іншим чином.

Посилальна цілісність на зовнішніх ключах

СУБД може мати механізм автоматичної підтримки посилальної цілісності, оснований на явному описі посилань при створенні БД. При описі таблиць БД програміст явно описує, які поля таблиць є зовнішніми ключами і на які таблиці вони посилаються. Ця інформація зберігається в службових областях пам'яті БД. Будь-яка операція, що змінює дані в таблиці, викликає автоматичну перевірку посилальної цілісності.

При операції додавання або редагування запису автоматично перевіряється, чи посилаються зовнішні ключі у цьому записі на існуючі записи в заявлених при описі зв'язаних таблицях. Якщо з'ясовується, що операція призведе до появи некоректних посилань, вона не виконується – система повертає помилку.

При операції редагування запису перевіряється, чи не змінюється її первинний ключ і чи немає на неї посилань. Якщо первинний ключ змінюється і при цьому на даний запис є посилання, то операція редагування завершується з помилкою.

При операції видалення запису перевіряється, чи немає на нього посилань. Якщо посилання існує, то можливими є три варіанти подальших дій (варіант, що виконується, залежить від СУБД і вибору програміста, який він повинен зробити при описі зв'язку):

1. *Заборона* – видалення блокується і повертається помилка.
2. *Каскадне видалення* – в одній транзакції проводиться видалення цього запису і всіх записів, які посилаються на даний. Якщо на записи, що видаляються, також є посилання і настройки також потребують видалення, то каскадне видалення триває далі. Таким чином, після видалення цього запису в базі не залишається жодного запису, який прямо або побічно посилається на нього. Якщо хоча б один з таких записів видалити не

можна (або для нього налаштовано заборону, або відбувається якась ще помилка), то всі видалення забороняються.

3. *Обнулення зовнішніх ключів* – в усі зовнішні ключі записів, що посилаються на даний, записується псевдозначення NULL (SQL). Якщо хоча б для одного запису, що посилається, це є неможливим (наприклад, якщо поле зовнішнього ключа описано так, що його не можна обнуляти), то видалення забороняється.

2.5.2. Забезпечення продуктивності БД

Крім того, потрібно досягти й головної мети фізичного проектування реляційної бази даних – дати гарантію того, що база даних забезпечує необхідний рівень продуктивності. Зазвичай продуктивність бази даних вимірюється в термінах продуктивності транзакцій (transaction performance).

Головним фактором для розробника бази даних у боротьбі за продуктивність транзакцій є вибір фізичних конструкцій СУБД.

Щоб успішно це зробити, розробник бази даних повинен мати добре визначені транзакції, які можуть існувати в базі даних. Однак у більшості випадків отримати спочатку повністю прописані транзакції до бази даних є досить складним організаційним завданням. В умовах недостатніх відомостей про транзакції розробнику доводиться часто покладатися на свій власний досвід проектування, виконувати вибіркове настроювання отриманої першої ітерації фізичної моделі бази даних і переадресовувати остаточне виконання цього завдання адміністратору бази даних. До того ж частина проблем, яка пов'язана з продуктивністю бази даних, може бути виявлена лише на стадії тестування й дослідної експлуатації бази даних.

Зазначимо, що при виконанні цього етапу можна спиратися тільки на знання стандарту SQL. На допомогу приходять конструкції конкретної СУБД, вибраної для реалізації бази даних. Основними механізмами промислових СУБД для підвищення продуктивності є денормалізація, індекси, кластерізація й розділення.

3. ДЕТАЛЬНИЙ ПЛАН ПРОЕКТУВАННЯ БАЗИ ДАНИХ НА ОСНОВІ КОНЦЕПТУАЛЬНОГО МОДЕЛЮВАННЯ

Відповідно до викладеного матеріалу наведемо детальний план проектування бази даних на основі концептуального моделювання.

Попереднім етапом проектування БД є аналіз вимог до програмного забезпечення.

Етап 1. Аналіз вимог до БД

На основі аналізу UML-діаграми варіантів використання (use-case-діаграми) і вимог до програмного забезпечення слід:

– виділити ті функції розроблюваного ПЗ, для реалізації яких буде необхідною БД;

– проаналізувати дані, які зберігатимуться в БД;

– проаналізувати вихідні дані (документи), які будуть видаватися (генеруватися) на основі даних БД.

Етап 2. Концептуальне проектування бази даних

Для побудови ER-моделі, яка є концептуальною моделлю предметної області, необхідно виконати такі дії:

– виділити сутності предметної області;

– визначити атрибути кожної сутності;

– для кожної сутності визначити атрибути, які мають унікальні значення (можуть бути первинним ключем даної сутності);

– установити зв'язки між сутностями і задати їх імена;

– установити тип зв'язків між сутностями: "один до одного" (1:1), "один до багатьох" (1:N), "багато до багатьох" (N:M);

– установити обов'язковість зв'язків між сутностями;

– побудувати ER-модель предметної області за допомогою вибраної нотації (графічної мови моделювання);

– виділити обмеження, які накладаються бізнес-правилами (наприклад, "Клієнт може відкрити не більше трьох банківських рахунків").

Етап 3. Логічне проектування БД

На основі спроектованої концептуальної моделі предметної області у вигляді ER-моделі слід побудувати модель (схему) реляційної бази даних.

Для цього необхідно:

– за допомогою правил перетворення ER-моделі в реляційну схему бази даних визначити набір таблиць БД (див. підрозд. 2.4.1);

– реалізувати зв'язки між сутностями шляхом міграції первинних ключів і дочірні таблиці (див. підрозд. 2.4.2);

– визначити домени стовпців таблиць (число, рядок, дата);

– визначити обмеження на значення стовпців таблиць (обов'язковість даних, перелік можливих значень, значення за замовчуванням);

– перевірити отримані відношення за допомогою правил нормалізації (обґрунтувати кількість нормальних форм, яким відповідає розроблена БД);

– словесно сформулювати основні запити до БД (наприклад, "Вивести список усіх працівників для заданого відділу" і т. д.);

– словесно сформулювати призначення необхідних функцій і процедур (наприклад, "Виконати розрахунок премій співробітників: 15 % від продажів у розрахунковому місяці");

– словесно сформулювати необхідні тригери для реалізації обмежень, які накладаються бізнес-правилами і будуть виділені на етапі концептуального проектування (наприклад, "Неприпустимо приймати на роботу співробітників, яким більше 60 років");

– забезпечити цілістність БД, для цього слід описати можливі транзакції (наприклад, "Перерахування грошей з банківського рахунка А на рахунок В": дія 1 – списання суми грошей з рахунка А; дія 2 – зарахування такої самої суми грошей на рахунок В).

Етап 4. Фізичне проектування БД

На основі побудованої логічної моделі БД потрібно розробити її фізичну модель.

Для цього необхідно:

– обґрунтовано вибрати СУБД;

– реалізувати таблиці, тобто розробити SQL-скрипт побудови таблиць для вибраної СУБД;

– указати обов'язковість (NOT NULL) значень атрибутів таблиць;

– якщо необхідно, вказати значення за замовчуванням (DEFAULT);

– реалізувати обмеження (constraints) предметної області на значення атрибутів таблиць (BETWEEN, CHEC);

– указати первинні ключі таблиць (PRIMARY KEY);

– реалізувати обмеження зовнішньої посилальності, тобто вказати зовнішні ключі (FOREING KEY);

– визначити індекси;

– реалізувати запити до БД, які було сформульовано на етапі логічного проектування;

– у разі потреби розробити процедури й функції, що зберігаються;

– якщо необхідно, розробити тригери бази даних;

– у разі потреби реалізувати транзакції;

– визначити вимоги до дискової пам'яті для забезпечення функціонування БД;

– проаналізувати пропускну здатність транзакцій;

– проаналізувати час відповіді для виконання однієї транзакції.

4. ПРИКЛАД ПРОЕКТУВАННЯ РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ ДЛЯ БІБЛІОТЕКИ НА ОСНОВІ КОНЦЕПТУАЛЬНОГО МОДЕЛЮВАННЯ

4.1. Системний аналіз предметної області

Розглянемо приклад системного аналізу предметної області, наведений у роботі [9].

Нехай потрібно розробити інформаційну систему для автоматизації обліку отримання й видачі книг у бібліотеці. Система повинна передбачати режими ведення системного каталогу, що відображає перелік галузей знань, за якими є книги в бібліотеці. Усередині бібліотеки області знань в систематичному каталозі можуть мати унікальний внутрішній номер і повне найменування. Кожна книга може містити відомості з кількох областей знань і бути в бібліотеці в декількох примірниках. Усі книги, що зберігаються в бібліотеці, характеризуються такими параметрами:

- унікальний шифр;
- назва;
- прізвища авторів (можуть бути відсутніми);
- місце видання (місто);
- видавництво;
- рік видання;
- кількість сторінок;
- вартість;
- кількість примірників книги в бібліотеці.

Книги можуть мати однакові назви, але вони розрізняються за своїм унікальним шифром (ISBN).

У бібліотеці ведеться картотека читачів.

На кожного читача в картотеку заносяться такі відомості:

- прізвище, ім'я, по батькові;
- домашня адреса;
- телефон (будемо вважати, що є два телефони – робочий і домашній);
- дата народження.

Опис предметної області

Кожному читачеві присвоюється унікальний номер читацького квитка.

Кожен читач може одночасно тримати на руках не більше п'яти книг. Читач не повинен одночасно тримати більше одного примірника книги однієї назви.

Кожна книга в бібліотеці є в декількох примірниках. Кожен екземпляр має такі характеристики:

- унікальний інвентарний номер;
- шифр книги, який збігається з унікальним шифром з опису книг;

– місце розміщення в бібліотеці.

При видачі книги на руки читачеві в бібліотеці залишається спеціальний вкладиш, в якому вказано:

– номер квитка читача, що взяв книгу;
– дату видачі книги;
– дату повернення книги.

Передбачити такі обмеження на інформацію в системі:

1. Книга може бути без єдиного автора.
2. У бібліотеку можуть бути записані читачі не молодше 17 років.
3. У бібліотеці знаходяться книги, видані з 1960 року по поточний рік.
4. Кожному читачеві видається на руки не більше п'яти книг.
5. Кожен читач при реєстрації в бібліотеці повинен залишити для зв'язку номер свого телефону (робочий або домашній).
6. Кожна галузь знань може містити посилання на безліч книг і кожна книга може належати до різних областей знань.

З цією інформаційною системою повинні працювати такі групи користувачів:

– бібліотекарі;
– читачі;
– адміністрація бібліотеки.

Можливості бібліотекаря

При роботі з системою **бібліотекар** повинен:

1. Приймати нові книги і реєструвати їх у бібліотеці.
2. Відносити книги до однієї або кількох галузей знань.
3. Проводити каталогізацію книг, тобто призначати нові інвентарні номери щойно прийнятим книгам, і розміщувати їх на полицях бібліотеки, запам'ятовуючи місце розміщення кожного примірника.
4. Проводити додаткову каталогізацію, якщо надійшло кілька примірників книги, яка вже є в бібліотеці, при цьому інформація про книгу в предметний каталог не вноситься, а кожному новому примірнику присвоюється новий інвентарний номер і для нього визначається місце на полиці бібліотеки.
5. Проводити списання старих і книг, які не користуються попитом. Списувати книги можна за умови, що жоден екземпляр не знаходиться у читачів. Списання проводиться за спеціальним актом, який затверджується адміністрацією бібліотеки.
6. Вести облік виданих читачам книг, при цьому передбачається два режими роботи: видача книг читачеві й приймання книг, які повертаються в бібліотеку. При цьому фіксується дата видачі й те, який екземпляр книги було видано певному читачеві, а також до якого терміну читач повинен повернути цей екземпляр книги.

Наявність вільного примірника книги і його конкретний номер можуть визначатися за заданим унікальним шифром цієї книги або ж інвентарний номер може бути відомий заздалегідь. Не потрібно вести "історію" читання книг, тобто слід відображати тільки поточний стан бібліотеки. Інвентарний номер книги, що повертається, повинен відповідати виданому номеру. Повернута книга займає своє колишнє місце на полиці бібліотеки.

7. Списувати загублені читачем книги за спеціальними актами списання або заміни, підписаними адміністрацією бібліотеки.
8. Закривати абонемент читача, тобто видаляти дані про нього, якщо читач хоче виписатися з бібліотеки і не є її боржником (за ним не числиться ні одна бібліотечна книга).

Можливості читача

Читач повинен мати можливість:

1. Переглядати системний каталог, тобто перелік усіх областей знань книг, які є в бібліотеці.
2. За вибраною галуззю знань отримати повний перелік книг, які є у наявності в бібліотеці.
3. Для вибраної книги отримати інвентарний номер її вільного примірника або повідомлення про те, що вільних примірників книги немає. У разі відсутності вільних примірників книги читач повинен мати можливість дізнатися дату найближчого передбачуваного повернення примірника цієї книги. Читач не може отримати дані про те, у кого в цей момент знаходяться на руках примірники цієї книги.
4. Для вказаного автора отримати список його книг, які є в бібліотеці.

Можливості адміністрації бібліотеки

Адміністрація бібліотеки повинна мати можливість отримувати:

– відомості щодо читачів-боржників бібліотеки, які не повернули в строк узяті книги;

– відомості щодо книг, які не є популярними, тобто жоден примірник яких не знаходиться на руках у читачів;

– відомості щодо вартості конкретної книги, для того щоб відшкодувати вартість втраченої книги або замінити її іншою книгою;

– відомості про найбільш популярні книги, тобто такі, усі примірники яких знаходяться на руках у читачів.

Таким чином, зовсім невеличкий приклад показує, що перед початком розроблення інформаційної системи з базою даних необхідно мати точне уявлення про те, що ж має виконуватися в системі, які користувачі в ній будуть працювати, які завдання буде вирішувати кожен користувач. І це є правильним, адже перш ніж побудувати будинок,

заздалегідь визначають, для яких цілей його призначено, в якому кліматі він буде стояти, на якому ґрунті, і залежно від цього проектувальники пропонують той чи інший проект. Але, на жаль, дуже часто вважають, що все можна визначити потім, коли проект системи вже створено. Відсутність чітких цілей створення БД може звести нанівець всі зусилля розробників, і проект БД буде "поганим", незручним, таким, що не відповідає ні реально модельованому об'єкту, ні завданням, які вирішуватимуться з використанням цієї БД.

4.2. Побудова ER-моделі предметної області

Як приклад спроектуємо концептуальну модель бази даних у вигляді ER-моделі, призначену для зберігання інформації про книги й області знань, що є в бібліотеці (предметну область було описано у попередньому розділі). Розроблення моделі почнемо з виділення основних сутностей і зв'язків між ними.

Отже, можна виділити сутність КНИГА. Що характеризує кожну книгу? - Назва книги, автор, рік видання, кількість сторінок, номер ISBN. Оскільки номер ISBN є унікальним для кожної книги, то саме цей атрибут буде ключем сутності КНИГА. Ну а як же видавництво? Адже кожна книга має видавництво і місце видання.

Уведемо сутність ВИДАВНИЦТВО з такими атрибутами: код видавництва, назва видавництва і місце видавництва. Ключовим атрибутом у цій сутності буде код видавництва.

Між сутностями ВИДАВНИЦТВО і КНИГА є зв'язок "один до багатьох" (1:N), тому що в одному конкретному видавництві може бути видано багато книг, але книга з деяким ISBN друкується в одному конкретному видавництві.

Зауважимо, що кожен екземпляр сутності КНИГА відповідає не конкретній книжці, яка стоїть на полиці, а опису деякої книги, який дається зазвичай у предметному каталозі бібліотеки. Кожна книга може бути в декількох примірниках, і це як раз ті конкретні книги, які стоять на полицях бібліотеки. Для того щоб відобразити це, слід увести сутність ПРИМІРНИК КНИГИ, яка буде містити описи всіх примірників книг, які зберігаються в бібліотеці. Кожен екземпляр сутності ПРИМІРНИК КНИГИ відповідає конкретній книзі на полиці. Що характеризує кожну конкретну книгу в бібліотеці, яка стоїть на полиці? – Інвентарний номер, наявність у бібліотеці (книга може знаходитися або в бібліотеці, або на руках у читача), а в разі, якщо книгу видано, – дата видачі книги читачеві й дата передбачуваного її повернення. Оскільки інвентарний номер книги є унікальним, то саме цей атрибут і буде ключем сутності ПРИМІРНИК КНИГИ.

Між сутностями КНИГИ і ПРИМІРНИК КНИГИ існує зв'язок "один до багатьох" (1:N), обов'язковий з обох кінців.

Опишемо сутність ЧИТАЧ. Визначимо набір атрибутів для сутності ЧИТАЧ, які характеризують її екземпляри. Екземпляром сутності ЧИТАЧ буде конкретний читач бібліотеки.

Щоб сформувати набір атрибутів сутності ЧИТАЧ, необхідно визначити, що характеризує конкретного читача бібліотеки: прізвище, ім'я, по батькові, дата народження, домашня адреса, мобільний телефон, місце роботи, посада, робочий телефон.

Опишемо сутність ЧИТАЦЬКИЙ КВИТОК. Визначимо набір атрибутів для сутності ЧИТАЦЬКИЙ КВИТОК, які характеризують її екземпляри. Екземпляром цієї сутності буде виписаний читацький квиток.

Що характеризує конкретний читацький квиток? – Номер читацького квитка, дата, коли було виписано цей квиток. Оскільки номер читацького квитка є унікальним, то саме цей атрибут і буде ключем сутності ЧИТАЦЬКИЙ КВИТОК.

Оскільки один читач має тільки один читацький квиток у деяку бібліотеку і конкретний квиток може належати тільки одному читачеві, то між сутностями ЧИТАЧ і ЧИТАЦЬКИЙ КВИТОК є зв'язок "один до одного" (1:1).

З опису предметної області відомо, що кожен читач може тримати на руках декілька книг. Таким чином, між сутностями ЧИТАЧ і ПРИМІРНИК КНИГИ існує зв'язок "один до багатьох" (1:N). Але чому зв'язок установлюється не між сутностями ЧИТАЧ і КНИГА? Тому що читач бере з бібліотеки конкретний примірник книги, а не просто книгу. Водночас один і той же примірник книги можуть читати багато читачів, тому між сутностями ЧИТАЧ і ПРИМІРНИК КНИГИ існує зв'язок "один до багатьох" (1:N) і в іншому напрямку. Таким чином, між сутностями ЧИТАЧ і ПРИМІРНИК КНИГИ існує зв'язок "багато до багатьох" (N:M). Ця залежність є необов'язковою з обох кінців, тому що читач у цей момент може не мати жодної книги на руках, а з іншого боку, цей примірник книги може не перебувати ні в кого, а просто стояти на полиці в бібліотеці.

Розглянемо останню сутність в цьому прикладі, яка буде відповідати системному каталогу в бібліотеці, з якого зазвичай починається пошук книг, якщо автор або назва книги є невідомими. Введемо сутність СИСТЕМНИЙ КАТАЛОГ. Визначимо набір атрибутів для сутності СИСТЕМНИЙ КАТАЛОГ, які характеризують її екземпляри. Екземпляром сутності СИСТЕМНИЙ КАТАЛОГ буде системний каталог певної бібліотеки.

Що характеризує системний каталог деякої бібліотеки? – Код області знань, назва галузі знань. Атрибут "код області знань" буде ключовим атрибутом сутності.

З опису предметної області відомо, що кожна книга може містити відомості з кількох областей знань, а з іншого боку, в бібліотеці може бути багато книг, що належать до однієї і тієї ж галузі знань, тому між сутностями СИСТЕМНИЙ КАТАЛОГ і КНИГИ має бути встановлений

зв'язок "багато до багатьох" (N:M), обов'язковий з обох кінців. Дійсно, в системному каталозі не має бути таких областей знань, відомості про які не наведено ні в одній книзі бібліотеки, інакше це є безглуздим. І навпаки, кожна книга має бути віднесена до однієї або декількох галузей знань для того, щоб читач міг її швидше знайти.

Концептуальну модель предметної області "Бібліотека" подано на рис. 4.1.

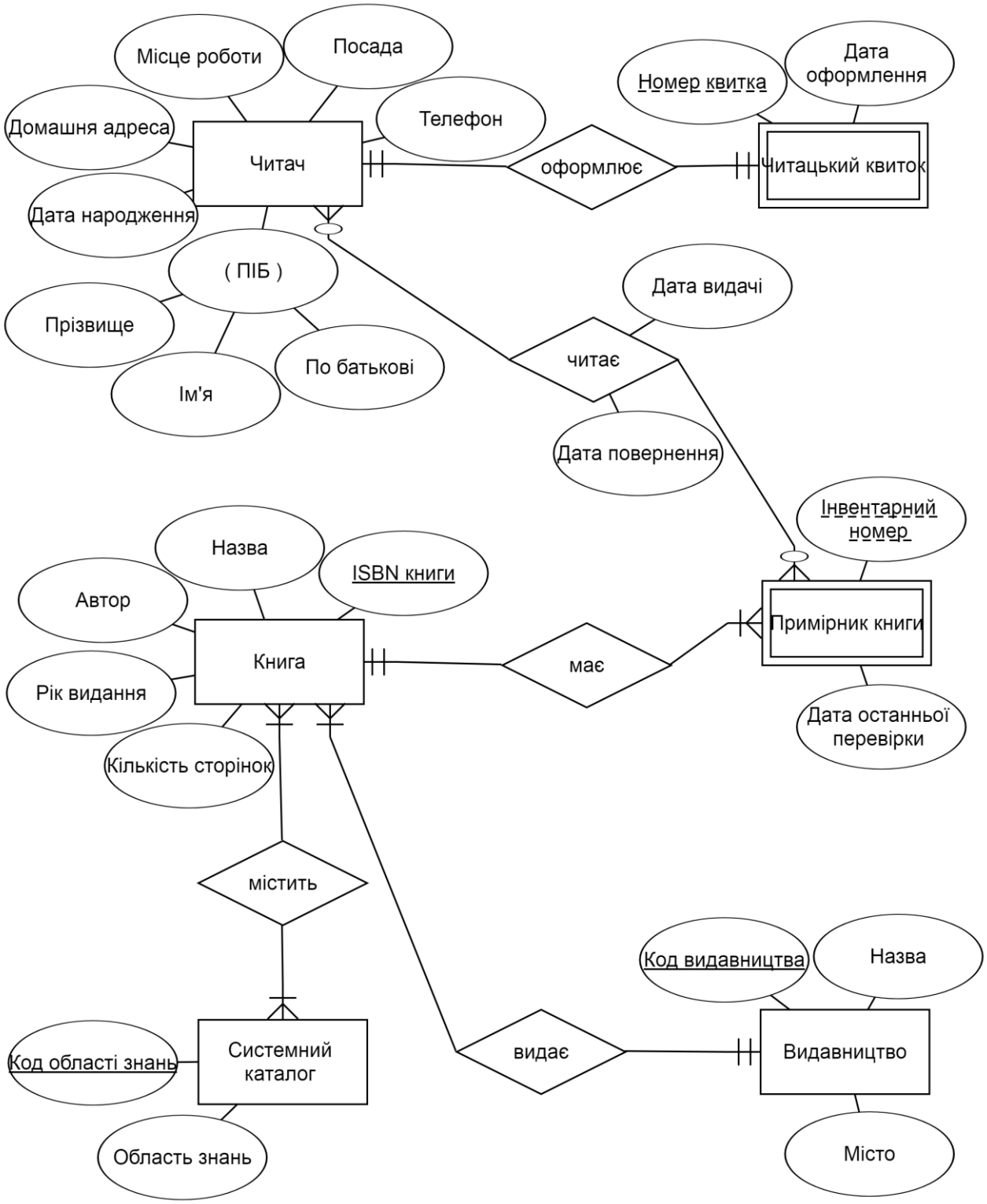


Рис. 4.1. ER-модель предметної області "Бібліотека" в нотації Мартіна

4.3. Логічне проектування бази даних

Наступним етапом проектування баз даних для бібліотеки є перетворення побудованої в попередньому розділі ER-моделі предметної області. Для отримання набору таблиць бази даних необхідно скористатися правилами перетворення, описаними в підрозд. 2.4.1 і 2.4.2.

Спочатку кожній сутності ставимо у відповідність таблицю з такою ж назвою. Кожному атрибуту сутності в таблиці буде відповідати стовпець з такою ж назвою. Отримуємо такий набір таблиць: "Читач", "Читацький квиток", "Книга", "Примірник книги", "Системний каталог", "Видавництво" (рис. 4.2).

Рис. 4.2. Набір реляційних таблиць для предметної області "Бібліотека"

Розглянемо більш детально реалізацію зв'язків між сутностями ER-моделі. Прокоментуємо отримання набору таблиць із зазначенням усіх стовпців, первинних і зовнішніх ключів (стовпців, що мігрують) (рис. 4.3).

Зв'язок "оформлює" між сутностями "Читач" і "Читацький квиток" має тип "1:1" і є обов'язковим з обох кінців. Це означає, що немає необхідності в таблиці "Читацький квиток", можна всі стовпці цієї таблиці перенести в таблицю "Читач" і первинним ключем таблиці "Читач" зробити стовпець "Номер квитка".

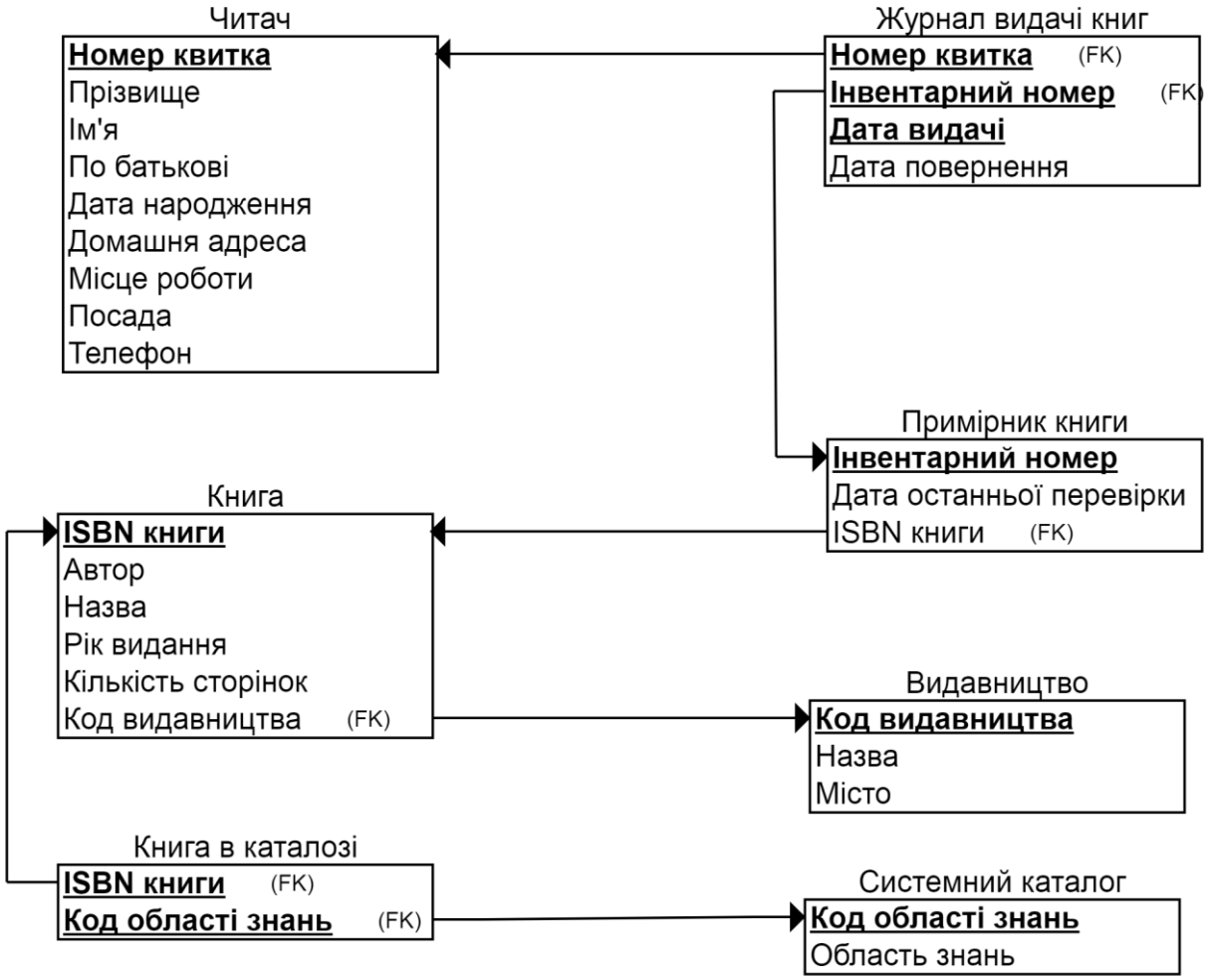


Рис. 4.3. Логічна модель бази даних "Бібліотека"

Зв'язок "читає" між сутностями "Читач" і "Примірник книги" має тип "N:M". Згідно з правилами перетворення зв'язків будь-який зв'язок типу "N:M" має бути поданий додатковою (асоціативною) таблицею. Назвемо цю таблицю "Журнал видачі книг". До неї будуть мігрувати (переміщуватися) первинні ключі таблиць "Читач" і "Примірник книги". Виходячи із рис. 4.1, до таблиці "Журнал видачі книг" необхідно додати стовпці "Дата видачі" й "Дата повернення". Первинним ключем асоціативної таблиці буде комбінація з трьох стовпців ("Номер квитка", "Інвентарний номер", "Дата видачі"). Чому ми вибрали такий складний первинний ключ? В іншому випадку один і той же читач не зможе взяти в бібліотеці більше одного разу книгу з однаковим інвентарним номером.

Зв'язок "має" між сутностями "Книга" та "Примірник книги" має тип "1:N" і є обов'язковим з обох кінців. Згідно з правилами перетворення зв'язків до таблиці "Примірник книги" має мігрувати первинний ключ таблиці "Книга". Первинним ключем в таблиці "Примірник книги" залишається стовпець "Інвентарний номер", оскільки він однозначно ідентифікує кожен примірник книги в бібліотеці.

Зв'язок "видає" між сутностями "Книга" і "Видавництво" також має тип "1:N" і є обов'язковим з обох кінців. Це означає, що первинний ключ таблиці "Видавництво" мігрує до таблиці "Книга". Первинним ключем у таблиці "Книга" залишається стовпець "ISBN книги", оскільки він однозначно ідентифікує кожну книгу.

Зв'язок "містить" між сутностями "Книга" і "Системний каталог" має тип "N:M". Згідно з правилами перетворення зв'язків будь-який зв'язок типу "N:M" має бути поданий додатковою (асоціативною) таблицею. Назвемо її "Книга в каталозі". До цієї таблиці будуть мігрувати первинні ключі таблиць "Книга" і "Системний каталог". Виходячи із рис. 4.1, до таблиці "Книга в каталозі" додаткові стовпці додавати не треба (зв'язок "містить" не має власних атрибутів). Первинним ключем асоціативної таблиці буде комбінація з двох стовпців ("ISBN книги", "Код області знань").

Опишемо отримані таблиці. Зазначимо, що первинні ключі (PRIMARY KEY), які мігрували до інших таблиць, будуть зовнішніми ключами (FOREIGN KEY) у нових таблицях:

Таблиця "**Читач"**

Атрибути	*Тип*	*Опис*
Номер квитка	Ціле число	Первинний ключ (5 цифр)
Прізвище	Рядок	Максимальна довжина – 20 символів
Ім'я	Рядок	Максимальна довжина – 10 символів
По батькові	Рядок	Максимальна довжина – 20 символів
Дата народження	Дата	
Домашня адреса	Рядок	Максимальна довжина – 30 символів
Місце роботи	Рядок	Максимальна довжина – 20 символів
Посада	Рядок	Максимальна довжина – 20 символів
Телефон	Рядок	Максимальна довжина – 20 символів

Таблиця "**Книга"**

Атрибути	*Тип*	*Опис*
ISBN книги	Рядок	Первинний ключ (максимальна довжина – 15 символів)
Автор	Рядок	Максимальна довжина – 20 символів
Назва	Рядок	Максимальна довжина – 30 символів
Рік видання	Рядок	Максимальна довжина – 4 символа
Кількість сторінок	Ціле число	
Код видавництва	Ціле число	Зовнішній ключ

Таблиця **"Примірник книги"**

Атрибути	*Тип*	*Опис*
Інвентарний номер	Ціле число	Первинний ключ
Дата останньої перевірки	Дата	
ISBN книги	Рядок	Зовнішній ключ

Таблиця **"Журнал видачі книг"**

Атрибути	*Тип*	*Опис*	
Номер квитка	Ціле число	Зовнішній ключ	Первинний ключ (складений)
Інвентарний номер	Ціле число	Зовнішній ключ	
Дата видачі	Дата	Значення за замовчуванням: поточна дата	
Дата повернення	Дата		

Таблиця **"Видавництво"**

Атрибути	*Тип*	*Опис*
Код видавництва	Ціле число	Первинний ключ
Назва	Рядок	Максимальна довжина – 50 символів
Місто	Рядок	Максимальна довжина – 50 символів

Таблиця **"Системний каталог"**

Атрибути	*Тип*	*Опис*
Код області знань	Ціле число	Первинний ключ
Область знань	Рядок	Максимальна довжина – 20 символів

Таблиця **"Книга в каталозі"**

Атрибути	*Тип*	*Опис*	
ISBN книги	Рядок	Зовнішній ключ	Первинний ключ (складений)
Код області знань	Ціле число	Зовнішній ключ	

4.4. Фізичне проектування бази даних

Згідно з детальним планом проектування баз даних одним із етапів фізичного проектування бази даних є розроблення SQL-скриптів побудови таблиць. Нижче наведено SQL-скрипти побудови таблиць "Книга", "Читач", "Примірник книги", "Системний каталог", "Книга в каталозі", "Журнал видачі книг" для СУБД Oracle.

```
CREATE TABLE Книга
(
  ISBN_книги VARCHAR(15) NOT NULL,
  Автор VARCHAR(20) NOT NULL,
  Назва VARCHAR(30) NOT NULL,
  Рік_видання VARCHAR(4),
  Кількість_сторінок INT NOT NULL,
  PRIMARY KEY (ISBN_книги));
```

```
CREATE TABLE Читач
(
 Читацький_квиток INT NOT NULL,
 Прізвище VARCHAR(20) NOT NULL,
  Ім'я VARCHAR(10) NOT NULL,
  По батькові VARCHAR(20) NOT NULL,
  Дата_народження DATE NOT NULL,
  Домашня_адреса VARCHAR(30) NOT NULL,
  Місце_роботи VARCHAR(20) NOT NULL,
  Посада VARCHAR(20) NOT NULL,
  Телефон VARCHAR(10) NOT NULL,
   PRIMARY KEY (Читацький_квиток));

CREATE TABLE Примірник_книги
(
  Інвентарний_номер INT NOT NULL,
  Дата_останньої_перевірки DATE NOT NULL,
  ISBN_книги VARCHAR(15) NOT NULL,
  PRIMARY KEY (Інвентарний_номер),
  FOREIGN KEY (ISBN_книги) REFERENCES Книга (ISBN_книги));

CREATE TABLE Системний_каталог
(
  Код_області_знань INT NOT NULL,
  Область_знань VARCHAR(20) NOT NULL,
  PRIMARY KEY (Код_області_знань));

CREATE TABLE Книга_в_каталозі
(
  ISBN_книги VARCHAR(15) NOT NULL,
  Код_області_знань INT NOT NULL,
  PRIMARY KEY (ISBN_книги, Код_області_знань),
  FOREIGN KEY (ISBN_книги) REFERENCES Книга (ISBN_книги),
  FOREIGN KEY (Код_області_знань) REFERENCES Системний_каталог
(Код_області_знань));

CREATE TABLE Журнал_видачі_книг
(
 Читацький_квиток INT NOT NULL,
  Інвентарний_номер INT NOT NULL,
  Дата_видачі DATE NOT NULL DEFAULT SYSDATE,
  Дата_повернення DATE,
  PRIMARY KEY (Читацький_квиток, Інвентарний_номер),
  FOREIGN KEY (Читацький_квиток) REFERENCES Читач (Читацький_квиток),
  FOREIGN KEY (Інвентарний_номер) REFERENCES Примірник_книги
(Інвентарний_номер));
```

Зазначимо, що на цьому процес фізичного проектування бази даних не завершується. Потрібно виконати всі етапи, визначені у детальному плані в розд. 3.

5. ІНШІ ПРИКЛАДИ ПРОЕКТУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

5.1. Проектування бази даних "Інтернет-магазин"

Нижче наведено розроблену ER-модель для предметної області "Інтернет-магазин" і результат перетворення її на логічну модель бази даних.

На етапі побудови ER-моделі було виділено чотири сутності: "Користувач", "Замовлення", "Група товарів" і "Товар" (рис. 5.1).

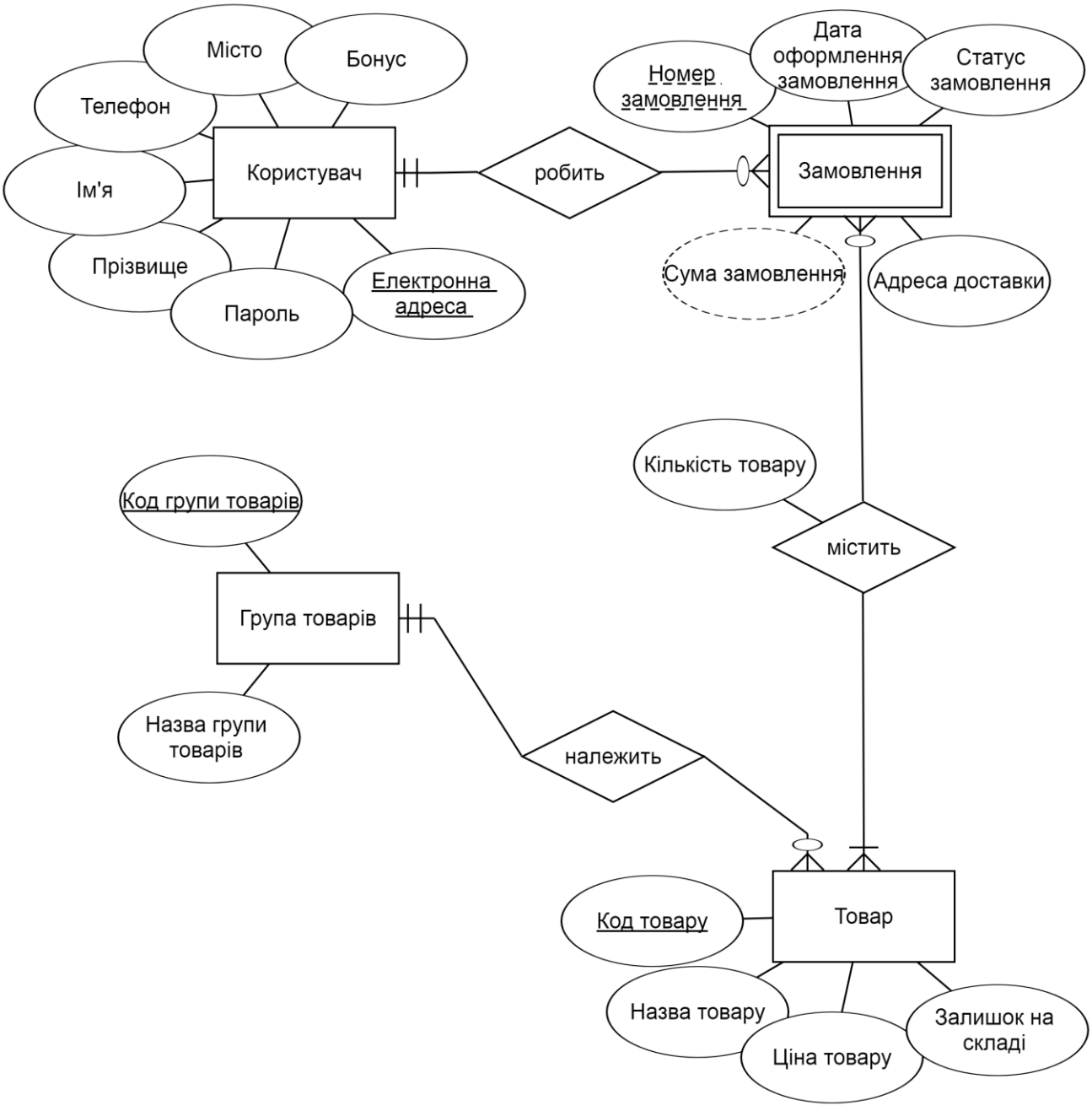


Рис. 5.1. ER-модель предметної області "Інтернет-магазин"

У логічній моделі бази даних є п'ять таблиць (рис. 5.2). Додаткову таблицю "Вміст замовлення" було створено внаслідок реалізації зв'язку типу "N:M" між сутностями "Замовлення" і "Товар". Первинним ключем додаткової таблиці "Вміст замовлення" є два стовпці: стовпець"Номер замовлення", який мігрував з таблиці "Замовлення", і стовпець "Код товару", який мігрував з таблиці "Товар". Такий складений первинний ключ потрібен для того, щоб мати можливість зберігати інформацію про різні товари в одному замовленні.

Рис. 5.2. Логічна модель бази даних "Інтернет-магазин"

Виходячи із рис. 5.1, до таблиці "Вміст замовлення" необхідно додати стовпець "Кількість одиниць" (зв'язок "містить" між сутностями "Замовлення" і "Товар" має атрибут "Кількість товару").

Зв'язок "робить" між сутностями "Користувач" і "Замовлення" має тип "1:N": один "Користувач" може мати нуль або декілька "Замовлень", тоді як одне "Замовлення" належить тільки одному "Користувачу". Зв'язок є обов'язковим з одного кінця: "Замовлення" повинне обов'язково мати свого власника – "Користувача", а ось "Користувач" може ще не мати оформлених "Замовлень". Зв'язок "робить" буде реалізовано через міграцію до таблиці "Замовлення" первинного ключа таблиці "Користувач".

Зв'язок "належить" між сутностями "Група товарів" і "Товар" має тип "1:N": одна "Група товарів" має нуль або декілька "Товарів", тоді як один "Товар" може належати тільки одній "Групі товарів". Зв'язок є обов'язковим з одного кінця: "Товар" повинен обов'язково мати "Групу товарів", тоді як може бути "Група товарів", яка ще не містить жодного "Товару". Зв'язок "належить" буде реалізовано через міграцію до таблиці "Товар" первинного ключа таблиці "Група товарів".

Нижче описано отримані таблиці. Зазначимо, що первинні ключі (PRIMARY KEY), які мігрували до інших таблиць, будуть зовнішніми ключами (FOREIGN KEY) у нових таблицях:

Таблиця **"Користувач"**

Атрибути	*Тип*	*Опис*
Електронна адреса	Рядок	Первинний ключ Максимальна довжина – 30 символів
Пароль	Рядок	Максимальна довжина – 10 символів
Прізвище	Рядок	Максимальна довжина – 20 символів
Ім'я	Рядок	Максимальна довжина – 10 символів
Телефон	Рядок	Максимальна довжина – 10 символів
Місто	Рядок	Максимальна довжина – 15 символів
Бонус	Ціле число	

Таблиця **"Група товарів"**

Атрибути	*Тип*	*Опис*
Код групи товарів	Ціле число	Первинний ключ
Назва групи товарів	Рядок	Максимальна довжина – 20 символів

Таблиця **"Товар"**

Атрибути	*Тип*	*Опис*
Код товара	Ціле число	Первинний ключ
Назва товара	Рядок	Максимальна довжина – 20 символів
Ціна товара	Ціле число	
Залишок на складі	Ціле число	
Код групи товарів	Ціле число	Зовнішній ключ

Таблиця **"Замовлення"**

Атрибути	*Тип*	*Опис*
Номер замовлення	Ціле число	Первинний ключ
Електронна адреса	Рядок	Максимальна довжина – 30 символів
Дата оформлення замовлення	Дата	За замовчуванням – поточна дата
Статус замовлення	Рядок	Перелік значень: "не підтверджений", "підтверджений", "виконаний"
Адреса доставки	Рядок	Максимальна довжина – 30 символів
Сума замовлення	Ціле число	

Таблиця **"Вміст замовлення"**

Атрибути	*Тип*	*Опис*	
Номер замовлення	Ціле число	Зовнішній ключ	Первинний ключ (складений)
Код товару	Ціле число	Зовнішній ключ	
Кількість одиниць	Ціле число		

5.2. Проектування бази даних "Кінотеатр"

Нижче наведено розроблену ER-модель для предметної області "Кінотеатр" і результат перетворення її на логічну модель бази даних.

На етапі побудови ER-моделі було виділено п'ять сутностей: "Відвідувач", "Квиток", "Зал", "Сеанс" і "Фільм" (рис. 5.3).

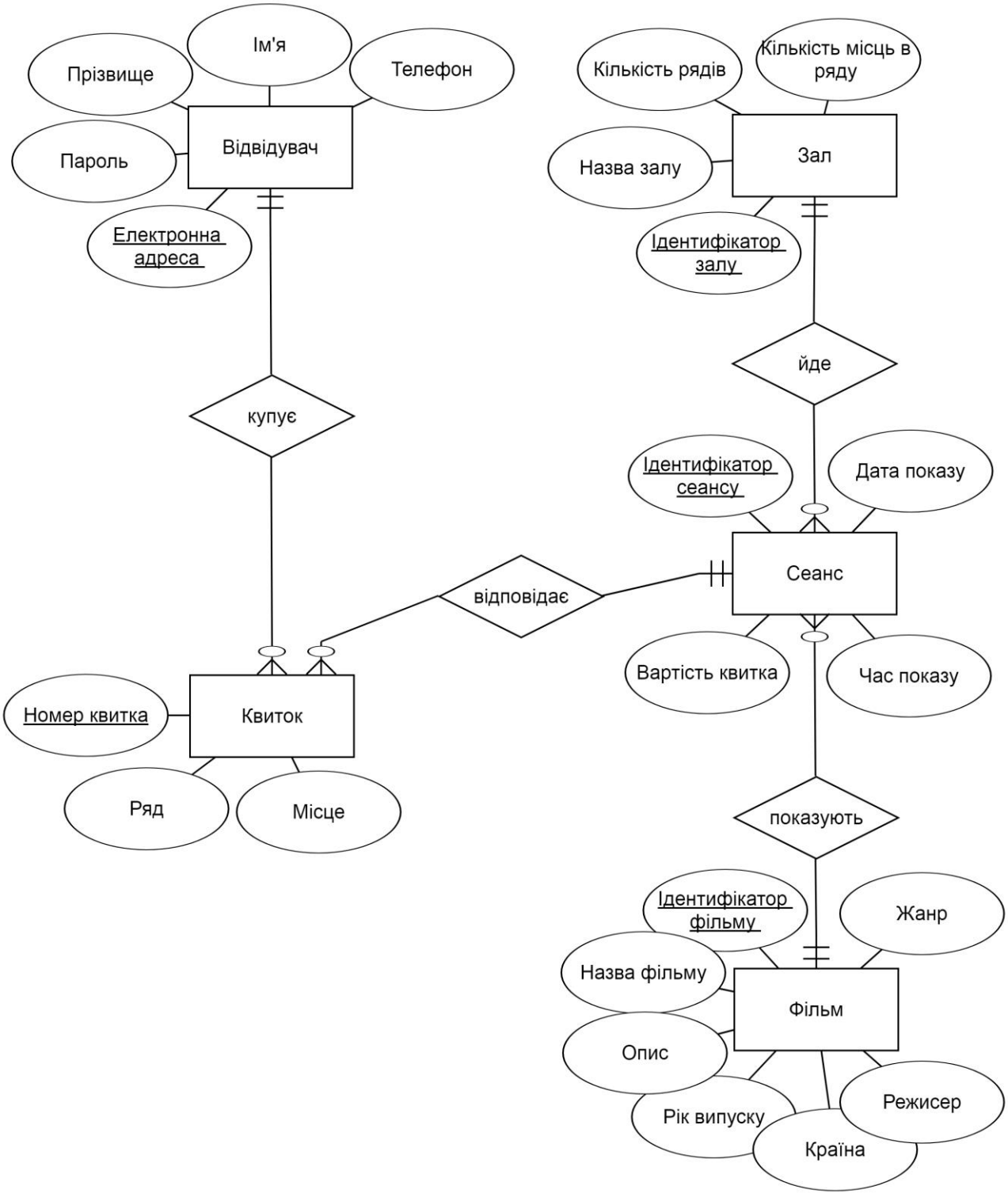


Рис. 5.3. ER-модель предметної області "Кінотеатр"

У логічній моделі бази даних є п'ять таблиць (рис. 5.4). Додаткових таблиць для цієї схеми бази даних створювати не потрібно, тому що спроектована ER-модель не має жодного зв'язку типу "N:M".

Зв'язок "купує" між сутностями "Відвідувач" і "Квиток" має тип "1:N": один "Відвідувач" може купувати нуль або декілька "Квитків", тоді як один "Квиток" належить тільки одному "Відвідувачу". Зв'язок є обов'язковим з одного кінця: "Квиток" обов'язково має свого власника – "Відвідувача", тоді як "Відвідувач" може ще не мати жодного купленого квитка. Цей зв'язок буде реалізовано через міграцію до таблиці "Квиток" первинного ключа таблиці "Відвідувач".

Зв'язок "йде" між сутностями "Зал" і "Сеанс" має тип "1:N": в одному "Залі" можуть проходити нуль або декілька "Сеансів", тоді як один "Сеанс" проходить тільки в одному "Залі". Зв'язок є обов'язковим з одного кінця: "Сеанс" обов'язково йде у деякому "Залі", але "Зал" може не мати закріплених за ним "Сеансів". Цей зв'язок буде реалізовано через міграцію до таблиці "Сеанс" первинного ключа таблиці "Зал".

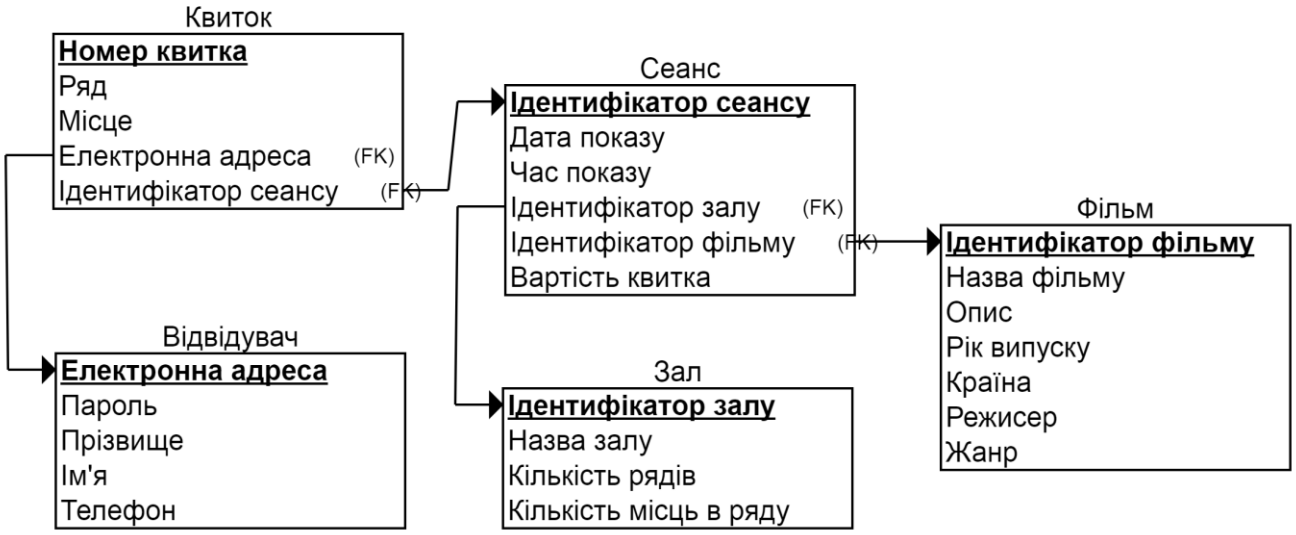


Рис. 5.4. Логічна модель бази даних "Кінотеатр"

Зв'язок "показують" між сутностями "Фільм" і "Сеанс" має також тип "1:N": один і той самий "Фільм" демонструють на нуль або декількох "Сеансах", тоді як на одному "Сеансі" демонструють тільки один "Фільм". Зв'язок є обов'язковим з одного кінця: "Сеанс" обов'язково має відповідний "Фільм", тоді як "Фільм" може ще не мати жодного "Сеансу", на якому би його демонстрували. Цей зв'язок буде реалізовано через міграцію до таблиці "Сеанс" первинного ключа таблиці "Фільми".

І останній зв'язок "відповідає" між сутностями "Сеанс" і "Квиток" також має тип "1:N": на один "Сеанс" може бути куплено нуль або декілька "Квитків", тоді як один "Квиток" відповідає тільки одному "Сеансу". Зв'язок є

обов'язковим з одного кінця: "Квиток" обов'язково має "Сеанс", якому він відповідає, тоді як "Сеанс" може не мати жодного купленого на нього "Квитка". Цей зв'язок буде реалізовано через міграцію до таблиці "Квиток" первинного ключа таблиці "Сеанс".

Нижче описано отримані таблиці. Зазначимо, що первинні ключі (PRIMARY KEY), які мігрували до інших таблиць, будуть зовнішніми ключами (FOREIGN KEY) у нових таблицях:

Таблиця **"Відвідувач"**

Атрибути	*Тип*	*Опис*
Електронна адреса	Рядок	Первинний ключ Максимальна довжина – 20 символів
Пароль	Рядок	Максимальна довжина – 10 символів
Прізвище	Рядок	Максимальна довжина – 20 символів
Ім'я	Рядок	Максимальна довжина – 10 символів
Телефон	Рядок	Максимальна довжина – 10 символів

Таблиця **"Зал"**

Атрибути	*Тип*	*Опис*
Ідентифікатор залу	Ціле число	Первинний ключ
Назва залу	Рядок	Максимальна довжина – 15 символів
Кількість рядів	Ціле число	
Кількість місць в ряду	Ціле число	

Таблиця **"Фільм"**

Атрибути	*Тип*	*Опис*
Ідентифікатор фільму	Ціле число	Первинний ключ
Назва фільму	Рядок	Максимальна довжина – 30 символів
Опис	Рядок	Максимальна довжина – 100 символів
Рік випуску	Ціле число	
Країна	Рядок	Максимальна довжина – 15 символів
Режисер	Рядок	Максимальна довжина – 20 символів
Жанр	Рядок	Перелік значень: "пригоди", "комедія", "фантастика", "мультфільм"

Таблиця **"Сеанс"**

Атрибути	*Тип*	*Опис*
Ідентифікатор фільму	Ціле число	Первинний ключ
Дата показу	Дата	
Час показу	Дата	
Ідентифікатор залу	Ціле число	Зовнішній ключ
Ідентифікатор фільму	Ціле число	Зовнішній ключ
Вартість квитка	Ціле число	

Таблиця **"Квиток"**

Атрибути	*Тип*	*Опис*
Номер квитка	Ціле число	Первинний ключ
Ряд	Ціле число	
Місце	Ціле число	
Електронна адреса	Рядок	Зовнішній ключ
Ідентифікатор сеансу	Ціле число	Зовнішній ключ

5.3. Проектування бази даних "Екзаменаційна успішність студентів"

Нижче наведено розроблену ER-модель для предметної області "Екзаменаційна успішність студентів" і результат перетворення її на логічну модель бази даних.

На етапі побудови ER-моделі було виділено десять сутностей: "Студент", "Група", "Спеціальність", "Залікова книжка", "Дисципліна", "Навчальний план", "Зміст навчального плану", "Викладач", "Кафедра" і "Факультет" (рис. 5.5).

У логічній моделі бази даних є також десять таблиць (рис. 5.6). Додаткових таблиць для цієї схеми бази даних створювати не потрібно, тому що спроектована ER-модель не має жодного зв'язку типу "N:M".

Зв'язок "належить" між сутностями "Спеціальність" і "Група" має тип "1:N": одна "Спеціальність" може мати нуль або декілька "Груп", тоді як одна "Група" може належати тільки до однієї "Спеціальності". Цей зв'язок буде реалізовано через міграцію до таблиці "Групи" первинного ключа таблиці "Спеціальність".

Зв'язок "'є зарахованим" між сутностями "Група" і "Студент" має тип "1:N": одна "Група" може мати нуль або декілька "Студентів", тоді як один "Студент" може бути зарахований тільки до однієї "Групи". Цей зв'язок буде реалізовано через міграцію до таблиці "Студент" первинного ключа таблиці "Групи".

Зв'язок "має" між сутностями "Факультет" і "Кафедра" має тип "1:N": один "Факультет" може мати нуль або декілька "Кафедр", тоді як одна й та сама "Кафедра" може належати тільки до одного "Факультету". Цей зв'язок буде реалізовано через міграцію до таблиці "Кафедри" первинного ключа таблиці "Факультет".

Зв'язок "працює" між сутностями "Кафедра" і "Викладач" має тип "1:N": на одній "Кафедрі" можуть працювати нуль або декілька "Викладачів", тоді як один і той самий викладач може бути закріплений тільки за однією "Кафедрою". Цей зв'язок буде реалізовано через міграцію до таблиці "Викладач" первинного ключа таблиці "Кафедра".

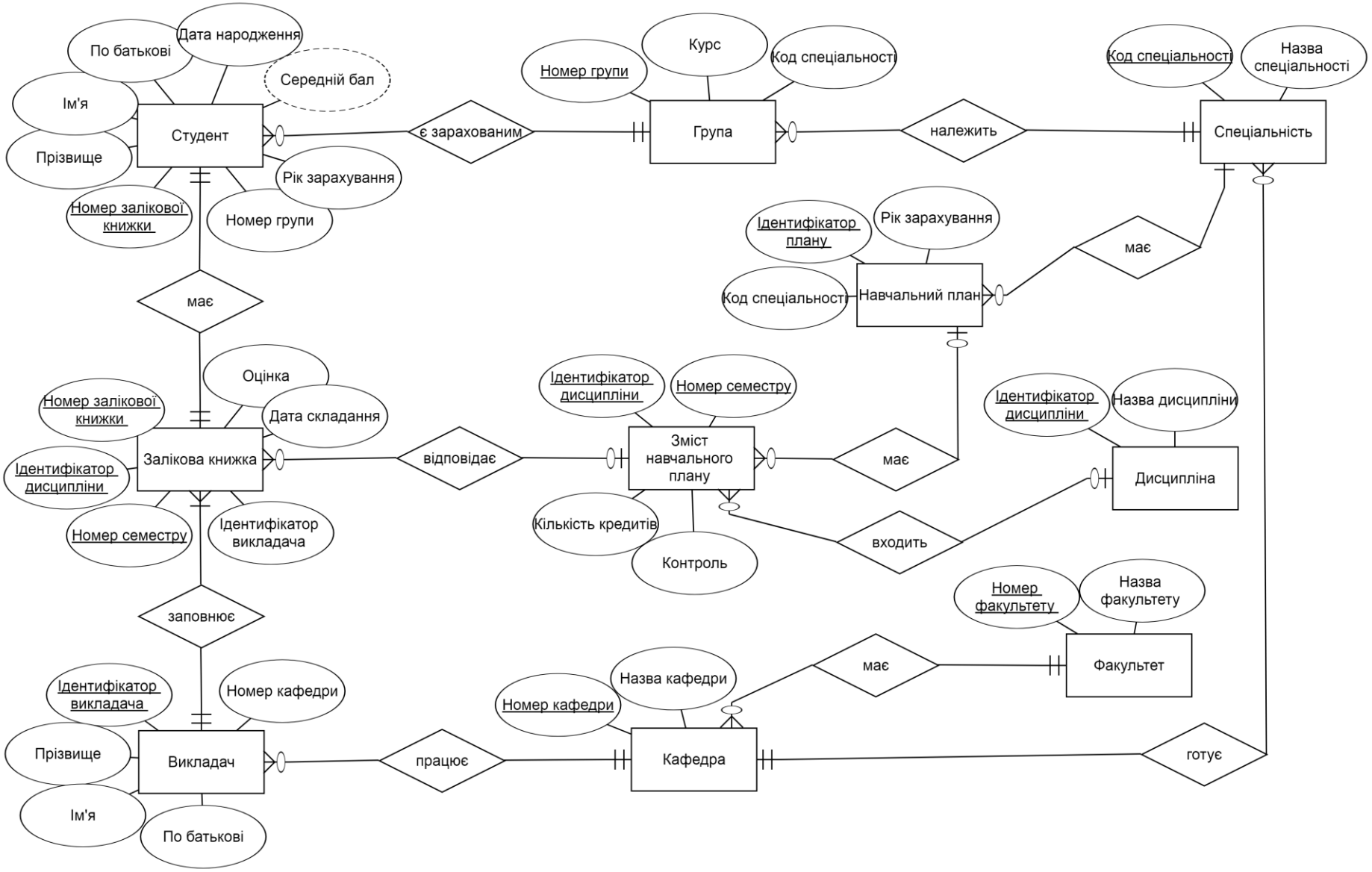


Рис. 5.5. ER-модель предметної області "Екзаменаційна успішність студентів"

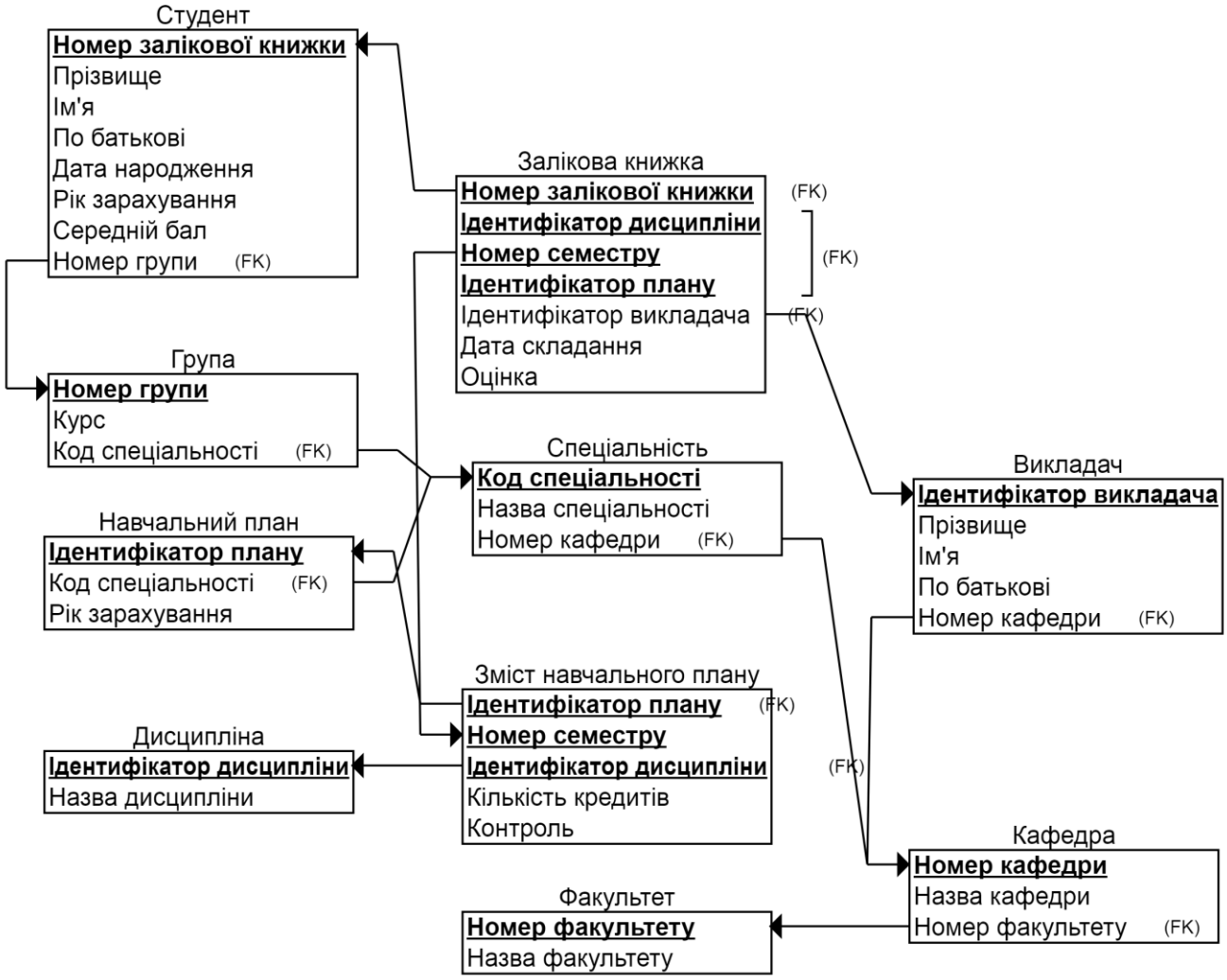


Рис. 5.6. Логічна модель бази даних "Екзаменаційна успішність студентів"

Зв'язок "готує" між сутностями "Кафедра" і "Спеціальність" має тип "1:N": одна "Кафедра" може готувати студентів з нуль або декілька "Спеціальностей", тоді як студентів з однієї й тієї самої "Спеціальності" можуть готувати тільки на одній "Кафедрі". Цей зв'язок буде реалізовано через міграцію до таблиці "Спеціальність" первинного ключа таблиці "Кафедра".

Зв'язок "має" між сутностями "Спеціальність" і "Навчальний план" має тип "1:N": одна "Спеціальність" може мати декілька навчальних планів для різних років набору студентів, тоді як один "Навчальний план" розробляється тільки для однієї "Спеціальності". Цей зв'язок буде реалізовано через міграцію до таблиці "Навчальний план" первинного ключа таблиці "Спеціальність".

Зазначимо, що між сутностями "Навчальний план" і "Дисципліна" насправді є зв'язок типу "N:M": один "Навчальний план" містить нуль або декілька "Дисциплін", тоді як одна "Дисципліна" може входити до нуль або декілька "Навчальних планів". А сутність "Зміст начального плану" є

додатковою, яка допомагає розкрити зв'язок типу "N:M" між сутностями "Навчальний план" і "Дисципліна". Розкриття зв'язку типу "N:M" на етапі розроблення ER-моделі предметної області можна було б не роботи, але онлайн-ресурс https://erdplus.com, за допомогою якого розроблялися всі наведені в цьому розділі ER-моделі, не дає змоги встановити зв'язок між сутністю "Залікова книжка" і сутностями "Навчальний план" і "Дисципліна". "Залікова книжка" є відображенням певного змісту навчального плану на конкретного студента, тобто маємо зв'язок між трьома сутностями: "Залікова книжка", "Навчальний план", "Дисципліна".

Зв'язок "входить" між сутностями "Дисципліна" і "Зміст навчального плану" має тип "1:N": одна й та сама "Дисципліна" може нуль або декілька разів зустрічатися в "Змісті навчального плану". Цей зв'язок буде реалізовано через міграцію до таблиці "Зміст навчального плану" первинного ключа таблиці "Дисципліна".

Зв'язок "має" між сутностями "Навчальний план" і "Зміст навчального плану" має тип "1:N": один і той самий "Навчальний план" може мати нуль або декілька пунктів у "Змісті навчального плану". Цей зв'язок буде реалізовано через міграцію до таблиці "Зміст навчального плану" первинного ключа таблиці "Навчальний план".

Зв'язок "відповідає" між сутностями "Зміст навчального плану" і "Залікова книжка" має тип "1:N": один і той самий "Зміст навчального плану" може бути розкритий у нуль або декількох "Залікових книжках". Цей зв'язок буде реалізовано через міграцію до таблиці "Залікова книжка" первинного ключа таблиці "Зміст навчального плану".

Нарешті, розглянемо останній зв'язок "заповнює" між сутностями "Викладач" і "Залікова книжка". Цей зв'язок має тип "1:N": один "Викладач" заповнює нуль або декілька "Залікових книжок", тоді як один іспит (або залік) у "Заліковій книжці" приймає тільки один "Викладач".

Опишемо отримані таблиці. Зазначимо, що первинні ключі (PRIMARY KEY), які мігрували до інших таблиць, будуть зовнішніми ключами (FOREIGN KEY) у нових таблицях:

Таблиця **"Факультет"**

Атрибути	*Тип*	*Опис*
Номер факультету	Число	Первинний ключ
Назва факультету	Рядок	Максимальна довжина – 30 символів

Таблиця **"Кафедра"**

Атрибути	*Тип*	*Опис*
Номер кафедри	Число	Первинний ключ
Назва кафедри	Рядок	Максимальна довжина – 30 символів
Номер факультету	Число	Зовнішній ключ

Таблиця **"Викладач"**

Атрибути	*Тип*	*Опис*
Ідентифікатор викладача	Число	Первинний ключ
Прізвище	Рядок	Максимальна довжина – 20 символів
Ім'я	Рядок	Максимальна довжина – 10 символів
По батькові	Рядок	Максимальна довжина – 20 символів
Номер кафедри	Число	Зовнішній ключ

Таблиця **"Спеціальність"**

Атрибути	*Тип*	*Опис*
Код спеціальності	Число	Первинний ключ
Назва спеціальності	Рядок	Максимальна довжина – 30 символів
Номер кафедри	Число	Зовнішній ключ

Таблиця **"Група"**

Атрибути	*Тип*	*Опис*
Номер групи	Рядок	Первинний ключ
Курс	Число	Можливі значення – від 1 до 5
Код спеціальності	Число	Зовнішній ключ

Таблиця **"Студент"**

Атрибути	*Тип*	*Опис*
Номер залікової книжки	Рядок	Первинний ключ
Прізвище	Рядок	Максимальна довжина – 20 символів
Ім'я	Рядок	Максимальна довжина – 10 символів
По батькові	Рядок	Максимальна довжина – 20 символів
Дата народження	Дата	
Рік зарахування	Число	Чотири цифри
Середній бал	Число	Дійсне число (два знаки після коми) Можливі значення – від 0 до 100
Номер групи	Рядок	Зовнішній ключ

Таблиця **"Дисципліна"**

Атрибути	*Тип*	*Опис*
Ідентифікатор дисципліни	Число	Первинний ключ
Назва дисципліни	Рядок	Максимальна довжина – 20 символів

Таблиця **"Навчальний план"**

Атрибути	*Тип*	*Опис*
Ідентифікатор плану	Число	Первинний ключ
Код спеціальності	Число	Зовнішній ключ
Рік зарахування	Число	

Таблиця **"Зміст навчального плану"**

Атрибути	*Тип*	*Опис*	
Ідентифікатор плану	Число	Зовнішній ключ	Первинний ключ (складений)
Номер семестру	Число	Можливі значення – від 1 до 10	
Ідентифікатор дисципліни	Число	Зовнішній ключ	
Кількість кредитів	Число	Можливі значення – від 1 до 7	
Контроль	Рядок	Перелік значень: "залік", "диференційний залік", "іспит"	

Таблиця **"Залікова книжка"**

Атрибути	*Тип*	*Опис*	
Номер залікової книжки	Рядок	Зовнішній ключ	Первинний ключ (складений)
Ідентифікатор дисципліни	Число	Зовнішній ключ	
Номер семестру	Число	Можливі значення – від 1 до 10	
Ідентифікатор плану	Число	Зовнішній ключ	
Ідентифікатор викладача	Число	Зовнішній ключ	
Дата складання	Дата		
Оцінка	Число	Можливі значення – від 1 до 100 балів	

6. ПРИКЛАД ПРОЕКТУВАННЯ БАЗИ ДАНИХ У СЕРЕДОВИЩІ ERWIN З ВИКОРИСТАННЯМ НОТАЦІЇ IDEF1X

У цьому розділі розглянуто приклад поетапної побудови бази даних для системи автоматизації розрахунків за послугу "Холодне водопостачання" для комунального підприємства з постачання холодної води.

6.1. Аналіз вимог до бази даних

На рис. 6.1 показано діаграму варіантів використання для системи автоматизації розрахунків за послугу "Холодне водопостачання" для комунального підприємства з постачання холодної води.

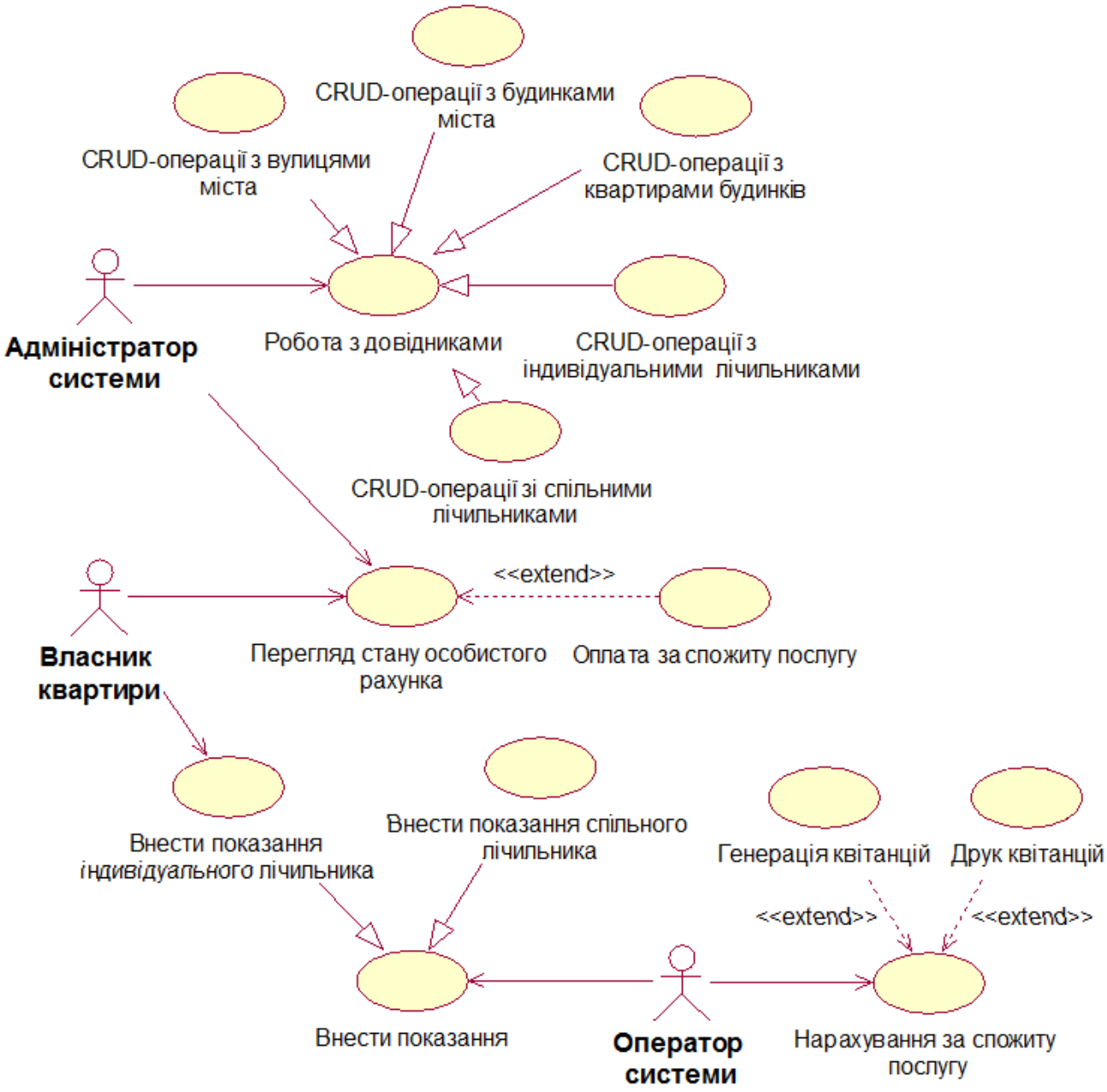


Рис. 6.1. Діаграма варіантів використання системи "Холодне водопостачання"

На попередньому етапі проектування бази даних потрібно проаналізувати варіанти використання розроблюваного програмного забезпечення (ПЗ) з метою виявлення таких варіантів використання, для реалізації яких необхідною буде взаємодія з БД.

Унаслідок проведеного аналізу діаграми варіантів використання було виділено варіанти використання, які потребують взаємодії з БД.

Після аналізу функцій програмного забезпечення, що розробляється, було виділено дані, які необхідно зберігати в БД: назви вулиць, номери будинків, квартири, інформація про їхніх власників, встановлені

індивідуальні й спільні лічильники холодної води, передані показання лічильників холодної води, оплати за спожиту холодну воду.

На основі даних, які будуть зберігатися у системі розрахунків "Холодне водопостачання", відбуватимуться нарахування за місяць за спожиту послугу, а також генеруватимуться квитанції для оплати з подальшим їх друкуванням.

6.2. Концептуальне проектування бази даних

На етапі концептуального проектування БД необхідно на основі проведеного аналізу вимог до ПЗ побудувати модель "сутність – зв'язок" (ER-модель) предметної області.

Для графічного подання ER-моделі слід визначитися з нотацією. Для подання ER-моделі виконання розрахунків за послугу "Холодне водопостачання" було вибрано нотацію IDEF1X, яка підтримується CASE-засобом ERWin Data Modeler [15].

Для побудови ER-моделі потрібно виділити сутності предметної області. Унаслідок аналізу предметної області було виділено такі сутності: "Вулиця", "Будинок", "Квартира", "Спільн_лічильники", "Показання_спільн_ліч", "Індивід_лічильники", "Показання_індив_ліч", "Оплата". Кожна сутність в нотації IDEF1X зображується прямокутником (рис. 6.2).

Рис. 6.2. Сутності предметної області

Після виділення сутностей предметної області необхідно визначити їх атрибути. На рис. 6.3 наведено виділені сутності з визначеними для них атрибутами. Для позначення доменів атрибутів сутності використовують графічну форму їх відображення.

Для кожної сутності визначимо один атрибут або групу атрибутів, які будуть мати унікальні (без повторень) значення й однозначно ідентифікувати кожен екземпляр цієї сутності. На рис. 6.4 показано сутності з ідентифікувальними атрибутами (такі атрибути розміщуються в верхній частині прямокутника, яким позначається сутність).

Рис. 6.3. Атрибути сутностей предметної області

Рис. 6.4. Первинні ключі сутностей предметної області

Наступним етапом побудови ER-моделі є визначення зв'язків між сутностями. На рис. 6.5 зображено виділені сутності з встановленими між ними зв'язками. Сутність, яка знаходиться на кінці зв'язку з жирною крапкою, називається дочірньою, інша сутність зв'язку – батьківською. Наприклад, для зв'язку "на Вулиці знаходяться Будинки", сутність "Вулиця" є батьківською, тоді як сутність "Будинок" – дочірньою.

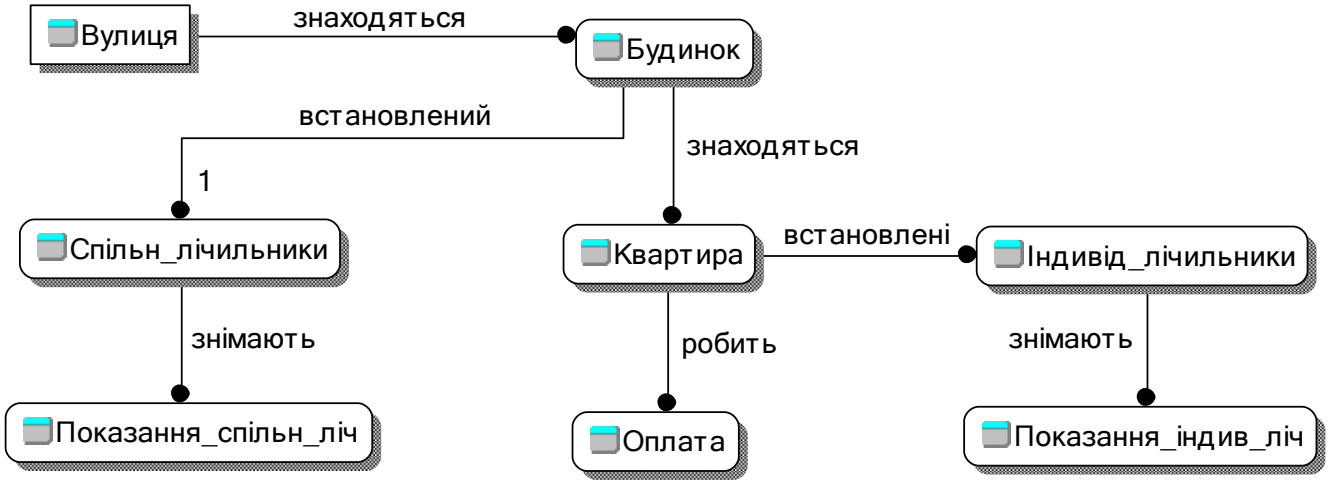


Рис. 6.5. Зв'язки між сутностями предметної області

У нотації IDEF1x виділяють два типи дочірніх сутностей: залежні й незалежні. Для зазначення того, що дочірня сутність буде залежати від батьківської, у нотації IDEF1x використовується *ідентифікувальний зв'язок* (позначається суцільною лінією, в цьому випадку дочірня сутність позначається прямокутником зі закругленими кутами). Для вказання того, що дочірня сутність не буде залежати від батьківської, використовується *неідентифікувальний зв'язок* (позначається пунктирною лінією).

На рис. 6.5 всі зв'язки є ідентифікувальними, але необхідно проаналізувати побудовану ER-модель на предмет наявності залежних і незалежних дочірніх сутностей, тобто змінити там, де це необхідно, ідентифікувальний зв'язок на неідентифікувальний.

Унаслідок проведеного аналізу дочірніх сутностей було отримано змінену ER-модель, показану на рис. 6.6.

6.3. Логічне проектування бази даних

На етапі логічного (даталогічного) проектування БД необхідно на основі побудованої ER-моделі предметної області розробити схему реляційної бази даних, тобто отримати набір реляційних таблиць.

CASE-засіб ERWin автоматично перетворює ER-модель на схему реляційної БД за такими правилами:

– кожній сутності ER-моделі ставиться у відповідність таблиця БД;

– зв'язки "один до одного", "один до багатьох" реалізуються шляхом міграції первинного ключа батьківської сутності в дочірню;

– зв'язок "багато до багатьох" реалізується шляхом створення асоціативної таблиці, до якої мігрують первинні ключі обох сутностей, що знаходяться в зв'язку "багато до багатьох".

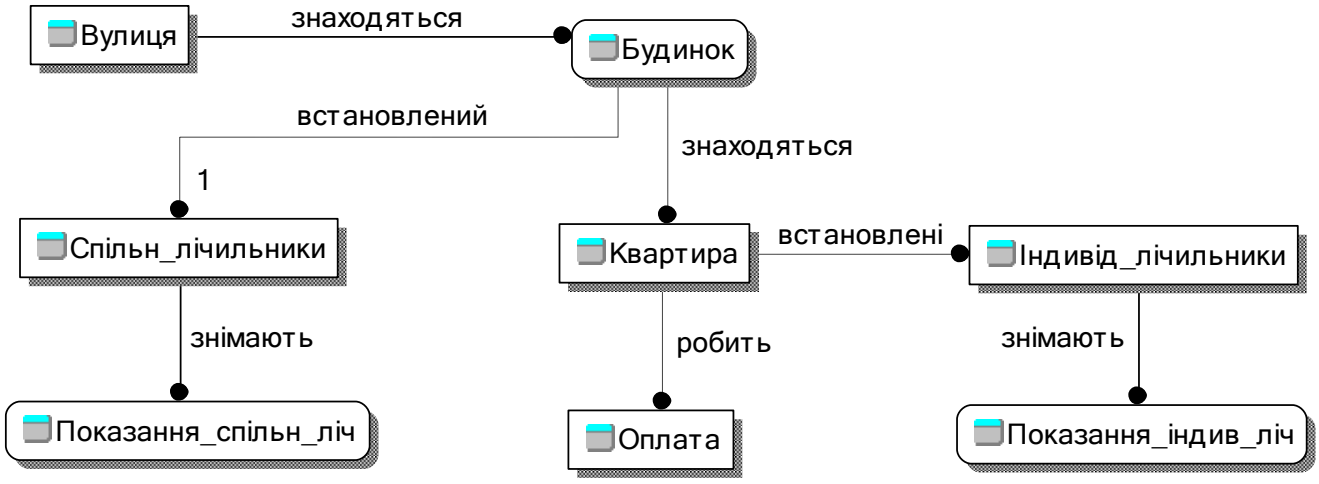


Рис. 6.6. ER-модель предметної області в нотації IDEF1x

Для створення асоціативної таблиці достатньо виділити необхідний зв'язок, а потім натиснути праву кнопку миші й вибрати пункт "Створення асоціативної таблиці".

На рис. 6.7 подано таблиці БД для виконання розрахунків за послугу "Холодне водопостачання", які було отримано автоматично за допомогою ERWin Data Modeler. Кількість таблиць повністю збігається з кількістю сутностей у побудованій ER-моделі, тому що між сутностями немає зв'язків "багато до багатьох".

Бачимо, що отримана в автоматичному режимі схема реляційної БД має недоліки. Наприклад, розглянемо таблицю "Будинок", яка має атрибути Ід_вулиці і Назва_вулиці. Атрибут Назва_вулиці було виділено на етапі побудови ER-моделі, тоді як атрибут Ід_вулиці є атрибутом, який мігрував до таблиці "Будинок" унаслідок реалізації зв'язку "на Вулиці знаходяться Будинки". Оскільки за значенням атрибута Ід_вулиці можна однозначно визначити назву вулиці (для отримання цієї інформації достатньо використати таблицю "Вулиця"), тому видалимо з таблиці "Будинок" атрибут Назв_вулиці.

Зауважимо, що атрибут Назв_вулиці таблиці "Будинок" мігрує в дочірні таблиці, тому його видалення з таблиці "Будинок" призведе до змін у структурі інших таблиць. Унаслідок проведеного аналізу всіх таблиць було отримано схему БД (рис. 6.8).

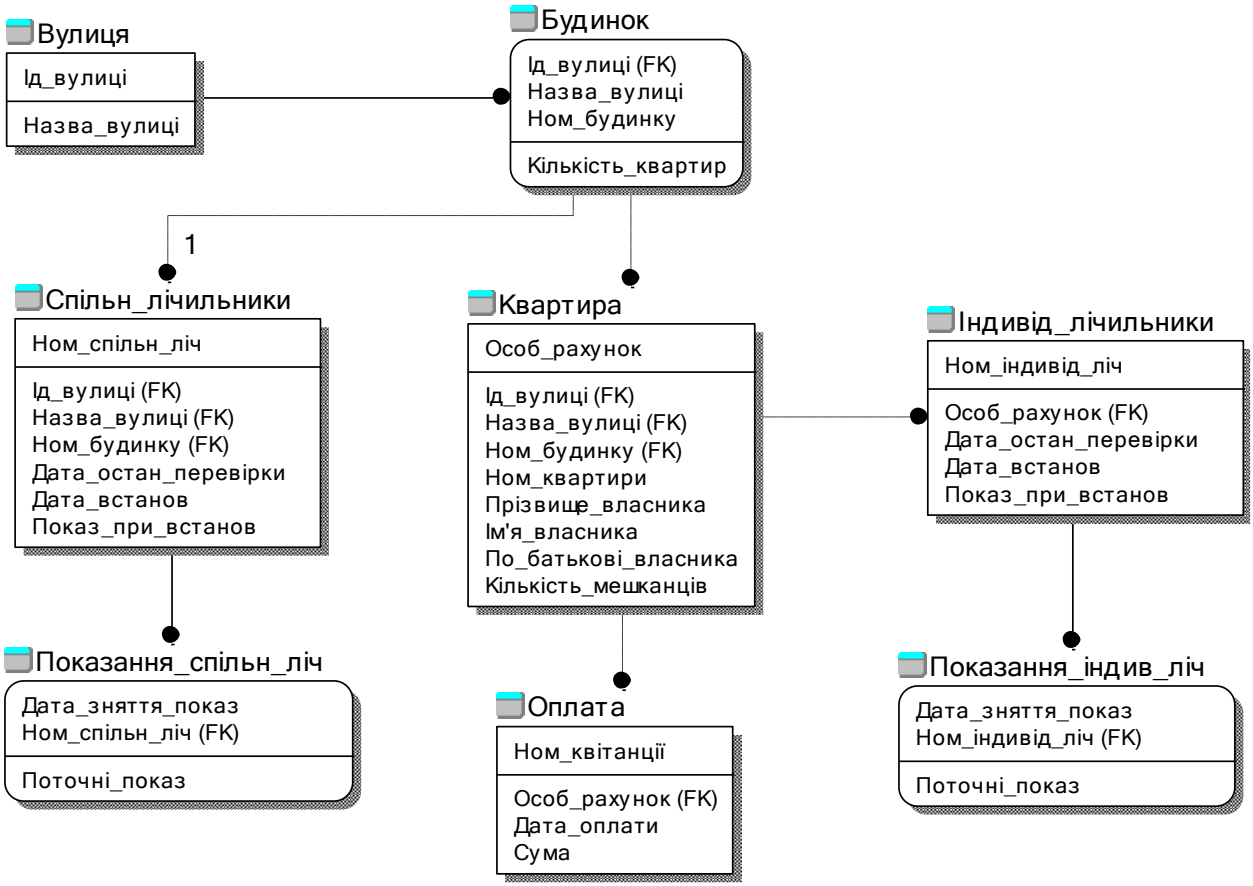


Рис. 6.7. Схема бази даних

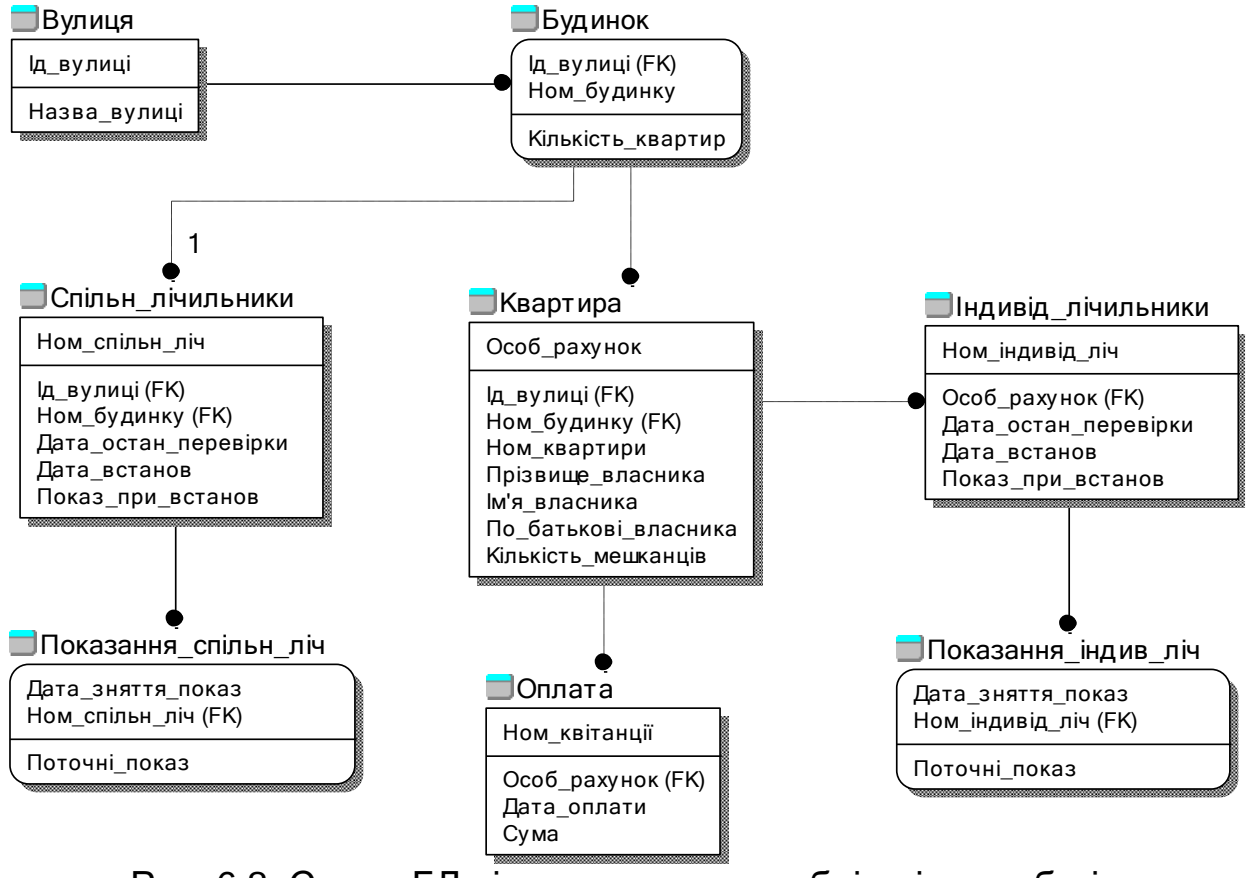


Рис. 6.8. Схема БД після видалення дублікатів атрибутів

Проілюструємо синтезовану схему БД із зазначенням доменів атрибутів таблиць (рис. 6.9).

Синтезовані відношення (таблиці) потребують перевірки за допомогою правил нормалізації. Нормалізація схеми бази даних сприяє більш ефективному виконанню системою управління базами даних операцій оновлення бази даних, оскільки скорочується кількість перевірок і допоміжних дій, що підтримують цілісність бази даних.

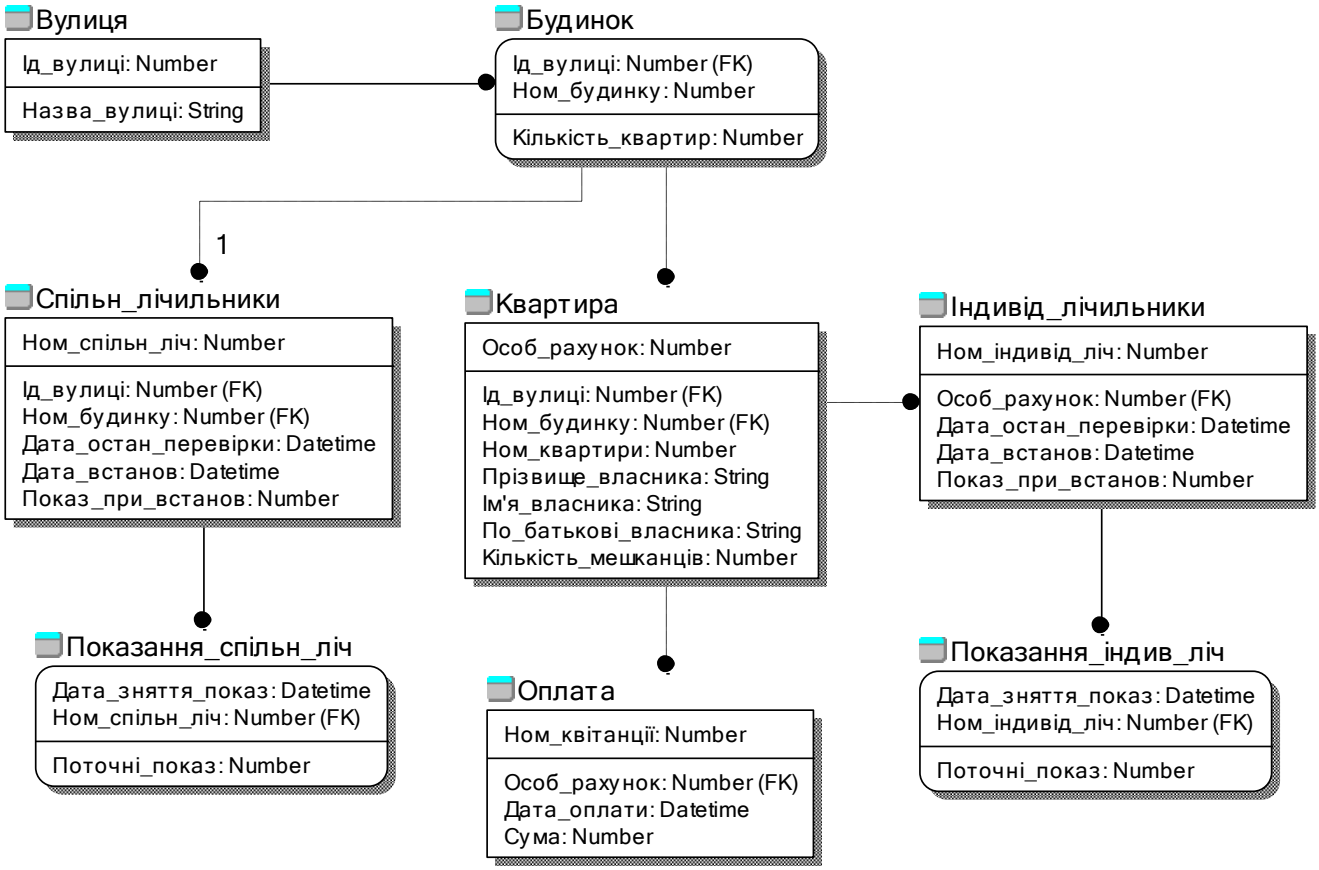


Рис. 6.9. Схема БД із зазначенням доменів атрибутів

Перевіримо побудовані відношення на відповідність нормальним формам. Для цього запишемо відношення в такому вигляді:

Вулиця (Ід_вулиці, Назва_вулиці)

Будинок (Ід_вулиці, Ном_будинку, Кількість_квартир)

Квартира (Особ_рахунок, Ід_вулиці, Ном_будинку, Ном_квартири, Прізвище_власника, Ім'я_власника, По_батькові_власника, Кількість_мешканців)

Спільн_лічильники (Ном_спільн_ліч, Ід_вулиці, Ном_будинку, Дата_остан_перевірки, Дата_встанов, Показ_при_встанов)

Індивід_лічильники (Ном_індивід_ліч, Особ_рахунок, Дата_остан_перевірки, Дата_встанов, Показ_при_встанов)

Показання_спільн_ліч (Дата_зняття_показ, Ном_спільн_ліч, Поточні_показ)

Показання_індив_ліч (Дата_зняття_показ, Ном_індивід_ліч, Поточні_показ)

Оплата (Ном_квітанції, Особ_рахунок, Дата_оплати, Сума)

Наведені відношення знаходяться в першій нормальній формі (1NF), оскільки всі атрибути відношень набувають простих значень (атомарних або неподільних), які не є множиною або кортежем з більш елементарних складових.

Подані відношення знаходяться в другій нормальній формі (2NF), оскільки вони мають першу нормальну форму, і кожен неключовий атрибут конкретного відношення мінімально функціонально залежить від свого первинного ключа.

Наведені відношення знаходяться в третій нормальній формі (3NF), оскільки вони мають другу нормальну форму, і кожен неключовий атрибут конкретного відношення нетранзитивно функціонально залежить від свого первинного ключа.

Опишемо всі таблиці синтезованої схеми реляційної БД для виконання розрахунків за послугу "Холодне водопостачання" для комунального підприємства з постачання холодної води (всі атрибути виділених таблиць схеми БД є обов'язковими):

Таблиця **"Вулиця"**

Атрибути	*Тип*	*Опис*
Ід_вулиці	Ціле число	Первинний ключ
Назв_вулиці	Рядок	Максимальна довжина – 50 символів

Таблиця **"Будинок"**

Атрибути	*Тип*	*Опис*
Ід_вулиці	Ціле число	Первинний ключ
Ном_будинку	Ціле число	
Кількість_квартир	Ціле число	Число > 0

Таблиця **"Оплата"**

Атрибути	*Тип*	*Опис*
Ном_квітанції	Ціле число	Первинний ключ
Особ_рахунок	Ціле число	Зовнішній ключ
Сума	Дійсне число	Число >= 0

Таблиця **"Квартира"**

Атрибути	*Тип*	*Опис*
Особ_рахунок	Ціле число	Первинний ключ
Ід_вулиці	Ціле число	Зовнішній ключ
Ном_будинку	Ціле число	
Прізвище_власника	Рядок	Максимальна довжина – 40 символів
Ім'я_власника	Рядок	Максимальна довжина – 15 символів
По_батькові_власника	Рядок	Максимальна довжина – 20 символів
Кількість_мешканців	Ціле число	Число > 0

Таблиця **"Спільн_лічильники"**

Атрибути	*Тип*	*Опис*
Ном_спільн_ліч	Рядок	Первинний ключ
Ід_вулиці	Ціле число	Зовнішній ключ
Ном_будинку	Ціле число	
Дата_остан_перевірки	Дата	
Дата_встанов	Дата	
Показ_при_встанов	Дійсне число	Число >= 0

Таблиця **"Індивід_лічильники"**

Атрибути	*Тип*	*Опис*
Ном_індивід_ліч	Ціле число	Первинний ключ
Особ_рахунок	Ціле число	
Дата_остан_перевірки	Дата	
Дата_встанов	Дата	
Показ_при_встанов	Дійсне число	Число >= 0

Таблиця **"Показання_спільн_ліч"**

Атрибути	*Тип*	*Опис*
Дата_зняття_показ	Ціле число	Первинний ключ
Ном_спільн_ліч	Ціле число	
Поточні_показ	Дійсне число	Число >= 0

Таблиця **"Показання_індив_ліч"**

Атрибути	*Тип*	*Опис*
Дата_зняття_показ	Ціле число	Первинний ключ
Ном_індивід_ліч	Ціле число	
Поточні_показ	Дійсне число	Число >= 0

Опишемо дії з таблицями БД (у термінах ДОДАТИ, ВИДАЛИТИ, ЗМІНИТИ), які мають бути реалізовані відповідно до функцій розроблюваного ПЗ:

- додати/видалити/змінити назву вулиці;
- додати/видалити/змінити дані про будинок;
- додати/видалити/змінити дані про квартиру;
- додати/видалити/змінити дані про спільний лічильник;
- додати/видалити/змінити дані про індивідуальний лічильник;
- додати показання спільного лічильника;
- додати показання індивідуального лічильника;
- додати дані про сплату за послугу.

Наведемо **запити** до БД (у термінах ВИВЕСТИ ...), які необхідно буде реалізувати відповідно до функцій розроблюваного ПЗ.

Запит 1. Вивести список адрес з прізвищами власників і сумами оплат, зроблених у зазначений період.

Запит 2. Вивести загальну кількість спожитих за указаний період кубічних метрів холодної води для квартири, яка знаходиться по вулиці з назвою N і в будинку з номером K.

Опишемо **дії**, які необхідно реалізувати на основі БД відповідно до функцій розроблюваного ПЗ.

Дія 1. Виконати нарахування за поточний місяць для кожної квартири (за переданими поточними показаннями індивідуальних лічильників, а в разі відсутності таких – за поточними показниками загальних лічильників).

Дія 2. Отримати відсортований за убуванням список боржників для будинку з номером K, який знаходиться по вулиці N, із зазначенням поточної суми боргу.

Дія 3. Обчислити нарахування за спожиту холодну воду за вказаний період для будинку з номером K, який знаходиться по вулиці N.

Дія 4. Обчислити загальну суму боргу за вказаний період для будинку з номером K, який знаходиться по вулиці N.

Наведемо **тригери**, які необхідно буде реалізувати.

Тригер. Якщо оплату за послугу "Холодне водопостачання" не внесено протягом 20 перших днів наступного місяця, то нараховується пеня в розмірі 0,05 % від спожитої послуги за кожен прострочений день.

6.4. Фізичне проектування бази даних

На етапі фізичного проектування БД необхідно на основі побудованої логічної моделі БД розробити її фізичну модель.

Для побудови фізичної моделі перш за все слід вибрати СУБД. Для розроблення системи автоматизації розрахунків за послугу "Холодне водопостачання" замовники вибрали СУБД MySQL.

CASE-засіб ERWin Data Modeler дає змогу задати для кожного атрибута таблиці відповідного домену конкретний тип, який підтримується вибраною СУБД, а саме СУБД MySQL. Унаслідок задання конкретних типів атрибутів таблиць було отримано схему БД, наведену на рис. 6.10.

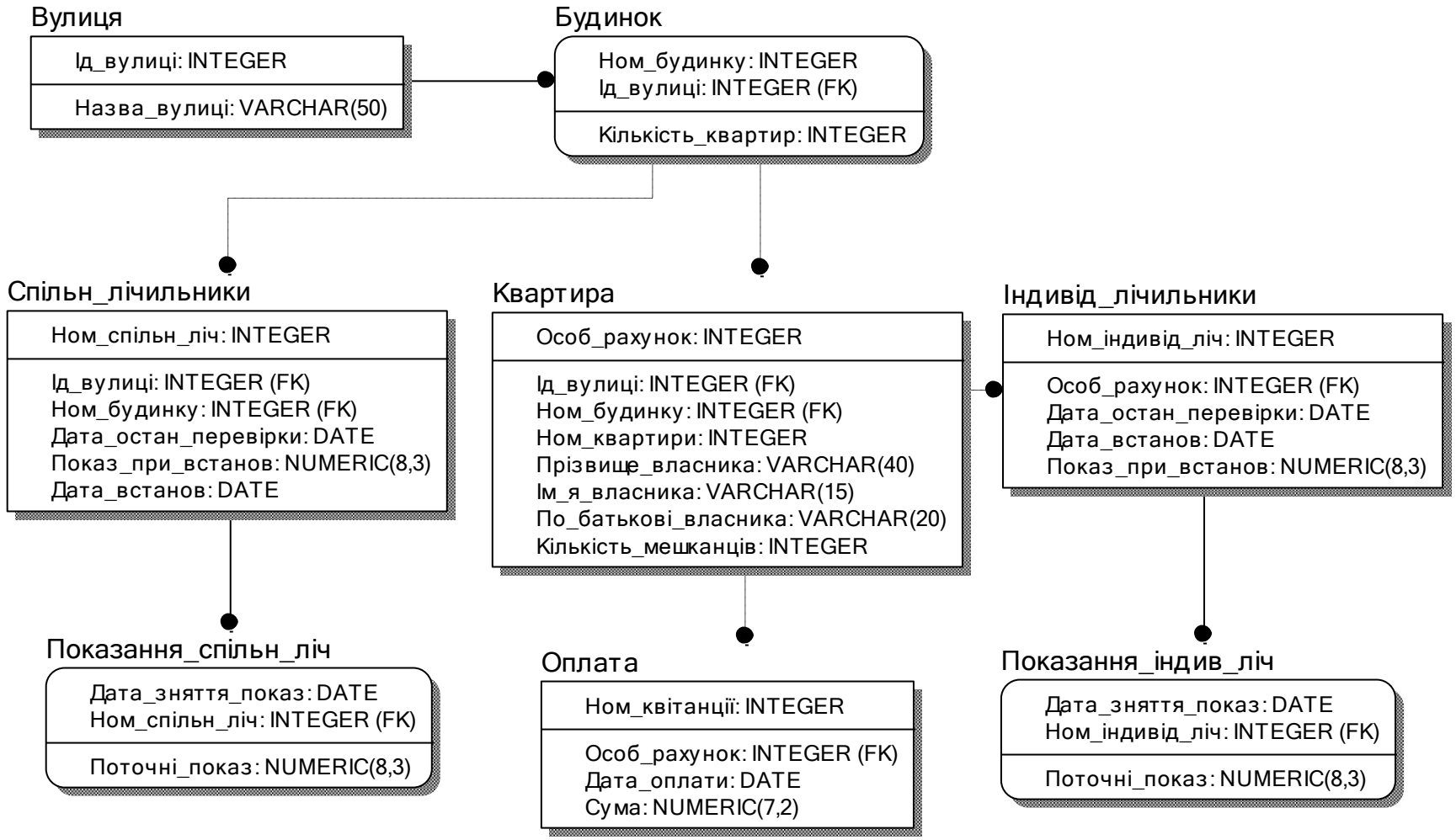


Рис. 6.10. Схема БД з конкретними типами атрибутів

CASE-засіб ERWin Data Modeler також дає змогу задати правила-обмеження на значення стовпців таблиць. Наприклад, можна задати правило, яке буде вказувати, що значення стовпця Показ_при_встанов таблиці "Спільн_лічильники" має бути додатним (рис. 6.11). Такі правила-обмеження на значення стовпців таблиць було створено відповідно до опису таблиць, наведених вище.

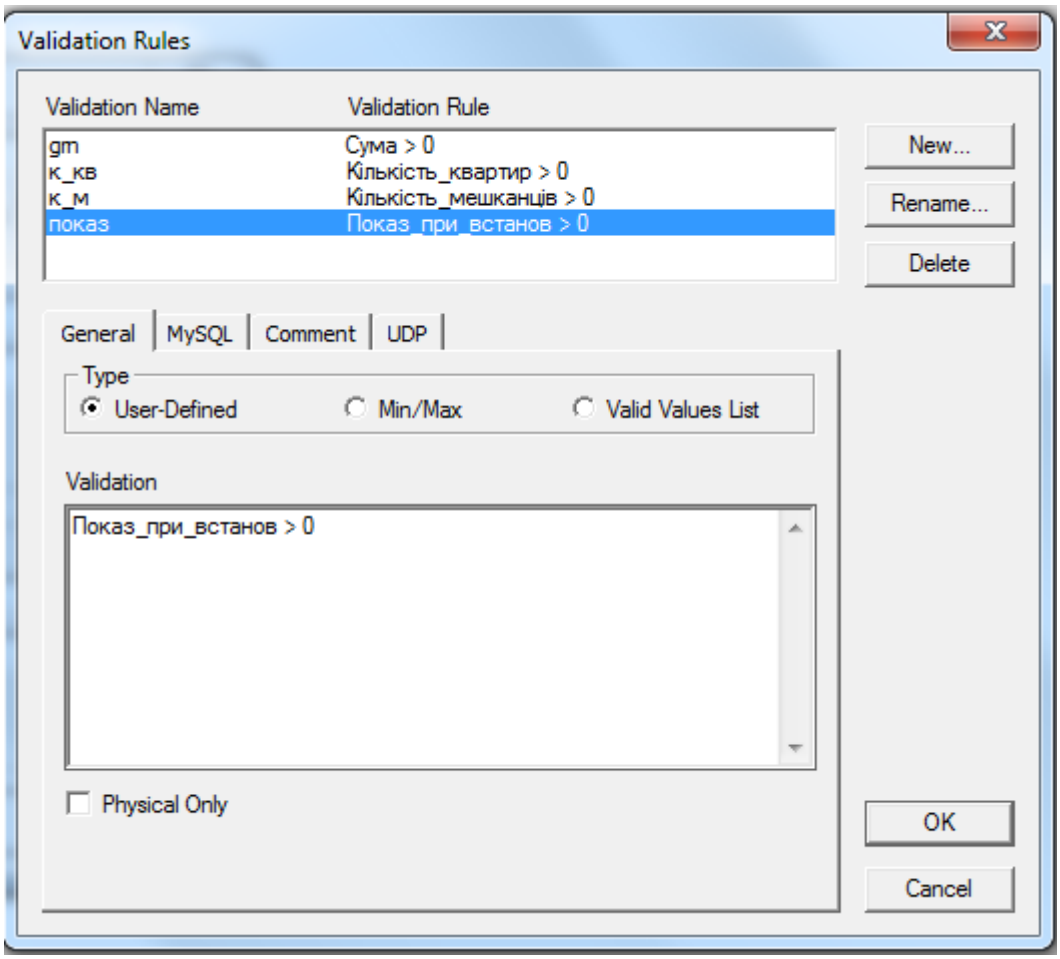


Рис. 6.11. Задання правил-обмежень на значення стовпців в ERWin

Крім того, CASE-засіб ERWin Data Modeler дає змогу в автоматичному режимі згенерувати SQL-скрипт. Зазначимо, що в отриманих таблицях всі стовпці є обов'язковими, тобто не допускається зберігання в них NULL-значень.

```
CREATE TABLE Вулиця
(       Ід_вулиці       INTEGER NOT NULL,
        Назва_вулиці    VARCHAR(50) NOT NULL,
        PRIMARY KEY (Ід_вулиці) );

CREATE TABLE Будинок
(       Ном_будинку     INTEGER NOT NULL,
        Ід_вулиці       INTEGER NOT NULL,
```

```
        Кількість_квартир    INTEGER NOT NULL CONSTRAINT к_кв
        CHECK (Кількість_квартир > 0),
        PRIMARY KEY (Ном_будинку, Ід_вулиці),
        FOREIGN KEY (Ід_вулиці) REFERENCES Вулиця (Ід_вулиці) );

CREATE TABLE Квартира
(       Особ_рахунок        INTEGER NOT NULL,
        Ід_вулиці           INTEGER NOT NULL,
        Ном_будинку         INTEGER NOT NULL,
        Ном_квартири        INTEGER NOT NULL,
        Прізвище_власника   VARCHAR(40) NOT NULL,
        Ім_я_власника       VARCHAR(15) NOT NULL,
        По_батькові_власника VARCHAR(20) NOT NULL,
        Кількість_мешканців  INTEGER NULL CONSTRAINT к_м
        CHECK (Кількість_мешканців > 0),
        PRIMARY KEY (Особ_рахунок),
        FOREIGN KEY (Ном_будинку, Ід_вулиці) REFERENCES Будинок
       (Ном_будинку, Ід_вулиці) );

CREATE TABLE Індивід_лічильники
(       Ном_індивід_ліч     INTEGER NOT NULL,
        Особ_рахунок        INTEGER NOT NULL,
        Дата_остан_перевірки   DATE NOT NULL,
        Дата_встанов        DATE NOT NULL,
        Показ_при_встанов   NUMERIC(8,3) NOT NULL CONSTRAINT показ
        CHECK (Показ_при_встанов > 0),
        PRIMARY KEY (Ном_індивід_ліч),
        FOREIGN KEY (Особ_рахунок) REFERENCES Квартира (Особ_рахунок) );

CREATE TABLE Показання_індив_ліч
(       Дата_зняття_показ   DATE NOT NULL,
        Ном_індивід_ліч     INTEGER NOT NULL,
        Поточні_показ       NUMERIC(8,3) NOT NULL CONSTRAINT показ
        CHECK (Показ_при_встанов > 0),
        PRIMARY KEY (Дата_зняття_показ, Ном_індивід_ліч),
        FOREIGN KEY (Ном_індивід_ліч) REFERENCES Індивід_лічильники
       (Ном_індивід_ліч) );

CREATE TABLE Спільн_лічильники
(       Ном_спільн_ліч      INTEGER NOT NULL,
        Ід_вулиці           INTEGER NOT NULL,
        Ном_будинку         INTEGER NOT NULL CONSTRAINT к_кв
        CHECK (Кількість_квартир > 0),
        Дата_остан_перевірки   DATE NOT NULL,
        Показ_при_встанов   NUMERIC(8,3) NOT NULL
        CONSTRAINT показ CHECK (Показ_при_встанов > 0),
        Дата_встанов        DATE NOT NULL,
        PRIMARY KEY (Ном_спільн_ліч),
        FOREIGN KEY (Ном_будинку, Ід_вулиці) REFERENCES
        Будинок (Ном_будинку, Ід_вулиці) );
```

```
CREATE TABLE Показання_спільн_ліч
(       Дата_зняття_показ    DATE NOT NULL,
        Ном_спільн_ліч       INTEGER NOT NULL,
        Поточні_показ      NUMERIC(8,3)  NOT NULL CONSTRAINT показ
        CHECK (Показ_при_встанов > 0),
        PRIMARY KEY (Дата_зняття_показ, Ном_спільн_ліч),
        FOREIGN KEY (Ном_спільн_ліч) REFERENCES Спільн_лічильники
        (Ном_спільн_ліч) );

CREATE TABLE Оплата
(       Ном_квітанції       INTEGER NOT NULL,
        Особ_рахунок        INTEGER NOT NULL,
        Дата_оплати          DATE NULL,
        Сума NUMERIC(7,2) NOT NULL CONSTRAINT grn CHECK (Сума > 0),
        PRIMARY KEY (Ном_квітанції),
        FOREIGN KEY (Особ_рахунок) REFERENCES Квартира (Особ_рахунок) );
```

Наступним етапом фізичного проектування БД є розроблення запитів, збережених процедур і тригерів БД, які покривають логіку функціонування ПЗ, що розробляється.

7. ПРАКТИКУМ З ПРОЕКТУВАННЯ БАЗ ДАНИХ

Лабораторна робота № 1
ПОБУДОВА ER-МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ

Мета роботи: навчитися будувати ER-модель (модель "сутність – зв'язок") для заданої предметної області.

Завдання:

1. Побудувати діаграму варіантів використання для програмного забезпечення (див. варіант нижче) і виконати її аналіз для виявлення тих функцій програмного забезпечення, які потребують збереження, вилучення й оброблення даних.

   Зауваження!!! Перелік основних функцій програмного забезпечення, зазначений у варіанті, не є вичерпним і має бути доповнений студентом (2–3 додаткові функції).

2. Виділити сутності предметної області та їх атрибути.
3. Установити зв'язки між виділеними сутностями.
4. Пойменувати виділені зв'язки і встановити їх кратність.
5. Показати виділені сутності й зв'язки за допомогою вибраної нотації (наприклад, нотація Чена, Мартіна і т. д.).
6. Підготувати звіт.

Лабораторна робота № 2
ПОБУДОВА ЛОГІЧНОЇ МОДЕЛІ БАЗИ ДАНИХ

Мета роботи: навчитися перетворювати ER-модель предметної області на логічну модель бази даних.

Завдання:

1. Визначити набір таблиць бази даних.
2. Реалізувати зв'язки між таблицями (крім зв'язків "багато до багатьох").
3. Перевірити отримані таблиці на відповідність першій, другій, третій нормальним формам і нормальній формі Бойса – Кодда.
4. У словесній формі сформулювати запити до таблиць для покриття функціональності програмного забезпечення.
5. У словесній формі сформулювати призначення процедур і функцій, а також тригерів для забезпечення функціональності розроблюваного програмного забезпечення.

Лабораторна робота № 3
ПОБУДОВА ФІЗИЧНОЇ МОДЕЛІ ДАНИХ

Мета роботи: навчитися перетворювати логічну модель бази даних на фізичну.

Завдання:

1. Перетворити зв'язки "багато до багатьох" між таблицями (створити асоціативні таблиці; за необхідності доповнити асоціативні таблиці додатковими атрибутами).
2. Обґрунтовано вибрати систему управління базою даних (СУБД).
3. Призначити стовпцям таблиць типи даних, що підтримуються вибраною СУБД, установити обмеження на значення стовпців таблиць бази даних.
4. Отримати SQL-скрипти для створення таблиць бази даних.
5. Створити таблиці бази даних (виконати SQL-скрипти, отримані в попередньому пункті).
6. Заповнити створені таблиці даними.

Лабораторна робота № 4
РОЗРОБЛЕННЯ SQL-СКРИПТІВ ЗАПИТІВ, ЗБЕРЕЖЕНИХ ПРОЦЕДУР І ФУНКЦІЙ, ТРИГЕРІВ

Мета роботи: розробити SQL-скрипти запитів, збережених процедур і функцій, а також тригерів для забезпечення функціональності програмного забезпечення.

Завдання:

1. Скласти, реалізувати й протестувати SQL-скрипти запитів для забезпечення функціональності розроблюваного програмного забезпечення.
2. Скласти, реалізувати й протестувати SQL-скрипти збережених процедур і функцій для забезпечення функціональності розроблюваного програмного забезпечення.
3. Скласти, реалізувати й протестувати SQL-скрипти тригерів для забезпечення функціональності розроблюваного програмного забезпечення.

Індивідуальне завдання
РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З ВИКОРИСТАННЯМ БАЗИ ДАНИХ

Мета роботи: спроектувати й реалізувати програмне забезпечення з використанням бази даних.

Порядок виконання роботи

1. Проектування бази даних.
2. Реалізація бази даних.
3. Проектування програмного забезпечення.
4. Реалізація програмного забезпечення.
5. Тестування програмного забезпечення.
6. Складання пояснювальної записки.

Перші два пункти індивідуального завдання повністю відповідають чотирьом лабораторним роботам, наведеним вище.

У пункті "Проектування програмного забезпечення" необхідно навести розроблену діаграму класів, описати методи класів.

У пункті "Реалізація програмного забезпечення" слід зробити обґрунтований вибір мови програмування й інструментального середовища програмування, навести екранні форми розробленого програмного забезпечення. Особливу увагу приділити опису механізму доступу до бази даних.

У пункті "Тестування програмного забезпечення" потрібно навести специфікації тестів і результати тестування.

Кожен етап роботи слід детально описати в пояснювальній записці. У кінці пояснювальної записки зробити висновки по роботі, навести список використаної літератури й лістинг коду розробленого програмного забезпечення. Обсяг пояснювальної записки має становити 35–40 сторінок.

Варіанти завдань

Варіант 1. Деканат факультету

Розробити програмне забезпечення для обліку сесійної успішності студентів факультету університету.

У програмі мають бути передбачені такі функції:

– облік спеціальностей факультету;

– облік навчальних планів для кожної спеціальності конкретного року набору студентів;

– облік сесійної успішності студентів (електронна залікова книжка студента: хто, що, як, коли, кому склав залік або іспит); оцінки проставляються за національною, 100-бальною і ECTS-шкалами;

– побудова рейтингового списку студентів за спеціальностями, курсом, факультетом за трьома шкалами (національною, 100-бальною і ECTS).

Варіант 2. Депозитні вклади

Розробити програмне забезпечення для обліку депозитних вкладів клієнтів банку, а також нарахування й виплати відсотків за ними. Депозитні програми пропонують різні умови:

– валюта вкладу (гривні, долари США, євро);

– термін вкладу;

– процентна ставка (залежно від валюти, терміну та суми вкладу);

– мінімальна сума вкладу (для кожної валюти);

– можливість поповнення депозиту (не для всіх вкладів);

– виплата відсотків (щомісячна або капіталізація відсотків, тобто зарахування їх на депозитний рахунок);

– пролонгація вкладу (виконується автоматично або не здійснюється).

У програмі мають бути передбачені такі функції:

– підтримка умов різних депозитних програм;

– облік депозитних вкладів клієнтів банку;

– нарахування відсотків за вкладами (капіталізація відсотків, якщо необхідно);

– пролонгація вкладу (якщо необхідно);

– створення власником депозитного вкладу особистого "кабінету" для перегляду нарахувань щомісячних відсотків і поточного стану депозитного вкладу.

Варіант 3. Готель

Розробити програмне забезпечення для обліку заселення номерного фонду готелю та його оплати.

У програмі мають бути передбачені такі функції:

– облік номерного фонду готелю;
– бронювання місць у готелі;
– заселення номерів у готелі;
– надання додаткових послуг (наприклад, харчування, прання й прасування одягу, Інтернет і т. ін.);
– формування квитанції за проживання в готелі (з урахуванням вартості бронювання й надання додаткових послуг).

Варіант 4. Приватна клініка

Розробити програмне забезпечення для обліку наданих платних послуг у приватній клініці.

У програмі мають бути передбачені такі функції:

– облік клієнтів клініки;
– облік прийомів лікарями різної спеціалізації (у кожного лікаря своя вартість прийому);
– облік платних лабораторних досліджень;
– формування квитанції для оплати наданих клієнту послуг;
– створення гнучкої системи знижок для клієнтів;
– облік доходів і витрат приватної клініки.

Варіант 5. Компанія експрес-доставки вантажів

Розробити програмне забезпечення для обліку замовлень на перевезення й доставку вантажів.

У програмі мають бути передбачені такі функції:

– облік клієнтів компанії;
– облік замовлень на перевезення вантажів;
– облік тарифів на доставку вантажів;
– облік тарифних зон;
– розрахунок вартості доставки (вартість доставки = ціна за 1 кг х маса вантажу + вартість оформлення замовлення + сума комісії, де сума комісії = = 0,5 % х заявлена вартість вантажу);
– відстеження перевезення вантажу.

Варіант 6. Відділ продажів фірми

Розробити програмне забезпечення для обліку замовлень на поставку виробів, що виготовляються фірмою.

У програмі мають бути передбачені такі функції:

– облік виробів, що виготовляються, і норм матеріалів для їх виготовлення;

– облік матеріалів;
– облік клієнтів фірми;
– облік замовлень на виготовлення виробів;
– формування квитанцій на оплату замовлення;
– аналіз можливості реалізації замовлення з урахуванням наявних ресурсів;
– закупівля необхідних матеріалів (формування закупівельної відомості);
– створення системи знижок для постійних клієнтів.

Варіант 7. Туристичне агентство

Розробити програмне забезпечення для обліку наявних і проданих турів клієнтам фірми.

У програмі мають бути передбачені такі функції:
– надання інформації про наявні тури по країнах і містах, а також про типи готелів;
– облік співробітників агентства;
– облік клієнтів туристичного агентства;
– оформлення турів;
– створення системи знижок;
– формування квитанцій на оплату;
– формування рейтингу готелів за кількістю оформлених турів;
– нарахування премії співробітникам за оформлення туру (премія = = вартість туру х преміальний відсоток);
– нарахування заробітної плати співробітникам (заробітна плата = = оклад + премія).

Варіант 8. Банківські операції

Розробити програмне забезпечення для відкриття, ведення й закриття банківських рахунків юридичних і фізичних осіб, а також оформлення операцій з купівлі-продажу іноземної валюти.

У програмі мають бути передбачені такі функції:
– відкриття рахунків юридичних і фізичних осіб із зазначенням банківського відсотка за зарахування на цей рахунок грошових коштів;
– ведення рахунків юридичних і фізичних осіб (зарахування й зняття коштів);
– закриття рахунків юридичних і фізичних осіб;
– облік операцій з купівлі-продажу іноземної валюти (з урахуванням поточного курсу валюти);
– формування квитанцій про зарахування або зняття грошових коштів, купівлю-продаж іноземної валюти;

– формування звіту про надходження або знаття коштів за вказаний період;

– формування звіту про обсяги купленої та проданої іноземної валюти за вказаний період.

Варіант 9. Мережа аптек

Розробити програмне забезпечення для обліку руху медичних препаратів і засобів.

У програмі мають бути передбачені такі функції:

– облік медичних товарів за категоріями й фірмами-виробниками;

– пошук медичних товарів;

– надходження товарів до аптеки із зазначенням їхньої вартості при поставці;

– формування націнки на медичні товари (за категоріями й фірмам-виробникам);

– продаж товарів (формування чека);

– облік залишків товару в аптеці;

– створення дисконтної програми (відсоток знижки змінюється залежно від загальної суми всіх куплених медичних товарів);

– переміщення товару між аптеками;

– резервування товару;

– формування прайс-листа;

– формування повного акта інвентаризації (список медичних товарів, які на цей момент мають бути в наявності в аптеці).

Варіант 10. Оператор мобільного зв'язку

Розробити програмне забезпечення для обліку телефонних розмов абонентів оператора мобільного зв'язку.

У програмі мають бути передбачені такі функції:

– зберігання інформації про тарифні плани;

– облік абонентів та інформації про їх тарифні плани;

– облік розмов абонента та їх тарифікація;

– облік sms-повідомлень та їх тарифікація;

– реєстрації для створення особистого "кабінету" абонента;

– формування звіту про дзвінки й надіслані sms-повідомлення абонента за вказаний період;

– формування звіту про поповнення рахунка абонента за вказаний період.

Варіант 11. Піцерія

Розробити програмне забезпечення для обліку замовлень і доставки піци клієнтам.

У програмі мають бути передбачені такі функції:

– зберігання інформації про види піци, її склад і вагу, а також про її вартість;
– облік клієнтів піцерії;
– облік замовлень клієнтів;
– зберігання інформації про акційні пропозиції та їх облік при формуванні замовлення;
– формування звітів щодо зроблених замовлень за видами піци за певний період з метою виявлення найбільш популярних пропозицій;
– створення накопичувальної системи знижок (відсоток знижки змінюється залежно від суми всіх зроблених клієнтом замовлень).

Варіант 12. Дитячий центр розвитку

Розробити програмне забезпечення для обліку відвідувань дітьми розвивальних гуртків.

У програмі мають бути передбачені такі функції:

– зберігання інформації щодо занять із зазначенням їхньої вартості;
– розподіл занять з урахуванням вікових категорій: від нуля до двох років, від двох до чотирьох років, від чотирьох до шести років;
– зберігання інформації щодо викладачів, які ведуть заняття;
– зберігання інформації щодо розкладу занять;
– зберігання інформації щодо дітей, які відвідують центр розвитку;
– зберігання інформації щодо того, які заняття відвідують діти;
– зберігання новин центру розвитку;
– зберігання відгуків батьків;
– формування звітів про надходження коштів за певний період часу;
– реалізація системи знижок для дітей, які відвідують кілька гуртків.

Варіант 13. Оптовий магазин

Розробити програмне забезпечення для обліку поставок і купівель в оптовому магазині.

У програмі мають бути передбачені такі функції:

– зберігання інформації про товари та їх залишки;
– зберігання інформації про постачання товарів в оптовий магазин (облік прибуткових накладних);
– зберігання інформації про продажі товарів (облік видаткових накладних) і їх доставку клієнтам;

– зберігання інформації про клієнтів;
– створення накопичувальної системи знижок (відсоток знижки змінюється залежно від суми всіх зроблених купівель);
– зберігання інформації про акції і їх облік при продажах товарів;
– оплата товару в розстрочку;
– облік платежів клієнта при купівлі товару в розстрочку;
– формування фінансових звітів (витрати / надходження) за вказаний період.

Варіант 14. Бібліотека

Розробити програмне забезпечення для обліку руху книжкового фонду бібліотеки.

У програмі мають бути передбачені такі функції:
– зберігання інформації про книжковий фонд бібліотеки;
– зберігання інформації про новинки книжкового фонду;
– зберігання інформації про читачів бібліотеки;
– зберігання електронних видань і забезпечення можливості доступу до них читачам бібліотеки;
– зберігання інформації щодо книг, які перебувають на руках у читача, із зазначенням терміну повернення книг;
– формування списку читачів, які не повернули книги в зазначений термін.

Варіант 15. Домашня бухгалтерія

Розробити програмне забезпечення для обліку руху фінансів усіх членів сім'ї.

У програмі мають бути передбачені такі можливості:
– облік витрат кожного члена сім'ї;
– облік доходів кожного члена сім'ї;
– облік кредитів та їх погашення;
– планування витрат (загальне і кожним членом сім'ї);
– планування доходів (загальне і кожним членом сім'ї);
– формування звітів і побудова діаграм щодо доходів і витрат сім'ї за місяць, квартал, рік;
– забезпечення конфіденційності витрат і доходів кожного члена сім'ї (якщо необхідно).

Варіант 16. Супермаркет

Розробити програмне забезпечення для обліку товарів, що надійшли й були продані.

У програмі мають бути передбачені такі функції:

– облік товарів за категоріями й підкатегоріями;
– продаж товарів (формування чека);
– облік дисконтних карт клієнтів;
– створення накопичувальної системи знижок;
– організація різних акцій на окремі товари за вказаний період часу (наприклад, зниження вартості, "два за ціною одного", "три за ціною двох" і т. ін.).

Варіант 17. Ріелторська фірма (агентство нерухомості)

Розробити програмне забезпечення для обліку укладених договорів купівлі-продажу й оренди комерційної та житлової нерухомості.

У програмі мають бути передбачені такі функції:

– облік ріелторів фірми;
– облік комерційної та житлової нерухомості, виставленої на продаж;
– облік комерційної та житлової нерухомості, запропонованої для оренди;
– облік укладених договорів (угод) купівлі-продажу й оренди;
– розрахунок комісії ріелторської фірми за здійснення угоди (комісія фірми = сума угоди х відсоток комісії);
– розрахунок премії ріелтора за угоду (премія = комісія фірми х х преміальний відсоток);
– нарахування заробітної плати ріелторам (заробітна плата = оклад + + преміальний відсоток від суми всіх укладених угод за місяць).

Варіант 18. Мережа магазинів побутової техніки

Розробити програмне забезпечення для обліку співробітників мережі магазинів побутової техніки й проданих ними товарів.

У програмі мають бути передбачені такі функції:

– облік співробітників за відділами магазину й посадовими категоріями з окладами, які відповідають категорії співробітника;
– облік товарів за категоріями й фірмами-виробниками;
– продаж товарів (формування видаткової накладної на конкретного співробітника);
– нарахування премії співробітнику за місяць (премія = продаж за місяць х відсоток премії, який залежить від категорії товарів);
– нарахування заробітної плати співробітникам за місяць (заробітна плата = оклад + премія);
– формування рейтингу співробітників за їх обсягом продажів за певний період;
– формування рейтингу співробітників за їх обсягом продажів по всій

мережі магазинів;

– формування рейтингу магазинів за їх обсягом продажів.

Варіант 19. Страхова компанія

Розробити програмне забезпечення для обліку страхових договорів і страхових випадків, що виникають.

У програмі мають бути передбачені такі функції:

– облік страхових послуг (назва та їх умови);
– облік укладених страхових договорів та їх оплата;
– облік страхових випадків, що виникли, та виплати по них;
– пролонгація страхового договору з встановленням знижки на страхові послуги;
– формування звітів (щомісячних, щоквартальних, щорічних).

Варіант 20. Бухгалтерія фірми

Розробити програмне забезпечення для обліку співробітників фірми, нарахування й виплати їм заробітної плати.

У програмі мають бути передбачені такі функції:

– облік співробітників по займаних посадах і відділах, за якими їх закріплено;
– переведення співробітників на іншу посаду або інше місце роботи;
– нарахування щомісячних преміальних виплат кожному співробітнику залежно від виконаних робіт (відсоток від суми окладу);
– нарахування заробітної плати (заробітна плата = оклад за посадою + + премія за поточний місяць – відрахування);
– здійснення виплати співробітникам (авансу і заробітної плати);
– формування щомісячних розрахункових листів для кожного співробітника із зазначенням нарахувань, утримань та виплат у поточному місяці.

Варіант 21. Автозаправна станція

Розробити програмне забезпечення для обліку поставленого й проданого палива.

У програмі мають бути передбачені такі функції:

– облік поставок палива різних видів;
– облік продажів палива різних видів;
– облік дисконтних карт клієнтів;
– створення накопичувальної системи бонусів (1 літр = 1 бонус);
– купівля супутніх товарів за бонуси;
– проведення різних акцій із зазначенням їх періоду (наприклад,

подвійне нарахування бонусів);

– списання бонусів (1 бонус = 0,1 грн), але списати можна не більше 30 % від вартості палива.

Варіант 22. Об'єднання співвласників багатоквартирного будинку

Розробити програмне забезпечення для обліку нарахувань та оплат за надані послуги холодного й гарячого водопостачання, теплопостачання, електропостачання, газопостачання й експлуатаційних витрат для мешканців об'єднання співвласників багатоквартирного будинку (ОСББ). Передбачається, що всі квартири ОСББ забезпечено лічильниками для обліку витрат холодної та гарячої води, електрики, газу й теплопостачання.

У програмі мають бути передбачені такі функції:

– облік квартирного фонду ОСББ із зазначенням загальної та житлової площ кожної квартири, прізвища, імені й по батькові власника квартири, кількості прописаних осіб і т. ін.;

– облік спожитих за місяць (згідно з лічильником) обсягів холодної та гарячої води (у кубічних метрах), електрики (у кіловатах), газу (у кубічних метрах), теплової енергії (у гігакалоріях) для кожної квартири;

– розрахунок експлуатаційних витрат на утримання будинку від загальної площі квартири з урахуванням установленого тарифу за 1 кв.м;

– нарахування за місяць для кожної квартири за холодне і гаряче водопостачання, електрику, газ, теплопостачання, експлуатацію будинку (для кожного виду послуг існують свої тарифи);

– формування щомісячної квитанції з усіма видами послуг для кожної квартири (нараховано до сплати = борг на початок поточного місяця + + нарахування за місяць);

– облік оплат мешканців на єдиний розрахунковий рахунок ОСББ;

– формування списку боржників на зазначену дату.

Варіант 23. Інтернет-магазин

Розробити програмне забезпечення для обліку товарів, що надійшли й були продані в Інтернет-магазині.

У програмі мають бути передбачені такі функції:

– реєстрація користувачів;

– облік товарів за категоріями й підкатегоріями;

– фільтрація товарів за вказаними параметрами;

– продаж товарів (наповнення кошика, оброблення й виконання замовлення, доставка товарів);

– створення накопичувальної системи знижок (відсоток знижки

змінюється залежно від суми всіх зроблених замовлень);

– організація акцій на окремі товари (зниження вартості товарів із зазначенням акційного періоду й кількості акційних товарів).

Варіант 24. WEB-форум

Розробити програмне забезпечення для створення користувачами (відвідувачами форуму) своїх тем з їх подальшим обговоренням, тобто розміщенням повідомлень всередині цих тем.

У програмі мають бути передбачені такі функції:

– робота з різними типами користувачів (модератор, адміністратор, зареєстрований і незареєстрований відвідувачі форуму);

– можливість створення модераторами й адміністраторами розділів для об'єднання тем однієї тематики;

– можливість створення зареєстрованим користувачем форуму теми у вибраному розділі, залишення повідомлень;

– редагування модераторами повідомлень відвідувачів або видалення їх, а також переміщення повідомлень в інші теми;

– обмін зареєстрованими користувачами особистими повідомленнями;

– наявність у зареєстрованого користувача особистого "кабінету", де він може побачити всі свої теми, залишені повідомлення-коментарі, вхідні й вихідні приватні повідомлення.

Варіант 25. Електронні громадські петиції

Розробити програмне забезпечення для обліку електронних громадських петицій до органів міського управління.

У програмі мають бути передбачені такі функції:

– реєстрація користувачів, які бажають подати електронну петицію;

– подання зареєстрованим користувачем петиції;

– можливість зареєстрованим користувачам проголосувати за вибрану їм петицію;

– можливість адміністратору змінити єдиний термін збору голосів;

– можливість адміністратору змінити єдину кількість голосів, необхідну для відправлення петиції на розгляд до органів міського управління;

– у разі, якщо петиція набрала необхідну кількість голосів, її відправляють на розгляд до органів міського управління, після чого публікується офіційна відповідь на петицію.

Варіант 26. Оренда автомобілів

Розробити програмне забезпечення для обліку наявного парку автомобілів, а також оренди автомобілів.

У програмі мають бути передбачені такі функції:

– реєстрація нового автомобіля;
– реєстрація клієнтів;
– бронювання автомобіля на певний період;
– облік оренди автомобілів;
– облік пошкоджень автомобілів під час їх оренди;
– генерація рахунків за оренду автомобіля з урахуванням майбутнього ремонту автомобіля;
– контроль оплати рахунків клієнта.

Варіант 27. Акційні пропозиції

Розробити програмне забезпечення для обліку купівель акційних пропозицій (купонів) на різні послуги.

У програмі мають бути передбачені такі функції:

– реєстрація клієнтів, які надають послуги;
– подання заявки на реєстрацію нової акційної пропозиції у вибраній категорії з установленням терміну дії акції та кількості пропонованих купонів і вартості купона;
– активація адміністратором нової акційної пропозиції;
– реєстрація користувача, який хоче скористатися акційними пропозиціями;
– можливість купівлі купонів, у разі якщо купони є в наявності й термін акції є дійсним;
– можливість повернення купленого купона;
– наявність особистого "кабінету" користувача з можливістю перегляду всіх куплених купонів.

Варіант 28. Анкетування

Розробити програмне забезпечення для створення анкети й обліку результатів анкетування.

У програмі мають бути передбачені такі функції:

– можливість складання анкети з запитань різних типів, а також різного виду заповнень (персоналізоване/анонімне);
– можливість заповнення анкети зареєстрованим або анонімним користувачем (залежно від її типу), у разі якщо анкетування є відкритим;
– збір статистичних даних на основі відповідей на різні запитання конкретної анкети.

Варіант 29. Тестування знань

Розробити програмне забезпечення для створення тестів та обліку результатів тестування. У програмі мають бути передбачені такі функції:

– можливість складання тесту з різних видів завдань закритого типу (з альтернативним і множинним вибором);

– можливість персоналізованого й анонімного проходження тесту;

– наявність особистого "кабінету" особи, яка тестується, з можливістю перегляду історії проходження вибраного тесту;

– генерація звітів про результати проходження вибраного тесту.

Варіант 30. Автомобільна інспекція

Розробити програмне забезпечення для обліку зареєстрованих транспортних засобів і обліку правопорушень правил дорожнього руху.

У програмі мають бути передбачені такі функції:

– реєстрація транспортних засобів за їх типам;

– фіксація правопорушень правил дорожнього руху конкретним транспортним засобом;

– облік оплат виписаних штрафів;

– збір статистичних даних про кількість зареєстрованих злочинів за певний період.

Варіант 31. Дистанційне навчання

Розробити програмне забезпечення для обліку учнів, зареєстрованих на дистанційне навчання.

У програмі мають бути передбачені такі функції:

– реєстрація викладачів дистанційних курсів;

– реєстрація дистанційних курсів: курс має різні теми, з кожної теми надаються завдання для самостійного виконання;

– формування розкладу дистанційних курсів;

– реєстрація учнів на дистанційні курси;

– облік виконаних самостійних завдань та оцінок, отриманих за ці завдання;

– облік платні за дистанційне навчання;

– видача сертифікатів після проходження дистанційного курсу.

Варіант 32. Дошка оголошень

Розробити програмне забезпечення для обліку оголошень купівлі/продажу товарів. У програмі мають бути передбачені такі функції:

– реєстрація користувача;

– подання користувачем оголошення за вибраною категорією;
– активація адміністратором поданого оголошення;
– зміна статусу оголошення: неактивоване – спочатку при поданні оголошення, активоване – після перевірки оголошення адміністратором, продане – при неактуальності оголошення, деактивоване – при закінченні строку подання оголошення;
– можливість відправити запитання власнику оголошення;
– можливість відповісти за запитання щодо оголошення;
– можливість залишити коментар про власника оголошення;
– наявність особистого "кабінету" з переглядом власних оголошень та надісланих запитань.

Варіант 33. Онлайн-бронювання оренди житла

Розробити програмне забезпечення для онлайн-бронювання оренди житла. У програмі мають бути передбачені такі функції:
– реєстрація орендодавця та його житла;
– наявність особистого "кабінету" користувача з історією оренди житла;
– бронювання житла на вільний період часу;
– можливість користувачем залишити коментар про певне житло й поставити оцінку за п'ятибальною шкалою.

Варіант 34. Податкова інспекція

Розробити програмне забезпечення для обліку платників податків. У програмі мають бути передбачені такі функції:
– реєстрація платника податків різної категорії;
– реєстрація податкових декларацій;
– облік сплачених податків;
– формування списків боржників, що не подали декларацій;
– формування списків боржників, що не сплатили податки.

Варіант 35. Онлайн-купівля авіаквитків

Розробити програмне забезпечення для обліку онлайн-купівель квитків на внутрішні та міжнародні авіарейси.

У програмі мають бути передбачені такі функції:
– реєстрація адміністратором авіарейсу і квитків;
– наявність особистого "кабінету" користувача з історією всіх купівель;
– бронювання користувачем квитків на авіарейс;
– купівля користувачем квитків на авіарейс;
– відмова користувачем від куплених квитків на авіарейс.

БІБЛІОГРАФІЧНИЙ СПИСОК

1. Дейт, К. Дж. Введение в системы баз данных [Текст] / К. Дж. Дейт. − М.: Вильямс, 2006. − 1328 с.
2. Гарсиа-Молина, Г. Системы баз данных. Полный курс [Текст] / Г. Гарсиа-Молина, Дж. Ульман, Дж. Уидом. − М.: Вильямс, 2003. − 1088 с.
3. Коннолли, Т., Бегг, К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика [Текст] / Т. Коннолли, К. Бегг. − М.: Вильямс, 2017. − 1440 с.
4. Зеленков, Ю. А. Введение в базы данных [Электронный ресурс] / Ю. А. Зеленков. – Режим доступа: http://www.mstu.edu.ru/study/materials/zelenkov/toc.html
5. Марченко, А. В. Об'єктно-орієнтовані бази даних [Електронний ресурс] / А. В. Марченко. – Режим доступу: https://ocw.sumdu.edu.ua/content/811#node86166
6. Кузнецов, С. Д. Введение в реляционные базы данных [Текст] / С. Д. Кузнецов. – М.: НОУ "Интуит", 2016. – 248 с.
7. Туманов, В. Е. Основы проектирования реляционных баз данных [Текст] / В. Е. Туманов. – М.: НОУ "Интуит", 2016. – 504 с.
8. Курс лекций по проектированию баз и хранилищ данных [Электронный ресурс]. – Режим доступа: http://www.radioland.net.ua/contentid-122-page1.html
9. Карпова, Т. С. Базы данных: модели, разработка, реализация [Текст] / Т. С. Карпова. – М.: НОУ "Интуит", 2016. – 403 с.
10. Грошев, А. С. Основы работы с базами данных [Текст] / А. С. Грошев. – М.: НОУ "Интуит", 2016. – 256 с.
11. Швецов, В. И. Базы данных [Текст] / В. И. Швецов. – М.: НОУ "Интуит", 2016. – 218 с.
12. Илюшечкин, В. М. Основы использования и проектирования баз данных [Текст] / В. М. Илюшечкин. – М.: Юрайт, 2017. – 214 с.
13. Баженова, И. Ю. Основы проектирования приложений баз данных [Текст] / И. Ю. Баженова. – М.: НОУ "Интуит", 2017. – 328 с.
14. Стружкин, Н. П., Годин, В. В. Базы данных: проектирование [Текст] / Н. П. Стружкин, В. В. Годин. – М.: Юрайт, 2018. – 292 с.
15. Маклаков, С.В. BPwin и ERwin: CASE-средства для разработки информационных систем [Текст] / С.В. Маклаков. – М.: Диалог-МиФи, 2001. – 304 с.

ЗМІСТ

Навчальне видання

Шевченко Ілона Володимирівна
Манжос Юрій Семенович

ПРОЕКТУВАННЯ БАЗ ДАНИХ

Редактор А. М. Ємленінова

Зв. план, 2018
Підписано до друку 15.06.2018
Формат 60x84 1/16. Папір офс. № 2. Офс. друк
Ум. друк. арк. 5,1. Обл.-вид. арк. 5,75. Наклад 100 пр. Замовлення 201.
Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
"Харківський авіаційний інститут"
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр "ХАІ"
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб'єкта видавничої справи
до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої
продукції сер. ДК № 391 від 30.03.2001