УКРАЇНСЬКА ДЕРЖАВНА АКАДЕМІЯ ЗАЛІЗНИЧНОГО ТРАНСПОРТУ

ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра „Обчислювальна техніка та системи управління"

С.Є. Бантюков, В.М. Бутенко, В.Г. Пчолін

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

"ІНФОРМАЦІЙНІ СИСТЕМИ НА ЗАЛІЗНИЧНОМУ ТРАНСПОРТІ"

Частина 1

Харків 2008

Бантюков С.Є., Бутенко В.М., Пчолін В.Г. Конспект лекцій з дисципліни “Інформаційні системи на залізничному транспорті”. – Харків: УкрДАЗТ, 2008. – Ч. 1. – 64 с.

Конспект розроблено у відповідності з програмою курсу "Інформаційні системи на залізничному транспорті". До нього включено розділи, що необхідні студентам для опанування теоретичних та практичних основ застосування інформаційних систем для рішення завдань управління на залізничному транспорті та виробництві, зокрема. проектування спеціалізованих реляційних баз даних.

Рекомендується для студентів факультету УПП денної форми навчання, що вивчають дисципліну "Інформаційні системи на залізничному транспорті".

Конспект лекцій розглянуто та рекомендовано до друку на засіданні кафедри “Обчислювальна техніка та системи управління” 28 березня 2007 р., протокол № 8.

Рецензент

проф. Г.І.Загарій

ЗМІСТ

ВСТУП

Головна мета викладання дисципліни “Інформаційні системи” в академії – це підготувати студентів до свідомого та ефективного застосування інформаційних систем у своїй майбутній фаховій діяльності. При цьому особлива увага приділяється всебічному розкриттю поняття автоматизованої інформаційної системи, класифікаціям інформаційних систем, їх характеристикам та напрямку їх розвитку, складу та властивостям їх основних компонентів, питанням практичного використання інформаційних систем у господарській сфері, зокрема у залізничних перевезеннях.

Іншою важливою частиною курсу слід вважати розгляд питань проектування баз даних, оскільки реляційна база даних є головною складовою кожної сучасної автоматизованої інформаційної системи. Після вивчення курсу студенти будуть здатні самостійно розробляти реляційні бази даних відповідно до потреб своєї професійної діяльності.

1 ІНФОРМАЦІЙНІ СИСТЕМИ (ІС) ТА НАУКОВО-ТЕХНІЧНИЙ ПРОГРЕС. РОЛЬ ІС В УПРАВЛІННІ ПІДПРИЄМСТВОМ

Програмне забезпечення (ПЗ) за півсторіччя свого існування зазнало величезних змін, пройшовши шлях від програм, здатних виконувати тільки найпростіші логічні й арифметичні операції, до складних систем управління підприємством.

У ПЗ завжди можна було виділити два основних напрямки розвитку:

1) виконання обчислень;

2) накопичування й обробка інформації.

Хоча спочатку комп’ютери призначалися, головним чином, для виконання складних математичних розрахунків, зараз домінуючим є другий напрямок.

Це пояснюється насамперед тим, що цивільні галузі застосування комп’ютерів набагато більше поширені, ніж наукові або військові, а зниження вартості комп'ютерів зробило їх доступними для невеликих підприємств і навіть приватних осіб.

Сьогодні управління підприємством неможливе без засобів обчислювальної техніки. Комп'ютери давно й міцно ввійшли в такі галузі управління, як бухгалтерський облік, управління складом, асортиментами, закупівлями. Однак на благополуччі роботи сучасного підприємства стали негативно відображатися помилки в його управлінні. Кваліфікованість персоналу, інтуїція та особистий досвід керівника вже недостатні для успішної роботи підприємства, потрібне вже більш широке застосування інформаційних технологій в його управлінні. Для прийняття грамотного управлінського рішення в умовах невизначеності й ризику необхідно постійно тримати під контролем різні аспекти фінансово-господарської діяльності. Тому сучасний підхід до керування припускає вкладення коштів в інформаційні технології. І чим більше і значніше підприємство, тим серйозніші мають бути такі вкладення.

Виділяють дві основні причини, які сприяли створенню ІС:

1 Незаперечний факт, що сучасне суспільство досягло такого стану, коли всі люди, що населяють землю, самі вже не можуть переробляти постійно зростаючі за кількістю потоки інформації.

2 Посилення протиріччя між своєчасністю й вірогідністю інформації, що використовується в управлінні.

Перша причина не має потреби в поясненнях, другу – пояснимо.

У процесі управління завжди задіяні об'єкт управління та орган управління. Орган управління керує об'єктом. Процес управління відбувається протягом часу, при цьому змінюють свої стани і об'єкт управління, і орган управління. Зобразимо зміну цих станів за допомогою часових діаграм (див. рисунок 1).

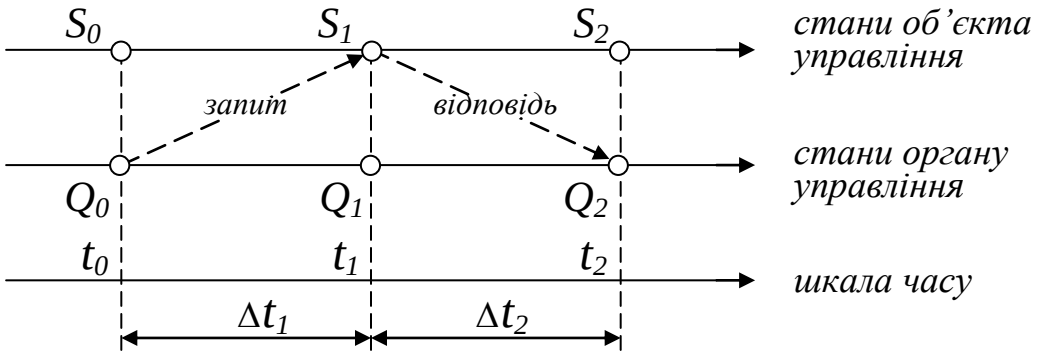


Рисунок 1

Щоб успішно керувати об’єктом управління, в органі управління повинна бути інформація про поточний стан об’єкта управління. Стани самого органу управління теж важливі для процесу управління, але ми не будемо на них загострювати свою увагу. У момент часу t_0 орган управління, перебуваючи в стані Q_0, робить запит про поточний стан об’єкта управління (тобто про стан S_0). На підготовку й відсилання запиту потрібен час: зобразимо його на діаграмах у вигляді відрізка Δt_1. Збирання необхідної інформації про об’єкт і зворотне відсилання відповіді на запит органу управління теж триває певний час: на діаграмах воно відображається відрізком Δt_2. Таким чином, орган управління в момент t_0 ніяк не буде мати інформацію про поточний стан об’єкта управління S_0, не буде він її мати і в момент одержання відповіді (тобто в момент t_2). Більше того, у момент t_2 органом управління буде отримана інформація не про поточний стан об’єкта управління (не про стан S_2), а про зовсім інший і уже застарілий стан S_1. Інакше кажучи, орган управління завжди має несвоєчасну й недостовірну інформацію про стан об’єкта управління. Звісно, щоб підвищити вірогідність своєчасності отримання інформації та її достовірність, треба якось зменшувати Δt_1 і Δt_2. Це можна зробити, тільки використовуючи інформаційні системи на базі комп’ютерів.

Однак існує зовнішня перешкода, яка обмежує розвиток ІС зверху. Вона виражається в так званому “законі (принципі) необхідної розмаїтості”, який було сформульовано В.Россом Ешбі[1)]. Відповідно до цього закону, система, що управляє, повинна мати не меншу розмаїтість станів, ніж та система, якою вона управляє, щоб мати можливість активного впливу на останню. Інакше кажучи, щоб уміти керувати об’єктом управління й реагувати на всі його зміни, орган управління повинен мати таку ж розмаїтість властивостей і можливостей, як і об’єкт управління, тобто мати детальну відповідність своїх властивостей і станів властивостям і станам об’єкта управління. Орган управління повинен уміщати в себе інформаційну модель об’єкта управління.

[1)]Вільям Росс Ешбі (William Ross Ashby), народився 6 вересня 1903 р. у Лондоні, Англія. Помер 15 листопада 1972 р. Англійський психіатр, видатний фахівець у галузі кібернетики, основоположник дослідження складних систем.

2 ТЛУМАЧЕННЯ ТЕРМІНА ІС

Основою більшості фінансово-господарських завдань є обробка інформації. Для полегшення й прискорення обробки інформації створюються інформаційні системи, тобто в широкому розумінні ІС – це будь-яка система обробки інформації (наприклад, система використання карток і каталогів у бібліотеках).

Автоматизованими називаються ІС, у яких застосовують технічні пристрої, зокрема ЕОМ. Більшість існуючих ІС є автоматизованими, тому для стислості будемо називати їх просто – ІС.

Однак існує й всюди використовується більш вузьке трактування терміна ІС як сукупності апаратно-програмних засобів, задіяних для вирішення деякого прикладного завдання, пов'язаного із збиранням, зберіганням і обробкою певних даних (наприклад, бухгалтерського обліку, обліку кадрів, матеріально-технічних засобів тощо). Більше того, оскільки зараз апаратною основою майже кожної ІС є обчислювальна система, що зібрана на базі стандартних універсальних цифрових ЕОМ, зв'язаних у мережі за допомогою стандартних допоміжних пристроїв, то можна розглядати ІС тільки як програмний продукт, тобто як спеціалізоване програмне забезпечення.

На ці три останні тлумачення терміна ІС ми будемо спиратися під час вивчення курсу.

3 РІВНІ ОБРОБКИ ІНФОРМАЦІЇ В ІС

Традиційно при функціонуванні ІС виділяють три рівні оброблюваної інформації:

1) оперативний;
2) тактичний;
3) стратегічний.

Інформація оперативного рівня потрібна в щоденній роботі. Вона первинна, часто оновлюється і є основою ієрархії ІС. Її збереження й обробку автоматизують у першу чергу.

Тактична інформація виходить шляхом узагальнення інформації оперативного типу. Вона призначається для

керівників середньої ланки і являє собою різні варіанти рішень відповідно до зроблених запитів.

Стратегічна інформація виходить на основі оперативної й тактичної інформації. Призначається для керівництва. Вона складається з коротких підсумків, звітів, прогнозів. На її основі здійснюється довгострокове планування й розробка політики підприємства в цілому.

4 МОЖЛИВОСТІ СУЧАСНИХ ІС

ІС сьогоднішніх світових лідерів в області розробки й впровадження корпоративних інформаційних систем (цей термін буде докладно розглянутий пізніше) надають такі можливості:

— виводять два-три десятки узагальнених показників, що описують поточний стан бізнесу;
— виконують прогнозування й показують динаміку розвитку бізнесу;
— дозволяють моделювати ситуацію в майбутньому на основі поточних показників;
— містять засоби автоматизації управління;
— дають можливість оцінити ризик для різних варіантів рішень керівника;
— забезпечують можливість роботи керівника в режимі віддаленого доступу.

5 ОСНОВНІ ВЛАСТИВОСТІ ІС

Залежно від предметної області ІС можуть досить значно розрізнятися за своїми функціями, архітектурою, реалізацією. Однак можна виділити ряд властивостей, які є загальними:

1) ІС призначені для збирання, зберігання й обробки інформації, тому в основі кожної з них лежить середовище зберігання й доступу до даних;
2) ІС орієнтовані на кінцевого користувача, який не володіє високою кваліфікацією в області обчислювальної техніки. Тому ІС повинні завжди мати простий, зручний та легкий у засвоюванні інтерфейс, що надає кінцевому користувачеві всі необхідні для роботи функції й у той

самий час не дає користувачу можливості виконувати які-небудь зайві дії.

Таким чином, при розробці ІС доводиться вирішувати два основних завдання:

1) розроблення бази даних (БД), призначеної для зберігання інформації;
2) розроблення графічного інтерфейсу користувача.

6 ГОЛОВНІ ГАЛУЗІ ЗАСТОСУВАННЯ ІС

Розглянемо найбільш важливі *завдання, які вирішуються за допомогою ІС*.

1 *Бухгалтерський облік*. Це класична на сьогоднішній день галузь, яка найчастіше застосовує інформаційні технології. Причини: 1) помилка бухгалтера може коштувати дуже дорого, тому вигода автоматизації бухгалтерії є очевидною; 2) завдання бухгалтерського обліку досить легко формалізується, тому розроблення систем автоматизації бухгалтерського обліку не являє собою технічно складної проблеми.

2 *Керування фінансовими потоками.* Впровадження інформаційних технологій у керування фінансовими потоками також обумовлено критичністю цієї області управління підприємством до помилок. Неправильна побудова системи фінансових розрахунків може спровокувати кризу готівки підприємства. І навпаки, коли ця система точно розраховується й жорстко контролюється, то може істотно збільшити оборотні кошти підприємства.

3 *Керування складом, асортиментом, закупівлями.* Автоматизація процесів аналізу руху товару, визначення асортименту допомагає дістати максимальний прибуток при обмежених фінансових ресурсах, дозволяє вчасно виявляти “заморожені” оборотні кошти у складському запасі й правильно працювати з номенклатурою, яка має найбільш динамічний рух.

4 *Управління виробничим процесом.* Оптимальне управління виробничим процесом є дуже трудомістким завданням, де основним механізмом слід вважати планування. Автоматизація дає можливість грамотно планувати, ураховувати витрати, проводити технічну підготовку виробництва, оперативно

управляти виробничим процесом відповідно до виробничої програми й технології.

Очевидно, що чим більше виробництво, тим більше виробничих процесів беруть участь у створенні прибутку, а отже, використання ІС у великому виробництві життєво необхідне.

5 *Управління маркетингом.* Збирання і аналіз даних про конкурентів: їхню продукцію, цінову політику, а також моделювання параметрів зовнішнього оточення для визначення оптимального рівня цін, прогнозування прибутку й планування рекламних компаній. Ці завдання можуть бути формалізовані і вирішені за допомогою ІС.

6 *Документооборот.* Це дуже важливий процес діяльності будь-якого підприємства. Добре налагоджена система обліку обороту документів відбиває поточну виробничу діяльність, що реально відбувається на підприємстві, дає керівникам можливість впливати на неї. Тому тут ІС дозволяє підвищити ефективність управління.

7 *Оперативне управління підприємством.* ІС, що вирішують завдання оперативного управління, будуються на основі БД, у якій фіксується вся можлива інформація про підприємство. Така ІС є інструментом для управління бізнесом усього підприємства й містить у собі безліч програмних рішень з автоматизації процесів у виробництві, що мають місце на конкретному підприємстві.

7 ВИМОГИ, ЩО ВИСУВАЮТЬСЯ ДО ІС

1 *Гнучкість,* тобто здатність ІС до адаптації й подальшого розвитку разом з новими умовами роботи й потребами підприємства. Це можливо, якщо на етапі розробки ІС використовувалися загальноприйняті засоби й методи документування. Це дозволяє згодом розібратися у структурі системи й внести до неї відповідні зміни.

Варто мати на увазі, що психологічно легше розібратися у власних розробках (нехай навіть створених давно), ніж у чужих. Тому рекомендується відразу супроводження системи довіряти особам, які її проектували.

2 *Надійність,* тобто функціонування ІС без випадків перекручування або втрати інформації. Надійність забезпечується створенням резервних копій даних, що зберігаються, виконанням операцій протоколювання, підтримуванням якості каналів зв'язку й фізичних носіїв інформації. Сюди ж варто віднести наявність захисту від випадкових втрат інформації в силу недостатньої кваліфікації персоналу.

3 *Ефективність.* Система є ефективною, якщо з урахуванням виділених їй ресурсів вона дозволяє вирішувати покладені на неї завдання в мінімальний термін.

Ефективність системи забезпечується оптимізацією даних і методів їхньої обробки, застосуванням оригінальних розробок, ідей, методів проектування. Крім того, оскільки працювати з ІС доводиться людям, які є фахівцями у своїй предметній області, але мають середні навички в роботі з комп'ютером, то інтерфейс ІС повинен бути їм інтуїтивно зрозумілий.

4 *Безпека,* тобто властивості ІС обмежувати доступ особам до інформаційних ресурсів організації, крім тих, для яких вони призначені. Захист інформації забезпечується управлінням доступом до ресурсів системи, використанням сучасних програмних засобів захисту інформації, застосуванням паролів і протоколюванням, постійним моніторингом стану безпеки операційних систем і засобів їхнього захисту.

8 ОСНОВНІ ПОНЯТТЯ ТЕОРІЇ ІС

Основні ідеї сучасних інформаційних технологій спираються на концепції баз даних. Відповідно до цього основою інформаційної технології є дані, які організовані в БД із метою відображення реального стану предметної області й задоволення потреб користувача.

Дані — це відомості про деяку сутність, які зафіксовані у вигляді значень і зберігаються на деяких носіях. Дані перетворюються в змістовну інформацію, коли вони осмислено оброблені й подані в потрібний час у потрібне місце.

Предметна область — набір об'єктів, що мають інтерес для актуальних або передбачуваних користувачів, коли реальний світ

відображається сукупністю конкретних і абстрактних понять, між якими фіксуються певні зв'язки.

База даних — іменована сукупність взаємозалежних даних, яка відображає стан об'єктів і їхніх відносин у деякій предметній області, яка використовується декількома користувачами й зберігається з мінімальною надлишковістю.

Система управління базою даних (СУБД) — це комплекс мовних і програмних засобів, що призначений для створення, ведення й спільного використання БД багатьма користувачами.

Вище вже розглядалися тлумачення терміна ІС. Наведемо загальне визначення ІС, яке подається у відповідній довідковій літературі.

Інформаційна система — це система, що реалізує автоматизоване збирання, зберігання й обробку даних і яка включає в себе технічні засоби, програмне забезпечення й відповідний персонал.

Банк даних (БнД) — це заснована на технології баз даних система спеціально організованих даних, а також програмних, мовних, організаційних і технічних засобів, призначених для централізованого накопичення й колективного багатоцільового використання даних.

БнД, по суті, є різновидом ІС, у яких реалізовані функції централізованого зберігання й накопичення оброблюваної інформації, організованої в одну або кілька баз даних.

Вирішенням проблеми в обробці даних є побудова БД. Однак бази даних можуть різнитися в принципах свого проектування і тому в більшому або в меншому ступені задовольняти розв'язання конкретної проблеми обробки даних.

Головну роль у проектуванні баз даних відіграє вибір *способу організації даних*.

9 МОДЕЛІ ОРГАНІЗАЦІЇ ДАНИХ

Збережені в базі дані мають певну логічну структуру, іншими словами, описуються деякою моделлю уявлення (або організації) даних (моделлю даних), що підтримується СУБД. До числа класичних і таких, що найбільш часто застосовуються, належать три моделі даних:

1) ієрархічна;
2) мережна;
3) реляційна.

Крім того, в останні роки з’явилися й стали впроваджуватися на практиці ще такі моделі даних: постреляційна модель, багатомірна модель, об’єктно-орієнтована модель. Розробляються також різні системи, побудовані на моделях даних, що комбінують відомі моделі. Є СУБД, що підтримують одночасно кілька моделей даних.

Тут ми розглянемо тільки три перші з названих моделей, як найпоширеніші.

Ієрархічна модель. У цій моделі зв’язки між даними можна описати за допомогою впорядкованого графа (або дерева). Спрощене уявлення зв’язків між даними в ієрархічній моделі можна зобразити так, як це зроблено на рисунку 2.

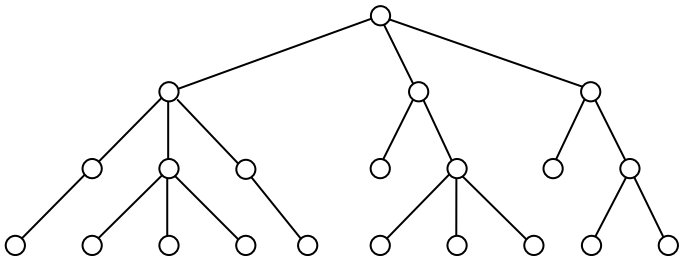


Рисунок 2

В ієрархічній моделі даних присутні два типи даних: тип “дерево” (“вузол”) і тип “запис” (“лист”).

Тип “дерево” є типом складеним. Він містить у собі деякі підлеглі дані, які, у свою чергу, можуть бути даними двох типів — типу “дерево” і типу “запис”. Тип “запис” може бути простим або складеним. У першому випадку це будь-яке значення (числове, символьне, логічне), у другому випадку — являє собою сукупність значень. Підлеглі дані є “нащадками” стосовно даних, які виступають для них у ролі “предків” (“батьків”). Існує ще кореневий тип даних, які мають підлеглі дані, але самі не є підлеглими.

Ієрархічна БД являє собою впорядковану сукупність екземплярів даних типу “дерево”, деякі з яких мають підлеглими екземпляри типу “запис”.

Достоїнства ієрархічної моделі даних — це ефективне використання пам’яті ЕОМ і непогані показники часу виконання основних операцій над даними. Ієрархічна модель даних зручна для роботи з ієрархічно впорядкованою інформацією.

Недоліками ієрархічної моделі є її громіздкість для обробки інформації з досить складними логічними зв’язками, а також складність розуміння для звичайного користувача.

Мережна модель. Ця модель даних дозволяє відображати різноманітні взаємозв’язки елементів даних у вигляді довільного графа, узагальнюючи тим самим ієрархічну модель даних. На рисунку 3 зображений спрощений приклад таких зв’язків між даними в мережній моделі.

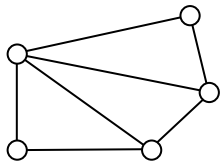


Рисунок 3

Для опису схеми мережної БД використовують дві групи типів даних: “записи” і “зв’язки”. Екземпляр даних типу “зв’язок” визначається для двох екземплярів типу “запис”. Дані типу “запис” мають структуру, аналогічну “записам” ієрархічної моделі. Мережна БД складається з набору записів і набору відповідних зв’язків. На формування зв’язків особливих обмежень не накладається. Якщо в ієрархічній структурі запис-“нащадок” міг мати тільки одного “предка”, то в мережній моделі даних запис-“нащадок” може мати довільне число записів-“предків” (“зведених батьків”).

Достоїнством мережної моделі даних є можливість її ефективної реалізації за показниками витрат пам'яті та оперативності. У порівнянні з ієрархічною моделлю мережна модель надає більше можливостей у розумінні допустимості утворення довільних зв’язків.

Недоліком мережної моделі даних є висока складність і твердість схеми БД, що побудована на її основі. Набори зв’язків і структуру записів доводиться задавати наперед. Зміна структури даних означає перебудову всієї БД. Крім того, мережна БД складна для розуміння та виконання обробки інформації для звичайного користувача і, через можливість установлення довільних зв’язків, має слабкий контроль цілісності даних, тобто збільшується ймовірність помилок при сприйнятті даних і їхній обробці.

Реляційна модель. Ця модель даних ґрунтується на понятті відношення. Відношення являє собою множину елементів, що називаються кортежами. Кожний кортеж являє собою однотипний набір описів конкретного екземпляра деякого об’єкта (сутності, явища тощо) із предметної області, для якої будується БД.

Наочною формою уявлення відношення є звична для людського сприйняття прямокутна таблиця. Таблиця має рядки (записи) і стовпці. Кожний рядок таблиці має однакову структуру й складається з полів. Поля зберігають значення описів об’єкта. Рядки таблиці відповідають кортежам відношення, а кожний стовпець містить у собі значення якого-небудь одного опису, що характеризує об’єкт, для якого задане відношення.

За допомогою однієї таблиці зручно описувати велику кількість зв’язків між даними, а саме розподіл одного об’єкта на безліч його екземплярів. Але, оскільки в рамках однієї таблиці не вдається описати більш складні логічні структури даних із предметної області, у реляційних БД задають ще зв’язування таблиць.

Достоїнство реляційної моделі даних полягає у простоті, зрозумілості та зручності фізичної реалізації на ЕОМ. Саме простота та зрозумілість для користувача стали основною причиною їхнього широкого використання. Крім того, незалежність даних у реляційній структурі дозволяє реляційним БД зростати й перетворюватися без перебудови накопиченої в них раніше інформації.

Основним недоліком реляційної моделі є відсутність стандартних засобів ідентифікації окремих записів, а також складність опису ієрархічних і мережних зв'язків.

Про широке розповсюдження та популярність реляційних СУБД свідчить факт, що майже всі закордонні СУБД для ПЕОМ, що використовуються в усьому світі, є реляційними. До них належать: dBase III Plus і dBase IV (фірма Ashton-Tate), DB2 (IBM), RBase (Microrim), FoxPro ранніх версій і FoxBase (FoxSoftware), Paradox і dBase for Windows (Borland), FoxPro більш пізніших версій, Visual FoxPro і Access (Microsoft), Clarion (Clarion Software), Ingres (ASK Computer Systems) і Oracle (Oracle).

До українських СУБД реляційного типу належить система ПАЛЬМА (ІК АН України), до російських — система НуТech (МІФІ).

10 КЛАСИФІКАЦІЯ ІС

ІС класифікуються за різними ознаками. Найчастіше використовуються такі класифікації:

1) за масштабом;
2) за сферою застосування;
3) за способом організації.

За масштабом ІС поділяються на такі групи:

1) одиночні;
2) групові;
3) корпоративні.

Одиночні ІС реалізуються, як правило, на автономному персональному комп'ютері (тобто мережа не використовується). Така система може містити кілька простих додатків, зв'язаних загальним інформаційним фондом. Вона розрахована на роботу одного користувача або групи користувачів, що розділяють за часом одне робоче місце.

Групові ІС орієнтовані на колективне використання інформації членами робочої групи й будуються на базі локальної обчислювальної мережі.

Корпоративні ІС є розвитком систем для робочих груп. Вони орієнтовані на великі компанії й можуть підтримувати територіально рознесені вузли або мережі. В основному вони мають ієрархічну структуру з декількох рівнів.

За сферою застосування ІС звичайно поділяються на чотири групи:

1) системи обробки транзакцій;
2) системи підтримки прийняття рішення;
3) інформаційно-довідкові системи;
4) офісні інформаційні системи.

Системи обробки транзакцій (транзакціями називають послідовності операцій над БД, які СУБД простежує як єдине ціле, тобто як одну “подію”) застосовуються для відображення реального стану предметної області в будь-який момент часу.

Системи підтримки прийняття рішення становлять такий тип ІС, у яких за допомогою досить складних запитів виконується відбір і аналіз даних у різних площинах: часових, географічних або за іншими показниками.

Інформаційно-довідкові системи засновані на обробці гіпертекстових документів і мультимедіа. Найбільший розвиток такі ІС отримали в мережах Інтернет.

Офісні ІС використовуються для переведення паперових документів до електронного вигляду, автоматизації діловодства та управління документооборотом.

За способом організації групові та корпоративні ІС поділяються на такі класи:

1) системи на основі архітектури файл-сервер;
2) системи на основі архітектури клієнт-сервер;
3) системи на основі багаторівневої архітектури;
4) системи на основі Інтернет і інтранет-технологій.

Ці ІС являють собою обчислювальні мережі, більшість вузлів яких є клієнтськими компонентами системи, й тільки невелика частина — серверними компонентами. Користувачі такої ІС звертаються до неї через клієнтські компоненти. Дані про предметну область розміщуються в серверних компонентах.

Щоб зрозуміти організаційні розбіжності в архітектурах ІС, у будь-якій ІС виділяють кілька функціональних складових:

— компоненти діалогу (вони поділяються на обслуговування подання даних і логіку подання даних);
— компоненти обробки даних (вони поділяються на прикладну логіку й логіку оперування даними);
— компоненти управління даними (вони поділяються на обслуговування баз даних та обслуговування файлової системи).

При архітектурі ІС по типу файл-сервер поділ компонентів діалогу відсутній, і клієнтські комп'ютери виконують всі функції з уведення й відображення даних та їх прикладної обробки. Файл-сервер тільки витягує дані з файлів БД. Основне навантаження з оперування даними лежить на клієнтах. Кожний новий клієнт додає обчислювальну потужність до мережі (це — достоїнство такої архітектури). Однак при виконанні деяких запитів клієнтові можуть передаватися дуже великі обсяги даних, що призводить до надмірного завантаження мережі й непередбачуваності часу реакції (це — істотний недолік).

Коли ІС організована по типу клієнт-сервер, то в ній існує дворівнева обробка даних. Компоненти забезпечення діалогу та частина компонентів обробки даних (логіка оперування даними) тут, як і раніше, є функціями клієнта. Але для скорочення навантаження на мережу та спрощення адміністрування компоненти прикладної логіки розміщуються на сервері.

Багаторівнева архітектура ІС має на увазі три і більше рівневу обробку даних. Організація діалогу належить клієнтові. Однак прикладна логіка й логіка оперування даними здійснюється на середньому рівні спеціальними серверами додатків. Верхній рівень, звичайно, являє собою віддалений спеціалізований сервер БД, який виділено для послуг з управління даними.

Інтернет/інтранет архітектура характеризується функціональною різноманітністю серверів. Клієнтським комп'ютерам залишається тільки обслуговування подання даних, а все інше беруть на себе різні сервери мережі.

11 ПРОЕКТУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

11.1 Базові концепції. Поняття реляційної БД

Теорія реляційних БД у своїх основах була розроблена Едгаром Ф. Коддом, відомим американським фахівцем в області БД. Основні концепції її були вперше опубліковані ним у 1970 р.

Будучи математиком за освітою, Кодд запропонував використовувати для обробки даних апарат теорії множин. Він показав, що будь-яке зображення даних зводиться до сукупності двовимірних таблиць особливого виду, відомих у математиці як відношення (англійською — relationship, звідси й назва — реляційні бази даних). Ця теорія зараз використовується в алгоритмічних методах проектування БД.

БД можна визначити як уніфіковану сукупність даних, якою спільно користується увесь персонал підприємства. Завдання БД — збереження всіх необхідних даних в одному місці.

Дані деякої предметної області в реляційній БД являють собою набір відношень, що змінюються у часі.

Математично відношення можна визначити таким способом. Нехай дано n множин: $D_1, D_2, D_3, \ldots, D_n$, тоді R є *відношення* над цими множинами, якщо R являє собою деяку множину із m кортежів вигляду $\left\langle \left\{a_1, d_1^i\right\}, \left\{a_2, d_2^i\right\}, \ldots, \left\{a_n, d_n^i\right\} \right\rangle, \quad i = \overline{1, m},$ де $d_k^i \in D_k$, $k = \overline{1, n}$. Тобто, елемент d_1^i належить множині D_1, елемент d_2^i належить множині D_2 і т.д. Останній елемент d_n^i належить множині D_n. Елементи кортежу $a_1, a_2, \ldots, a_n$ — це є *імена атрибутів*, множини $D_1, D_2, D_3, \ldots, D_n$ називають *доменами*, їхні елементи $d_1^i, d_2^i, \ldots, d_n^i \left(i = \overline{1, m}\right)$, що входять до усіх кортежів відношення R, є *значеннями атрибутів*, імена яких задані величинами $a_1, a_2, \ldots, a_n$. Натуральне число n називають *ступенем відношення* R, число m — *кардинальністю відношення* R (ще — *потужністю* R).

Розглянемо, як ці абстрактні поняття відображають таблиці реляційної БД, з якою так зручно працювати користувачеві.

Щоб створити БД, потрібно спочатку провести дослідження предметної області. Змістовно предметна область завжди являє собою систему об’єктів якогось реального або віртуального середовища (оточення), що взаємодіють між собою за допомогою різноманітних зв’язків. Об’єкти (предмети і явища) різняться між собою, але з них завжди можна виділити групи однорідних за своєю суттю об’єктів, які описуються однаковим набором властивостей і які відрізняються один від одного у групі тільки значеннями (чисельними або якісними) цих властивостей. Такі групи однорідних об’єктів із розглянутої предметної області називають *сутностями*. Властивості, сукупність значень яких характеризують кожний об’єкт (або екземпляр) сутності, називають *атрибутом* сутності. Кожне відношення відображає в БД одну якусь конкретну сутність у вигляді набору списків (кортежів) значень атрибутів, що описують екземпляри цієї сутності.

Таким чином, *сутність* — це сукупність однорідних об’єктів будь-якої природи, дані про які зберігаються у відношенні.

Атрибути являють собою властивості, що характеризують екземпляри сутності. У структурі відношення кожний атрибут іменується.

Домен являє собою множину усіх можливих значень атрибута відношення. Дані у відношеннях вважаються порівнянними тільки в тому випадку, якщо вони належать до одного домену. Якщо ж значення двох атрибутів беруться з різних доменів, то їхнє порівняння, у загальному випадку, позбавлено змісту.

Список імен атрибутів, що входять у структуру відношення (набір $a_1, a_2, \dots, a_n$ у нашому визначенні відношення R), називають *схемою* (або *заголовком*) відношення.

Кортеж, що відповідає даній схемі відношення, являє собою множину пар {*ім’я атрибута*, *значення*}, кожна з яких містить одне конкретне ім’я атрибута, що належить схемі відношення, і одне значення цього атрибута, що належить відповідному домену.

Множину кортежів, що належать до одного відношення, часто називають *вмістом* (або *тілом*) цього відношення.

У загальному випадку порядок кортежів у відношенні, як і в будь-якій множині, не визначений. Однак у реляційних СУБД для зручності роботи кортежі впорядковують. Для цього вибирається деякий атрибут. Якщо користувач не призначає атрибут для упорядкування, система автоматично привласнює номери кортежам у порядку їхнього уведення до відношення.

Формально, якщо переставити атрибути у відношенні, то виходить нове відношення. Однак у реляційних БД перестановка атрибутів не приводить до утворення нового відношення.

Користувачеві будь-яке відношення, що входить у реляційну БД, подається графічно двовимірною таблицею. Вкажемо відповідність між елементами теоретико-множинного уявлення реляційної моделі даних і поняттями-аналогами, що використовуються в описах до роботи із реляційними СУБД. Для цього розглянемо приклад табличного подання відношення "Співробітник" із БД деякого підприємства (рисунок 4).

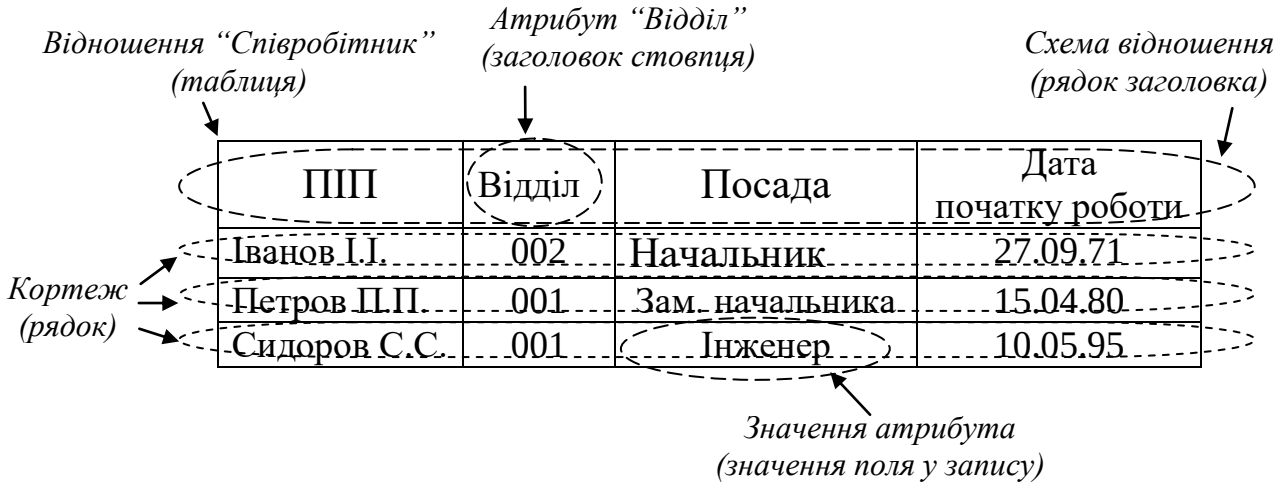


Рисунок 4

Відношення "Співробітник" пов'язане з чотирма доменами. Домен 1 містить прізвища всіх співробітників, домен 2 – номери всіх відділів підприємства, домен 3 – назви всіх посад, які є на підприємстві, домен 4 – дати початку роботи усіх співробітників підприємства.

Зауваження: В багатьох реляційних СУБД поняття домену не використовується. У таких випадках вважається достатнім, щоб усі значення одного атрибута у відношенні належали до одного типу даних (наприклад, були тільки числовими або тільки символьними). Однак це є спрощенням поняття домену для зручності користувачів.

Відношення “Співробітник” має три кортежі. У таблиці відношення значення атрибутів одного кортежу заносяться в один рядок. Кожному кортежу відповідає свій рядок у таблиці відношення. Кортеж відношення, таблиця якого наведена на рисунку 4, складається з 4 елементів —значень атрибутів, кожний з яких вибирається з відповідного домену.

Імена атрибутів з відношення розміщаються в заголовному (верхньому) рядку таблиці. Нижче імені атрибута у тому ж стовпці таблиці розміщуються всі значення атрибута, що входять у кортежі відношення. Таким чином, завжди значення атрибутів з одного стовпця таблиці належать одному домену.

Часто у посібниках до роботи з реляційними СУБД рядки таблиці, що описує відношення, називають *записами*, комірки, куди заносяться значення атрибутів, називають *полями*, а верхній рядок таблиці, де зберігаються імена атрибутів, називають *шапкою* таблиці. Іноді під терміном “поле” розуміється цілий стовпець таблиці, де повинні зберігатися значення одного домену.

На основі сказаного стає зрозуміло, що *реляційна БД* являє собою набір відношень, які вміщують усю інформацію про предметну область, необхідну для функціонування підприємства.

11.2 Індекси

У більшості СУБД кожне відношення із БД зберігається в окремому *файлі* (тобто в іменованому наборі даних, що записаний на зовнішньому носії інформації в ЕОМ). У великих СУБД кожне відношення зберігається в *індексованому файлі*.

Під *індексом* розуміють засіб прискорення операції пошуку записів у таблиці-відношенні, а отже, й інших операцій, які використовуються в БД і де є пошук даних.

Індекс можна уявити як таблицю, з якою пов’язана процедура, що сприймає на вході інформацію про значення деяких атрибутів із відношення та видає на виході інформацію, яка сприяє швидкому визначенню місця розташування на носії запису або записів, що мають задані значення атрибутів.

Пояснимо ідею індексування на прикладі. Нехай у деякої реляційної БД є відношення, що має кілька десятків атрибутів, і припустимо, що для подальшої обробки даних потрібно його кортежі впорядкувати за зростанням значень деякого числового атрибута А. При упорядкуванні елементів у послідовності потрібно багаторазово міняти їх місцями один з одним. Якщо таку процедуру проводити безпосередньо над записами з таблиці нашого відношення, які самі нараховують у собі десятки елементів-значень, то виявиться, що треба зробити занадто велику кількість операцій, які вимагають багато часу та інших ресурсів. У таких випадках застосовуються індекси. Будується таблиця, що за кількістю рядків (потужністю) дорівнює нашому відношенню. Кожний рядок цієї таблиці має вигляд, як показано на рисунку 5,

Значення А	*Адреса початку*

Рисунок 5

де “Значення А” — значення атрибута А одного із записів таблиці розглянутого відношення, “Адреса початку” — адреса місця розташування початку цього ж запису на фізичному носії. Кожному запису таблиці нашого відношення буде відповідати рядок таблиці індексу.

Здійснюється необхідне упорядкування рядків таблиці індексу, при якому буде зроблено в десятки разів менше операцій перестановок елементів, ніж було б зроблено при упорядкуванні безпосередньо таблиці розглянутого відношення.

Для апарату СУБД функціонально однаково, яка із двох таблиць була впорядкована, оскільки завжди перед використанням даних спочатку визначається їхня фізична адреса.

У прикладі був розглянутий простий спосіб завдання індексу. Існують і більш складні способи. Але скрізь головна причина підвищення швидкості виконання різних процедур з індексованими таблицями полягає в тому, що основна частина роботи здійснюється з невеликими індексами, а не із самими таблицями.

Повернемося до розгляду структури відношення в реляційній БД.

11.3 Порожні значення атрибутів

Атрибут сутності, що задана відношенням у реляційній БД, у кожному кортежі відношення представлений своїм значенням. Однак у деяких випадках будь-який атрибут у кортежі може не мати конкретного значення. Наприклад, нехай у відношенні “Співробітник” із БД підприємства є ще один атрибут: “Номер домашнього телефону”. Значення цього атрибута в деяких кортежах може бути відсутнім, якщо є на підприємстві співробітники, які не мають стаціонарного телефону у себе дома. Варто розуміти, що порожнє значення атрибута — це не нуль і не порожній рядок, а значення, що не визначене в даний момент часу, але яке, взагалі кажучи, може бути визначене пізніше.

Для позначення порожніх значень полів у записах таблиць із БД використовується універсальне *недійсне значення* NULL.

11.4 Первинний ключ відношення

Оскільки відношенням з математичної точки зору є множина, що не має однакових елементів, то ніякі два кортежі у відношенні не можуть бути дублікатами один одного в будь-який довільно заданий момент часу. Таким чином, у відношенні завжди повинен бути присутнім деякий атрибут (або набір атрибутів), що однозначно ідентифікує кожний з кортежів. Такий атрибут (або набір атрибутів) називається *первинним ключем відношення*. Для кожного відношення властивість унікальності має принаймні повний набір його атрибутів. Однак первинний ключ повинен складатися з мінімально необхідного набору атрибутів, тобто не бути надлишковим.

Більш строго визначити поняття первинного ключа можна так. Якщо R — відношення з атрибутами A_1, A_2, ... , A_n, то підмножина атрибутів $K = (A_i, A_j, \ldots , A_k)$ відношення R буде первинним ключем цього відношення тоді, і тільки тоді, коли задовольняються дві незалежних від часу умови:

1) унікальність — у довільний момент часу ніякі два кортежі відношення R не збігаються за комбінаціями значень атрибутів із K;
2) мінімальність — жоден з атрибутів $A_i, A_j, \ldots , A_k$ не може бути виключений з K без порушення унікальності.

Можливі випадки, коли відношення має кілька комбінацій атрибутів, кожна з яких однозначно визначає всі кортежі відношення. Всі ці комбінації атрибутів є *можливими* (або *потенційними*) ключами відношення. Кожний з можливих ключів може бути обраний як первинний.

Розрізняють ключі *прості* (вони складаються з одного атрибута) і *складені* або *складні* (вони складаються з декількох атрибутів).

Серед складених ключів одного відношення можуть зустрічатися *ключі, що перекриваються*, — складені ключі, які мають один або кілька спільних атрибутів.

11.5 Роль ключів в організації реляційних БД

Ключі використовуються для досягнення таких цілей:

1) для виключення дублювання значень у кортежах по ключових атрибутах (інші атрибути в розрахунок не приймаються);
2) для упорядкування кортежів (це зручно робити за значеннями ключових атрибутів, які не повторюються);
3) для прискорення роботи з кортежами (зручно із ключовими атрибутами задавати індекси);
4) для організації зв’язування таблиць, які описують відношення.

Останнє розглянемо докладніше.

11.6 Зв’язані відношення

У реляційній моделі дані подаються у вигляді сукупності взаємозалежних відношень (таблиць). Зв’язок між відношеннями — це ще одне важливе поняття в реляційній моделі даних. Для його пояснення введемо ще одне визначення.

Нехай у відношенні *R1* є неключовий атрибут *A*, значення якого є значеннями ключового атрибута *B* іншого відношення *R2*. Тоді говорять, що атрибут *A* відношення *R1* є *зовнішнім ключем*. Тобто зовнішній ключ — це атрибут (або множина атрибутів) одного відношення, що є ключем іншого відношення.

Зовнішні ключі використовуються для встановлення логічних зв’язків між відношеннями. У реляційній БД *зв’язок між двома таблицями*, що описують два відношення, установлюється шляхом присвоювання значень зовнішнього ключа однієї таблиці значенням ключа іншої.

11.7 Види зв’язків таблиць реляційної БД

При зв’язуванні двох таблиць виділяють *основну* й *додаткову (підлеглу)* таблиці.

Існує чотири види зв’язку:

1) ”один – один” (1:1). В основній таблиці поля зв’язку є ключ — у додатковій таблиці поля зв’язку теж є ключ;
2) ”один – багато” (1:M). В основній таблиці поля зв’язку є ключ, а в додатковій таблиці поля зв’язку не є ключем;
3) ”багато – один” (M:1). В основній таблиці атрибути для зв’язку не ключ — у додатковій таблиці атрибути для зв’язку є ключем.
4) ”багато – багато” (M:M). В основній таблиці атрибути для зв’язку не ключ — у додатковій таблиці атрибути для зв’язку також не є ключем.

Зовнішні ключі теж можуть бути простими й складеними.

Але для зв’язків використовувати складені ключі часто буває незручно. Тоді у відношення вводиться *штучний (сурогатний)* ключ. Штучний ключ завжди простий і являє собою формальний ідентифікатор (як правило, це номер) запису.

11.8 Умови цілісності даних

Інформація, що зберігається в БД, повинна бути однозначною та несуперечливою. Тому в реляційній моделі використовуються деякі *обмежувальні умови*.

Обмежувальні умови — це правила, що визначають можливі (дозволені для БД) значення даних. Вони забезпечують логічну основу для підтримки коректних значень даних у базі. Їх ще називають *обмеженнями цілісності*. Вони дозволяють звести до мінімуму помилки, що виникають при обробці й відновленні даних.

Найважливішими обмеженнями цілісності є:

1) категорійна цілісність;
2) посилальна цілісність.

Категорійна цілісність. Кортежі відношення представляють у БД об'єкти реального світу (тобто категорії). Наприклад, згадаємо відношення "Співробітник", де кожний кортеж описує конкретного співробітника підприємства. Первинний ключ таблиці однозначно визначає кожний кортеж. Тому для добування даних із БД про конкретний об'єкт або маніпулювання цими даними треба знати значення ключа відповідного запису. Правило категорійної цілісності: запис про об'єкт не може бути занесений до бази даних доти, поки не будуть визначені всі атрибути його первинного ключа. Є й коротке формулювання: поле ніякого атрибута первинного ключа у запису не може бути порожнім.

Посилальна цілісність. Її правило: якщо дві таблиці, що представляють відношення, зв'язані між собою, то зовнішній ключ однієї таблиці повинен містити тільки значення, що уже наявні серед значень ключа в іншій таблиці, за якою здійснюється зв'язок. Якщо цього правила не дотримуватися і не контролювати коректність зовнішніх ключів, то порушиться цілісність даних. Приклад: нехай у базі підприємства крім відношення "Співробітник" є пов'язане з ним відношення "Грошове нарахування". Якщо співробітник був звільнений (його кортеж видалили з відношення "Співробітник"), то в таблиці "Грошове нарахування", якщо не зробити відповідні корективи, залишаться дані про зарплату співробітника, що вже пішов з

підприємства. Така ж ситуація буде, якщо зовнішньому ключу таблиці “Грошове нарахування” помилково буде привласнене значення, що відсутнє в значеннях ключа зв’язаної таблиці.

Вирішення проблеми посилальної цілісності в СУБД:

1) при уведенні нових і модифікації кортежів, які вже є, стежать, щоб не з’явилися некоректні значення зовнішнього ключа;
2) при видаленні з відношення кортежу, на який вказує посилання, використовують один із трьох підходів:
    а) забороняється видаляти кортеж, на який існує посилання;
    б) при видаленні кортежу, на який є посилання, у всіх кортежах, що мають на нього посилання, значення зовнішнього ключа стають невизначеними;
    в) при видаленні з відношення кортежу, на який посилаються, з відношення, що посилається на нього, повинні видалятися всі кортежі, що посилаються (це *називається каскадним* видаленням).

11.9 Цілі проектування БД

Серед багатьох цілей, які переслідуються при проектуванні БД, виділимо основні:

1) можливість збереження всіх необхідних даних в одній БД;
2) виключення надмірності даних;
3) зведення кількості відношень у БД до мінімуму;
4) нормалізація БД для виключення проблем, пов’язаних з відновленням і видаленням даних.

Прокоментуємо ці цілі:

Ціль 1. Передбачається, що БД повинна вміщати всі дані, що становлять інтерес для підприємства. І тому першим кроком у проектуванні БД є визначення всіх атрибутів, які будуть поміщені в БД. Причому намагаються також уводити в БД атрибути “на перспективу”, тобто й такі, які на сучасний момент не дуже важливі для керування й роботи підприємства.

Тільки після цього визначаються кількість відношень, що входять у БД, яка проектується, і розподіляються між ними атрибути.

Ціль 2. БД повинна проектуватися так, щоб уміщати в себе якнайбільше потрібної інформації й щоб для цього було витрачено якнайменше місця на фізичному носії. А цього неможливо досягти, якщо дані, які зберігаються в БД, будуть у ній дублюватися. Крім того, дублювання даних може призвести до проблем при обробці даних у БД.

Але варто розрізняти *необхідне дублювання* даних (або *просте*, або *ненадлишкове*) і *надлишкове дублювання* даних. Перше може бути присутнім у БД, друге — треба виключати.

Пояснимо на прикладах різницю між необхідним і надлишковим дублюванням даних.

Припустимо, що є в нас таке відношення “Співробітник-Телефон” (див. рисунок 6).

Співробітник	**Телефон**
Іванов І.І.	12-34
Петров П.П.	56-78
Сидоров С.С.	56-78

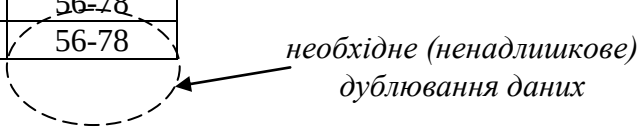


Рисунок 6

Тут є дублювання даних: номери телефонів. Але це необхідне (ненадлишкове) дублювання. Співробітники (2-й і 3-й кортежі) працюють в одному приміщенні, тому номери телефонів збігаються, хоча для кожного співробітника номер телефону є унікальним.

Розглянемо ще одне відношення “Співробітник-Кімната-Телефон” у двох варіантах, що подані на рисунку 7.

а)

Співробітник	Кімната	Телефон
Іванов І.І.	101	12-34
Петров П.П.	105	56-78
Сидоров С.С.	105	56-78

надлишкове дублювання даних

б)

Співробітник	Кімната	Телефон
Іванов І.І.	101	12-34
Петров П.П.	105	56-78
Сидоров С.С.	105	— // —

Рисунок 7

У варіанті а) є надлишкове дублювання даних (номери телефонів у 2-му і 3-му кортежах). Співробітники Петров і Сидоров працюють в одній кімнаті. Природно припустити, що всі співробітники, що працюють в одній кімнаті, мають один спільний телефонний номер. Тому про телефонний номер Сидорова можна довідатися з кортежу з відомостями про Петрова.

Звернемося до варіанта б). Тут замість номера телефону Сидорова стоїть прочерк, тобто недійсне (невизначене) значення. Це треба вважати невдалим способом уникнути надлишкового дублювання даних за такими причинами: 1) при побудові СУБД потрібно витратити додаткові зусилля на створення механізму пошуку інформації для прочерків таблиці; 2) місце на фізичному носії все однаково буде виділятися під значення атрибута, де є прочерк, хоча дублювання даних і виключено; 3) це дуже важливо, при звільненні Петрова кортеж з відомостями про нього буде виключений з відношення і, виходить, буде знищена інформація про телефон кімнати, де він працював, що є неприпустимим.

Можливий вихід з даної ситуації — подати відношення “Співробітник-Кімната-Телефон” у вигляді двох відношень: “Співробітник-Кімната” і “Кімната-Телефон” (див. рисунок 8).

Кімната	Телефон
101	12-34
105	56-78

Співробітник	Кімната
Іванов І.І.	101
Петров П.П.	105
Сидоров С.С.	105

Рисунок 8

Ціль 3. З пояснення цілі 2 ми бачимо, що відношення іноді має сенс розбивати на два відношення або більше, щоб позбутися надлишкового дублювання даних. Однак велика кількість відношень у БД робить її незручною у використанні та незрозумілою простому користувачеві. Тому не можна допускати необдуманого збільшення кількості відношень у БД при її проектуванні.

Ціль 4. Ця ціль (нормалізація відношень у БД) теж тісно пов'язана з ціллю 2. При поясненні цілі 2 було показано, як невдале рятування від надлишкового дублювання даних може призвести до втрати даних. Подібні ситуації, що пов'язані з надлишковим дублюванням даних, Едгар Ф. Кодд назвав "аномаліями відновлення відношення". Він показав, що для деяких відношень проблеми виникають при спробі видалення, додавання або редагування їхніх кортежів. Зараз у літературі з проектування БД *аномаліями* називають такі ситуації в таблицях БД, які призводять до протиріч у БД або істотно ускладнюють обробку даних.

Проектування БД повинне допомагати вчасно визначати аномалії у відношеннях і нормалізувати їх.

Нормалізація являє собою процес реорганізації відношень у БД шляхом ліквідації в них груп даних, які повторюються, і інших протиріч із метою приведення відношень у БД до вигляду, що дозволяє здійснювати коректну обробку даних.

11.10 Метод нормальних форм

Основною задачею, що розв'язується у процесі проектування БД, є задача нормалізації відношень. Розглянемо метод нормальних форм, який слід вважати класичним методом проектування реляційних БД.

Процес проектування БД із використанням методу нормальних форм полягає в послідовному переведенні відношень з однієї нормальної форми в інші нормальні форми більш високого порядку за певними правилами. Кожна нормальна форма забезпечує усунення конкретного типу аномалій при виконанні операцій над відношеннями БД і зберігає властивості нормальних форм більш низького порядку. Процедура, за

допомогою якої здійснюються переходи відношень від форми до форми, називають "*декомпозицією без втрат*". Ця процедура являє собою розкладання відношення на два або більше відношень, де принаймні одне зі знову отриманих відношень перебуває в нормальній формі більш високого порядку.

Застосовується така послідовність нормальних форм:

1) перша нормальна форма (1НФ);
2) друга нормальна форма (2НФ);
3) третя нормальна форма (3НФ);
4) посилена третя нормальна форма або нормальна форма Бойса-Кодда (БКНФ);
5) четверта нормальна форма (4НФ);
6) п'ята нормальна форма або нормальна форма проекції-з'єднання (5НФ).

Теоретично процес нормалізації повинен тривати доти, поки БД не буде складатися тільки з відношень, що перебувають в 5НФ. Але на практиці, як правило, обмежуються перетвореннями відношень до БКНФ або навіть зупиняються на 3НФ. У більшості випадків цього досить, щоб всі операції, які реально потрібні при роботі з БД, здійснювалися коректно. Більше того, деякі фахівці в області проектування БД вважають, що досить привести таблиці БД до БКНФ, і це гарантує те, що вони перебуватимуть у 5НФ. Це твердження має потребу в перевірці, але поки не існує ефективного способу такої перевірки.

Розглядаючи докладніше процес нормалізації відношень, не будемо підійматися вище БКНФ.

При реалізації процесу нормалізації застосовуються два підходи, що рівносильні між собою.

Перший підхід полягає в такій послідовній декомпозиції відношень, щоб усі знову отримані відношення перебували у нормальній формі більш високого порядку.

Другий підхід відрізняється від першого тим, що намагаються щоразу проводити декомпозицію відношень так, щоб у результаті хоча б одне зі знову отриманих відношень перебувало б у БКНФ. Це відношення далі вже не змінюють, а процес декомпозиції проводять із рештою.

Обидва ці варіанти нормалізації мають одну початкову точку — *універсальне відношення*, і спираються на одну загальну основу — поняття *функціональної залежності*.

11.10.1 Універсальне відношення

Проектування БД починається з визначення всіх об’єктів, відомості про які будуть включені до бази, і визначення їхніх атрибутів. Потім всі ці атрибути зводяться в одну загальну таблицю — універсальне відношення.

Пояснювати це зручно на прикладі. Нехай нам необхідно спроектувати БД, наприклад, для куратора курсу в академії, тобто викладача, якому, насамперед, треба знати успішність кожного студента на курсі, а ще де і як студента можна знайти поза заняттями. Для простоти будемо вважати, що всі студенти живуть в одному гуртожитку. Однак, нехай у кожній кімнаті гуртожитку є телефон.

Визначаємо всі ті показники, властивості, характеристики (одним словом — атрибути), значення яких кураторові треба знати, щоб мати уявлення про успіхи курсу й про те, як при необхідності відшукати будь-якого студента. Атрибути тут визначити легко, оскільки ми добре знаємо предметну область. Перелічимо імена цих атрибутів і дамо кожному короткий опис. НомСт — номер студента (ціле додатне число, служить для ідентифікації студента у списку або у БД, воно є унікальним для кожного студента). ПІБ — прізвище студента і його ініціали (допускається, що у декількох студентів можуть бути однаковими прізвища та ініціали). НомКом — номер кімнати в гуртожитку (допускається, що в одній кімнаті може проживати декілька студентів). НомТел — номер телефону студента (всі студенти, що проживають в одній кімнаті, користуються одним телефоном). Предм — назва предмета (зберігаються в базі дані тільки про той предмет, вивчення якого студент закінчив). Сем — номер семестру, у якому студент закінчив вивчення предмета. Оцінка — підсумкова оцінка за предмет.

При проектуванні реальної БД перший етап являє собою дуже кропітку роботу: за результатами спілкування із замовником необхідно визначити повний перелік відомостей,

якими замовник буде користуватися, і визначити, яку інформацію замовник хоче одержати в процесі експлуатації БД. Установити, як взаємодіють об’єкти предметної області, і як це відбивається на значеннях їхніх атрибутів.

Звернемось знову до прикладу. Уявимо, як виглядала б зведена таблиця, що зроблена, скажімо, на аркуші паперу, у якій розмістилися б всі значення перелічених нами атрибутів. Початок її був би приблизно такий, як показано на рисунку 9.

НомСт	ПІБ	НомКом	НомТел	Предм	Сем	Оцінка
001	Іванов І.І.	15	12-21	Фізика Хімія Ін. мова	2 2 1	3 4 3
002	Петров П.П.	15	12-21	Матем. Інформ. Хімія	2 1 1	5 4 4
003	Сидоров С.С.	10	22-23	Фізика	1	4

Рисунок 9

Ми знаємо, що реляційна БД — це є набір відношень. Але таблиця, що наведена на рисунку 9, не може вважатися поданням якого-небудь відношення, оскільки одна з вимог до таких таблиць, а саме, наявність тільки одного значення у кожному полі всякого запису, тут порушена. Щоб ця таблиця могла бути відображенням відношення, її треба перебудувати. Треба доповнити її новими записами, перерозподілити серед них значення полів, що вже є, та продублювати у порожні поля значення сусідів. Вигляд перетвореної таблиці повинен стати таким, який показаний на рисунку 10.

НомСт	ПІБ	НомКом	НомТел	Предм	Сем	Оцінка
001	Іванов І.І.	15	12-21	Фізика	2	3
001	Іванов І.І.	15	12-21	Хімія	2	4
001	Іванов І.І.	15	12-21	Ін. мова	1	3
002	Петров П.П.	15	12-21	Матем.	2	5
002	Петров П.П.	15	12-21	Інформ.	1	4
002	Петров П.П.	15	12-21	Хімія	1	4
003	Сидоров С.С.	10	22-23	Фізика	1	4

Рисунок 10

Ця таблиця вже може розглядатися як коректне зображення деякого відношення. Таке відношення називають *універсальним відношенням* проектованої БД. В універсальне відношення включаються всі атрибути, які цікавлять користувача (замовника) БД, тобто воно містить всі дані, які передбачається розміщати в БД. При проектуванні БД із застосуванням методу нормальних форм універсальне відношення використається як початкові дані для подальшого аналізу та перетворень.

Звісно, у реальності при проектуванні БД побудовою зведеної таблиці або заповненням конкретними даними універсального відношення ніхто не займається. Однак визначення всього набору атрибутів, а також з'ясування, до якого типу даних будуть належати їх значення, виконуються обов'язково.

Універсальне відношення як БД практично не використовується. Тільки невеликі БД вузького призначення можуть бути представлені однією таблицею (такі БД називають *плоскими таблицями*). Є відповідні програмні додатки, які дозволяють організувати ці БД і працювати з ними (наприклад, MS Works або MS Excel). Але, звичайно, в ІС їх ніколи не застосовують.

11.10. 2 Проблеми, що виникають при використанні єдиного відношення

Використання універсального відношення в БД тягне ряд проблем.

1 *Надмірність даних.* Значення стовпців таблиці універсального відношення багаторазово повторюються.

2 *Потенційна суперечливість*. Звернемося до прикладу. Якщо, скажімо, у слові "Фізика" — значення атрибута "Предмет", була допущена помилка, то для її виправлення треба знайти всі рядки, що містять відомості про цю дисципліну, і у всіх цих рядках зробити зміни. Більше того, при заповненні таблиці універсального відношення можуть бути використані різні форми запису того самого значення. Так у нашому прикладі можна записати: "Ін. мова", "Іноз. мова", "Іноземна мова" тощо.

З *Проблема обробки неповних даних*. Знову звернемося до нашого прикладу. Нехай є студент, що не встиг закінчити навчання з жодного предмета. Якщо вносити про нього відомості в таблицю, то треба використати значення NULL і, отже, задавати додаткові процедури для його обробки. Якщо ж відомості про такого студента не вносити в БД, то її користувачеві (тобто кураторові курсу) нічого про цього студента відомо не буде.

Ці проблеми розв’язуються шляхом розподілу даних у базі, і засобом для цього служить дослідження атрибутів на функціональну залежність.

11.10.3 Функціональна залежність

Функціональна залежність визначається так: нехай дано два атрибути *A* і *B*, тоді *B функціонально залежить від A*, якщо для кожного значення *A* існує рівно одне, пов'язане з ним, значення для *B*.

Атрибути *A* і *B* можуть бути складними (складовими), тобто бути не одиночними атрибутами, а групами атрибутів.

Математично функціональна залежність *B* від *A* позначається записом $A \rightarrow B$. Це означає, що у всіх кортежах з однаковими значеннями атрибута *A* атрибут *B* буде мати також однакові значення.

Функціональну залежність *B* від *A* можна ще зображувати графічно (рисунок 11).

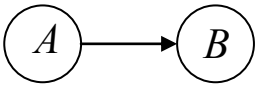


Рисунок 11

Уведемо ще кілька визначень.

Функціональна взаємозалежність. Якщо існують функціональні залежності $A \rightarrow B$ і $B \rightarrow A$, то між *A* і *B* є взаємно однозначна відповідність, або функціональна взаємозалежність. Наявність функціональної взаємозалежності між атрибутами *A* і *B* позначають як $A \leftrightarrow B$ або $B \leftrightarrow A$.

Ми вже говорили про нормальні форми відношень та про те, що при проектуванні БД застосовують процедуру нормалізації, за

допомогою якої відношення, що складають БД, переводяться з однієї нормальної форми в іншу більш високого порядку. Ми знаємо також, що найнижчим щаблем нормалізації відношення є 1НФ.

Визначення 1НФ. Відношення перебуває в 1НФ, якщо кожний його атрибут задається в кортежах атомарно (має у кожному кортежі тільки одне значення).

Висновок із цього визначення: будь-яке відношення (у тому числі, універсальне відношення) завжди перебуває принаймні в 1НФ.

Відношення в 1НФ має властивість: якщо відношення перебуває в 1НФ, то всі неключові його атрибути функціонально залежать від ключа з різним ступенем залежності.

Частковою залежністю (частковою *функціональною* залежністю) називається залежність неключового атрибута від частини складеного ключа.

Повною функціональною залежністю називається залежність неключового атрибута від усього складеного ключа.

Транзитивна залежність. Атрибут *C* залежить від атрибута *A* транзитивно (існує транзитивна залежність), якщо для атрибутів *A*, *B*, *C* виконуються умови $A \rightarrow B$ і $B \rightarrow C$, але зворотна залежність відсутня.

Повернемося до прикладу (проектування БД для куратора курсу). Розглянемо функціональні залежності, які мають атрибути нашого універсального відношення.

Основний спосіб визначення наявності функціональних залежностей — це ретельний аналіз семантики атрибутів. Для цього звичайно потрібно добре знати предметну область, розуміти її сутності й зв'язок між ними. У кожній конкретній ситуації функціональні залежності визначаються шляхом логічної деталізації властивостей атрибутів і їхньої взаємодії один з одним.

Після семантичного аналізу атрибутів нашого універсального відношення будемо мати такі функціональні залежності (ФЗ):

НомСт → ПІБ	НомКом → НомТел
НомСТ → НомКом	НомТел → НомКом
НомСт → НомТел	НомСт, Предм, Сем → Оцінка

Зобразимо всі ці залежності графічно (див. рисунок 12).

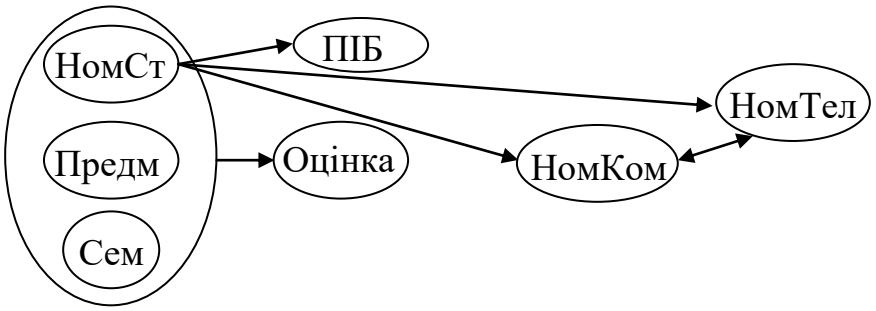


Рисунок 12

Наведені функціональні залежності були встановлені відповідно до таких міркувань.

1 ФЗ НомСт → ПІБ. Номер студента є унікальним атрибутом, оскільки відомо, що у списку одним номером можна визначити тільки одне прізвище. Зворотне твердження не є правильним (однакові прізвища та ініціали можуть бути у кількох людей у списку).

2 ФЗ НомСт → НомКом. Кожний студент, позначений своїм номером, проживає в якійсь одній кімнаті гуртожитку. Зворотної залежності немає, оскільки в одній кімнаті може жити кілька студентів.

3 ФЗ НомСт → НомТел. З кожним студентом можна зв’язатися за якимось одним номером телефону. Зворотне твердження не буде правильним, оскільки телефонні апарати стаціонарно знаходяться у кімнатах гуртожитку, й одним телефоном користуються всі студенти, що живуть у кімнаті.

4 ФЗ НомКом → НомТел і ФЗ НомТел → НомКом. За кожною кімнатою гуртожитку закріплений тільки один телефонний номер. І якщо подзвонити за якимось конкретним телефонним номером, то ми “потрапимо” тільки в одну конкретну кімнату гуртожитку. Тут ми маємо приклад функціональної взаємозалежності.

5 ФЗ НомСт, Предм, Сем → Оцінка. Оцінка може бути однозначно визначена тільки тоді, коли відомо, кому вона поставлена, за що й коли. Ця ФЗ являє собою приклад залежності простого атрибута від складеного атрибута.

11.10.4 Спосіб послідовної нормалізації реляційної БД

Розглянемо процес *послідовної нормалізації* реляційної БД *з поступовим підвищенням порядку нормальних форм відношень.*

Процес нормалізації складається з послідовності процедур декомпозиції відношень. Кожна декомпозиція ґрунтується на одній або декількох операціях проекції.

Визначимо *операцію проекції* так. Припустимо, що у відношенні *R(A, B, C, D, E, ...)*, де *A, B, C, D, E, ...* — атрибути відношення *R*, усунення функціональної залежності *C→D* дозволить перевести його в нову нормальну форму більш високого порядку. Тоді відношення *R* розкладається на два нових відношення *R1(A, B, C, E, ...)* і *R2(C, D)*. Відношення *R2* є *проекцією відношення R на атрибути C і D*. При цьому атрибути *C* і *D* можуть бути складеними (бути групами простих атрибутів).

Ми знаємо, що універсальне відношення перебуває в 1НФ. Необхідно перетворити його й одержати відношення, що перебувають у 2НФ.

Визначення 2НФ. Відношення перебуває в 2НФ, якщо воно перебуває в 1НФ, і кожний неключовий атрибут функціонально повно залежить від первинного ключа.

Звернемося до схеми функціональних залежностей універсального відношення із нашого прикладу (див. рисунок 12). Первинним ключем у цьому відношенні служить група атрибутів НомСт, Предм, Сем (тобто ключ є складеним). Неключові атрибути всі (як і має бути) залежать від ключа, але тільки атрибут “Оцінка” пов’язаний з ним повною функціональною залежністю. Із цього факту витікає: 1) наше універсальне відношення не перебуває у 2НФ; 2) щоб одержати з нього відношення, що перебуває у 2НФ, треба за допомогою операції проекції виключити ФЗ:

НомСт → ПІБ, НомКом, НомТел.

Виконаємо таку операцію: наше універсальне відношення *R(НомСт, Предм, Сем, ПІБ, НомКом, НомТел, Оцінка)* перетвориться у два відношення *R1(НомСт, Предм, Сем, Оцінка)* і *R2(НомСт, ПІБ, НомКом, НомТел)*. Графічне зображення

функціональних залежностей у відношеннях *R1* і *R2* показано на рисунку 13.

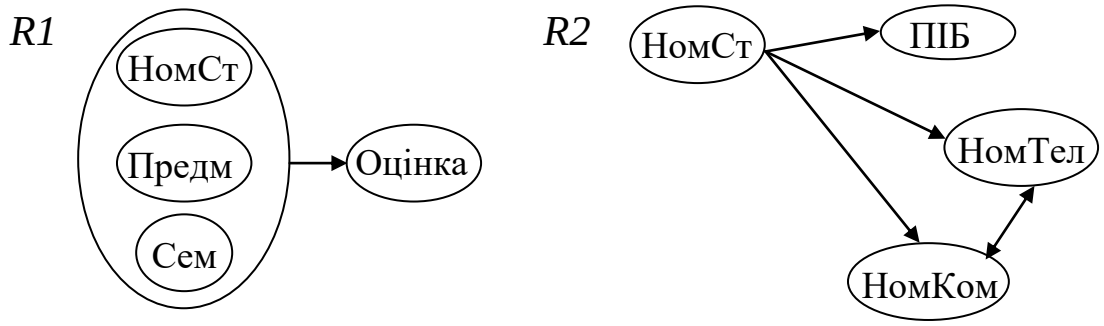


Рисунок 13

Визначення 3НФ. Відношення перебуває в 3НФ, якщо воно перебуває в 2НФ, і кожний неключовий атрибут нетранзитивно залежить від первинного ключа.

Існує й *альтернативне визначення 3НФ.* Відношення перебуває в 3НФ тоді і тільки тоді, коли всі неключові атрибути відношення взаємно незалежні й повністю залежать від первинного ключа.

Легко побачити, що у прикладі, який розглядаємо, *R1* вже перебуває в 3НФ. Але про відношення *R2* такого сказати не можна, оскільки, відповідно до першого визначення 3НФ, у ньому є атрибути, що мають транзитивну залежність:

НомСт → НомКом → НомТел і НомСт → НомТел → НомКом.

Крім цього, відповідно до другого визначення 3НФ, у відношенні *R2* є поміж неключовими атрибутами функціональні залежності: НомКом → НомТел і НомТел → НомКом.

Однак, якщо ми піддамо декомпозиції відношення R2 і за допомогою операції проекції виключимо взаємозалежність НомКом ↔ НомТел, то знову отримані відношення *R3(НомСт, ПІБ, НомКом)* і *R4(НомКом, НомТел)* будуть уже перебувати в 3НФ.

На практиці побудова 3НФ схем відношень у більшості випадків є достатньою, і процес проектування реляційної БД на них закінчується.

Якщо ж у відношеннях є залежності атрибутів складеного ключа від неключових атрибутів, то необхідно перейти до посиленої 3НФ.

Визначення посиленої 3НФ або нормальної форми Бойса-Кодда (БКНФ). Відношення перебуває в БКНФ, якщо воно перебуває в 3НФ, і в ньому відсутні залежності ключів (атрибутів складеного ключа) від неключових атрибутів.

У нашому прикладі подібних залежностей немає, тому проектування БД для куратора курсу закінчено. Його результат — набір відношень *R1, R3, R4*. Схема функціональних залежностей цих відношень показана на рисунку 14.

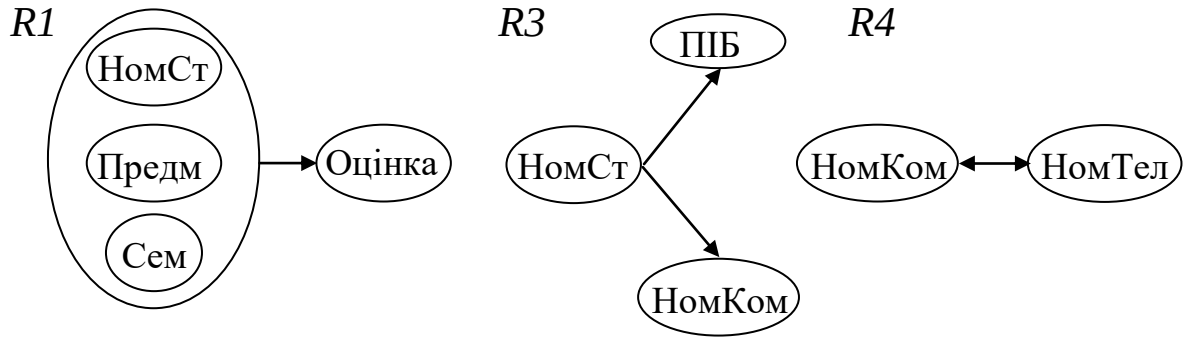


Рисунок 14

Розглянемо зв’язки між відношеннями у нормалізованій БД нашого прикладу. Очевидно, що зв’язок між двома відношеннями треба встановлювати за такими їхніми атрибутами, значення яких черпаються із одного домену. Такими атрибутами є звичайно ті, які знаходяться у лівих частинах записів функціональних залежностей, що стали основами для операцій проекції.

Вид виявлених зв’язків можна встановити або шляхом семантичного аналізу, або визначенням типу ключів, якими є атрибути, що утворили зв’язок.

Схематичне зображення зв’язків між відношеннями нашого прикладу наведено на рисунку 15.

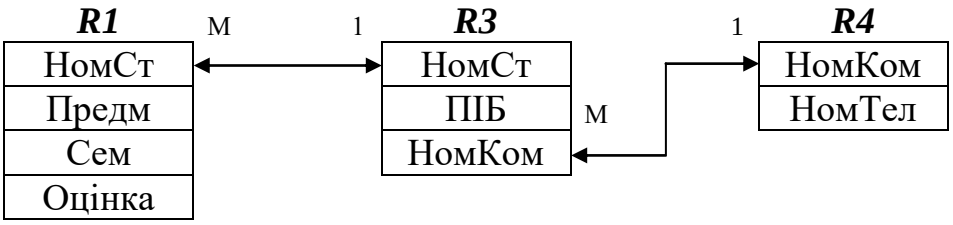


Рисунок 15

Нижче ознайомимося з ще одним підходом в організації процесу проектування БД із використанням методу нормальних форм.

11.10.5 Спосіб прямої нормалізації реляційної БД

Цей спосіб спирається на два поняття: можливий (потенційний) ключ і детермінант. Перше поняття вже було визначене раніше, треба визначити друге.

Детермінант. Якщо $A \rightarrow B$ є функціональна залежність і B не залежить функціонально від якої-небудь підмножини A, то кажуть, що A є детермінантом B.

Дамо ще одне формулювання БКНФ.

Альтернативне визначення БКНФ. Відношення перебуває в БКНФ тоді й тільки тоді, коли всі детермінанти у відношенні є можливими ключами.

Першим кроком процедури нормалізації БД тут, як і у попередньому способі, є визначення універсального відношення, а другим — встановлення функціональних залежностей між атрибутами цього відношення. Далі проводиться серія декомпозицій відношень. При цьому перевіряється кожне зі знову отриманих відношень, чи перебуває воно в БКНФ. Якщо — так, то таке відношення далі не перетворюється, інші — знову піддаються декомпозиції. Причому для кожної операції проекції опорна ФЗ вибирається така, щоб її виключення з відношення приводило б до утворення відношення, що вже перебуває у БКНФ.

Проілюструємо й цей спосіб нормалізації БД на прикладі, що розглядався раніше (проект БД для куратора курсу). Оскільки

початкові кроки першого й другого способів збігаються, звернемося відразу до набору ФЗ та їхнього графічного зображення на рисунку 12.

З рисунка ясно, що універсальне відношення *R* має тільки один можливий ключ: {НомСт, Предм, Сем}. Детермінантами цього відношення є ліві частини всіх функціональних залежностей: НомСт, НомКом, НомТел і {НомСт, Предм, Сем}. Із цього робимо висновок — універсальне відношення

R(НомСт, Предм, Сем, ПІБ, НомКом, НомТел, Оцінка)

не перебуває в БКНФ. Отже, його потрібно піддати декомпозиції.

Є декілька ФЗ для здійснення декомпозиції:

НомСт → ПІБ, Но мст →НомКом,
Но мст →НомТел, НомКом→НомТел,
НомТел→НомКом.

Але спочатку ми будемо використовувати ФЗ транзитивного типу, тобто ланцюжки виду *A→B→C*, при цьому для здійснення операції проекції треба вибирати праву ланку ланцюжка.

У нашому випадку таким ланцюжком є послідовність ФЗ: НомСт → НомКом → НомТел, а її правим кінцем є ФЗ: НомКом → НомТел, що і доцільно взяти основою для проекції.

У результаті цієї проекції універсальне відношення *R* розкладеться на відношення

R1(НомСт, Предм, Сем, ПІБ, НомКом, Оцінка)

і відношення *R2(НомКом, НомТел)*. Проаналізуємо обидва ці відношення. Відношення *R1* має один можливий ключ {НомСт, Предм, Сем} і два детермінанти: {НомСт, Предм, Сем} і НомСт. Висновок — *R1* не перебуває в БКНФ. Відношення *R2* має два можливих ключі (НомКом і НомТел) і два детермінанти (ті ж НомКом і НомТел). Отже, *R2* перебуває в БКНФ і не вимагає подальшої декомпозиції. Відношення *R1* необхідно ще розщеплювати. Очевидно, з нього варто видалити детермінант НомСт. Цей детермінант має відразу два залежних атрибути: НомСт → НомКом і НомСт → ПІБ, що можна розглядати як складну ФЗ НомСт → НомКом, ПІБ.

Операція проекції відношення *R1*, що породжується цією функціональною залежністю, дає два відношення

R3(НомСт, Предм, Сем, Оцінка) і *R4(НомСт, ПІБ, НомКом)*.

В *R3* можливий ключ один, і детермінант один, і це все є група {НомСт, Предм, Сем}. В *R4* ми спостерігаємо подібну картину: і можливим ключем, і детермінантом є атрибут НомСт.

Таким чином, остаточним проектом нашої БД є відношення:

R2(НомКом, НомТел), R3(НомСт, Предм, Сем, Оцінка), R4(НомСт, ПІБ, НомКом).

Тобто, ми дійшли до такого ж результату, що й у попередньому варіанті нормалізації.

11.11 Метод “сутність – зв’язок”

Проектування БД з великою кількістю атрибутів шляхом застосування декомпозицій та методу нормальних форм є дуже трудомістка і важка справа. Тому більшість фахівців воліють у таких випадках використовувати інші методи. Одним з таких методів називається “сутність – зв’язок”. Він відрізняється від методу декомпозиції тим, що ФЗ залучаються не на початковому, а на кінцевому етапі проектування.

11.11.1 Сутності та зв’язки

Припустимо, що проектується БД, призначена для збереження інформації про викладачів факультету і про ті курси, які вони читають. Двома головними об’єктами, або *сутностями*, є ВИКЛАДАЧ і КУРС. Ці дві сутності співвідносяться за допомогою *зв’язку* ЧИТАЄ, що дозволяє сказати:

ВИКЛАДАЧ ЧИТАЄ КУРС.

Зв’язок ЧИТАЄ, що існує між двома сутностями ВИКЛАДАЧ і КУРС, може бути графічно представлений двома способами. Рисунок 16 показує на прикладі використання *діаграми ER–екземплярів*. У цьому прикладі кожен викладач ідентифікується номером_викладача (НВ), кожен курс – назвою_курсу (НК). На рисунку 17 показана *діаграма ER– типу*, що містить ту ж інформацію, що і рисунок 16.

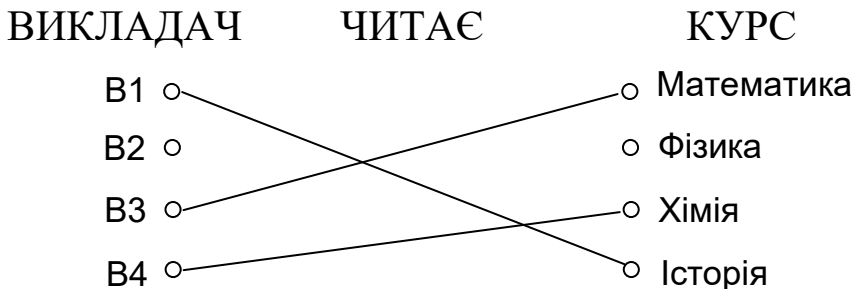


Рисунок 16

Рисунок 17

Розглянемо терміни, що будуть використовуватися далі.

СУТНІСТЬ. **Сутність** визначається як деякий об’єкт, що викликає інтерес. Цей об’єкт повинен мати *екземпляри*, що відрізняються один від одного і допускають однозначну ідентифікацію. Визначальна ознака, що може допомогти в знаходженні сутностей, полягає в тому, що сутність — це, як правило, іменник. Прикладами сутностей можуть бути студенти, предмети, машини тощо. На рисунках 16 і 17 сутностями є ВИКЛАДАЧ і КУРС, у той час як окремі екземпляри кожної сутності ідентифікуються за допомогою номера_викладача (В1, В2, В3, В4) і назви_курсу (Математика, Фізика, Хімія, Історія) відповідно.

ЗВ’ЯЗОК. **Зв’язок** являє собою з’єднання між двома або декількома сутностями. Зв’язок звичайно виражається дієсловом. Приклади зв’язків: студенти ВИВЧАЮТЬ предмети, робітники ОБСЛУГОВУЮТЬ машини тощо.

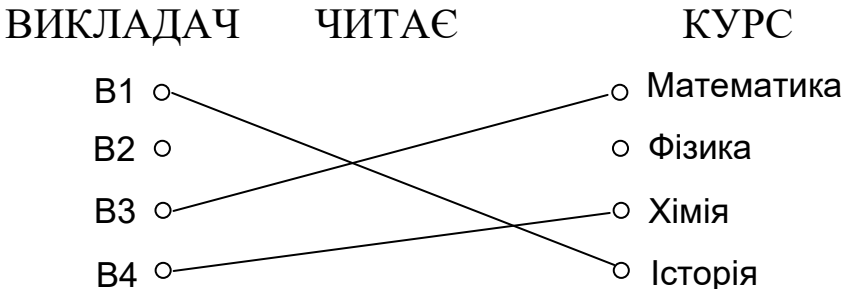
АТРИБУТ. **Атрибут** — це властивість сутності. Наприклад, атрибутами сутності СТУДЕНТ можуть бути: прізвище_студента, ім’я_студента, вік, факультет, курс тощо.

Атрибут або набір атрибутів, що використовується для ідентифікації екземпляра сутності, називається *ключем сутності*. Кожен екземпляр зв’язку однозначно визначається набором ключів сутностей, що з’єднуються цим зв’язком. Таким чином, <В3, Математика> є одним *ключем зв’язку*.

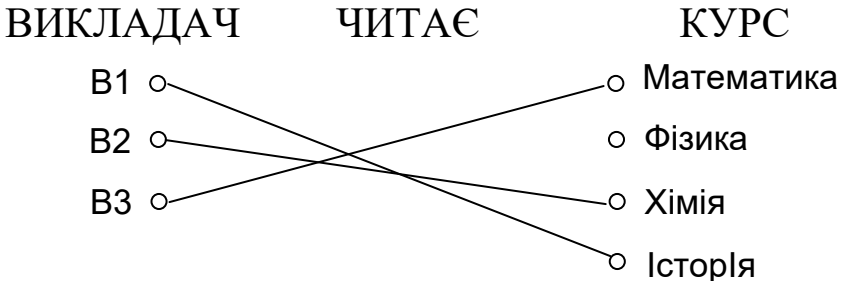
На рисунку 17 сутності показані у вигляді прямокутників, зв’язки — у вигляді ромбів. Під кожною сутністю розміщається атрибут або набір атрибутів, що є ключем сутності. Цифри “1” характеризують ступінь зв’язку, що буде розглянутий нижче.

11.11.2 Ступінь зв’язку

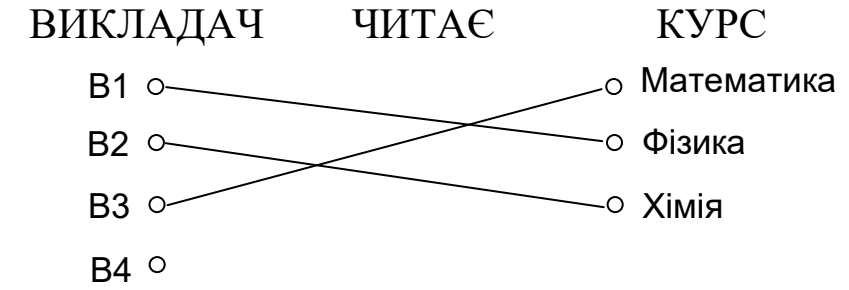
Важливою характеристикою зв’язку між двома і більше сутностями є *ступінь зв’язку*. Розглянемо це поняття на прикладі даних рисунка 16. Рисунок 18 показує всі можливі форми діаграми ER – екземплярів, що могли б існувати між сутностями ВИКЛАДАЧ і КУРС у тому випадку, коли ступінь зв’язку дорівнює 1:1. Кожен екземпляр сутності, що розташований як у лівій, так і у правій частинах діаграм, зв’язується тільки з одним екземпляром “протилежної” сутності. Це дає підставу визначити кожну з діаграм екземплярів, як діаграму, що має ступінь зв’язку 1:1. Кожна діаграма представляє набір можливих правил функціонування організації. Тільки одна з діаграм може бути істиною для організації в кожен момент часу.



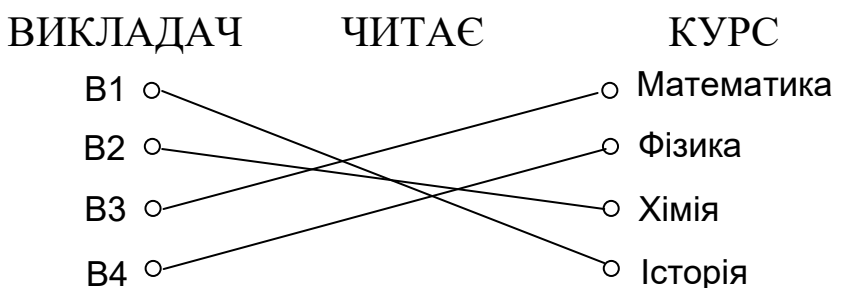
а) ступінь зв'язку дорівнює 1:1, і клас належності жодної із сутностей не є обов'язковим. Кожен викладач читає не більше одного курсу і кожен курс читається не більше ніж одним викладачем, тобто допускається наявність викладачів, що не читають жодного курсу, а також курсів, що не читаються.



б) ступінь зв'язку дорівнює 1:1, і клас належності сутності ВИКЛАДАЧ є обов'язковим. Кожен викладач читає тільки один курс, а кожен курс читається не більше ніж одним викладачем.



в) ступінь зв'язку дорівнює 1:1, і клас належності сутності КУРС є обов'язковим. Кожен викладач читає не більше одного курсу, а кожен курс читається тільки одним викладачем.



г) ступінь зв'язку дорівнює 1:1, і клас належності обох сутностей є обов'язковим. Кожен викладач читає тільки один курс і кожен курс читається тільки одним викладачем

Рисунок 18

На рисунку 19 наведено діаграми ER – типу, що відповідають діаграмам екземплярів, що наведено на рисунку 18.

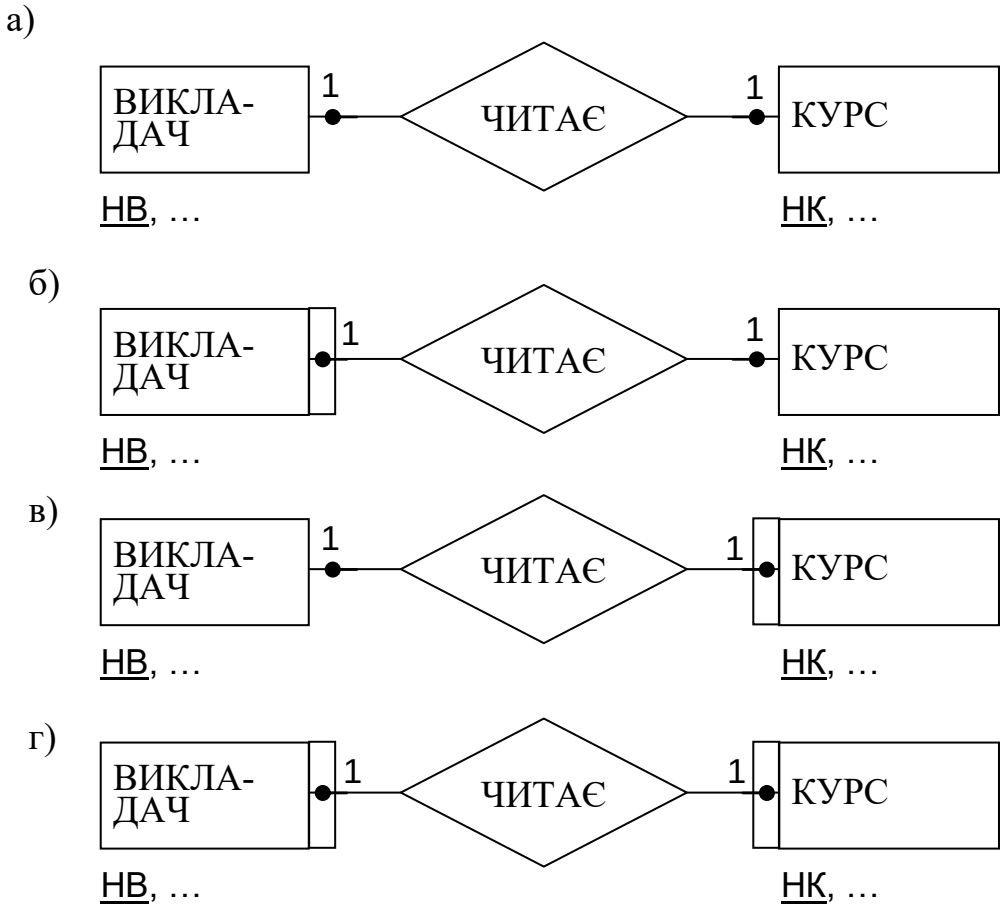


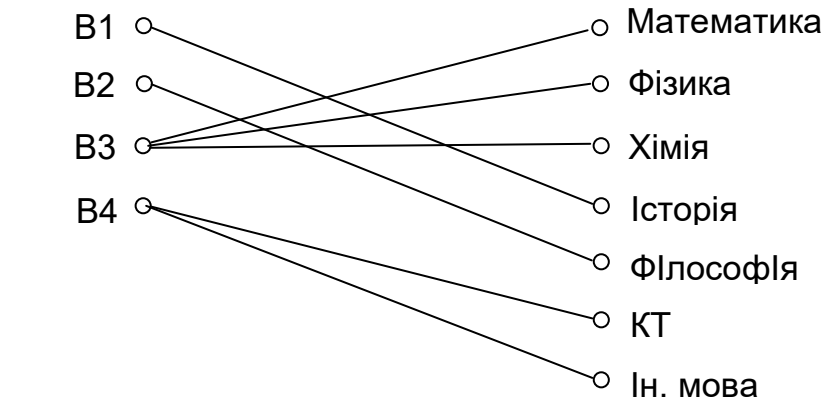
Рисунок 19

Якщо екземпляри даної сутності повинні брати участь у зв'язку, то участь називається *обов'язковою,* і цей факт відзначається розміщенням чорного кружка в блоці, що сполучений із блоком сутності. Якщо екземпляри даної сутності можуть не брати участь у зв'язку, то участь називається *необов'язковою* і кружок у блок не заноситься. *Клас належності* сутності повинен бути або обов'язковим, або необов'язковим і визначатися правилами діяльності організації.

Аналогічним способом будуються діаграми ER – екземплярів і діаграми ER – типу для випадків:

1 Кожен викладач може читати одночасно декілька курсів, але кожен курс читається не більше ніж одним викладачем. У цьому випадку ступінь зв’язку 1 : m. На рисунку 20 наведено один варіант класу належності для ступеня зв’язку 1 : m.

а)



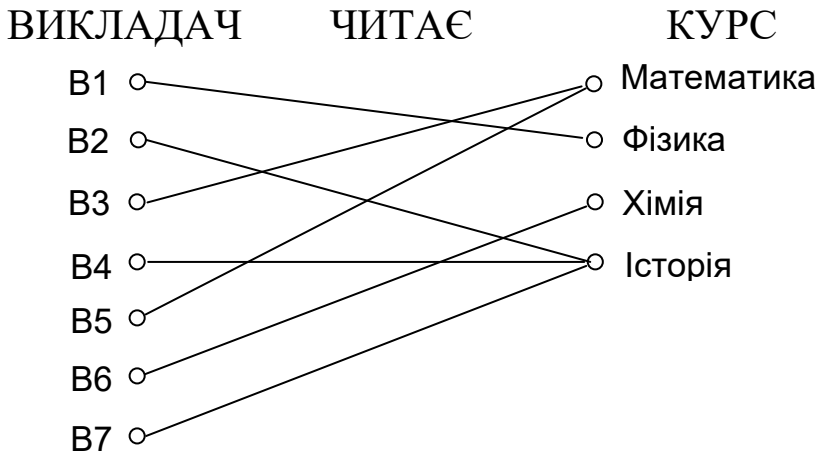
б)

Рисунок 20

2 Кожен викладач читає не більше одного курсу, але кожен курс може читатися відразу декількома викладачами. У цьому випадку ступінь зв’язку m:1. На рисунку 21 наведено один варіант класу належності для ступеня зв’язку m : 1.

3 Кожен викладач може читати кілька курсів, і кожен курс може читатися декількома викладачами. У цьому випадку ступінь зв’язку m : m. На рисунку 22 наведено один варіант класу належності для ступеня зв’язку m : m.

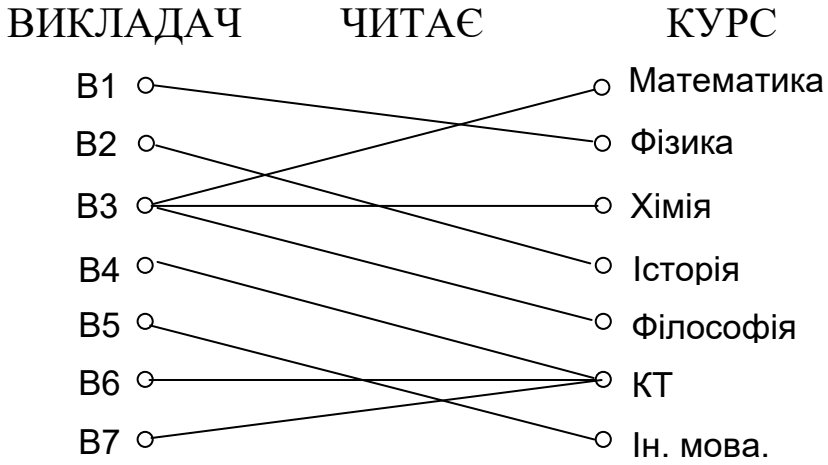
а)



б)

Рисунок 21

а)



б)

Рисунок 22

11.11.3 Одержання відношень з діаграм ER- типу

Зв'язок називається *бінарним*, якщо він зв'язує тільки дві сутності. Бінарні зв'язки зустрічаються найбільш часто.

Загальний підхід до побудови БД із використанням ER-методу полягає, насамперед, у побудові діаграми ER-типу, що включає в себе всі сутності і зв'язки. Другий крок у процесі проектування – побудова набору попередніх відношень і вказівка передбачуваного первинного ключа для кожного відношення. Останній крок полягає в підготовці списку всіх атрибутів, що мають інтерес, і призначенні кожного з цих атрибутів одному з попередніх відношень, з тією умовою, що ці відношення знаходилися в БКНФ. На цьому останньому кроці для кожного відношення повинні бути визначені ФЗ між атрибутами, за допомогою яких перевіряється відповідність відношень БКНФ.

Перелік загальних правил генерації відношень з діаграм ER-типу можна одержати, спираючись на клас належності та ступінь відношення. При розгляданні правил будемо використовувати приклад ВИКЛАДАЧ ЧИТАЄ КУРС і обмежимось випадками, в яких ступінь бінарного зв'язку дорівнює 1:1.

У випадку, коли ступінь зв'язку 1:1 і клас належності є обов'язковим і для сутності ВИКЛАДАЧ, і для сутності КУРС, гарантується однократна поява кожного значення НВ і кожного значення НК у будь-якому екземплярі відношення (рисунок 23). Це означає, що відношення ніколи не буде містити ані порожньої інформації, ані повторюваних груп надлишкових даних. Ключ сутності ВИКЛАДАЧ може служити первинним ключем для відношення, але може використовуватися і ключ сутності КУРС. У відношенні на рисунку 20 сутність ВИКЛАДАЧ доповнена двома атрибутами: прізвище викладача (ПрВикл) і телефон викладача (ТелВикл); сутність КУРС доповнена одним атрибутом: семестр курсу (Семестр).

Таким чином, можна сформулювати перше правило генерації відношень.

ПРАВИЛО 1. Якщо ступінь бінарного зв’язку дорівнює 1:1 і клас належності обох сутностей є обов’язковим, то потрібно тільки одне відношення. Первинним ключем цього відношення може бути ключ кожної з двох сутностей.

ВИКЛАДАЧ

НВ	ПрВикл	ТелВикл	НК	Семестр
В1	Федоров	20-01-02	Математика	1
В2	Козлов	99-55-44	Фізика	1
В3	Іваненко	22-33-22	Хімія	2
В4	Рябченко	12-34-56	Філософія	1

Рисунок 23

Якщо ступінь зв’язку дорівнює 1:1 і клас належності однієї сутності є обов’язковим, а іншої — необов’язковим, то одного відношення недостатньо. На рисунку 22 наведено екземпляр відношення у випадку, коли клас належності сутності ВИКЛАДАЧ є обов’язковим, а сутності КУРС — необов’язковим. У цьому випадку пропуски виникають в усіх кортежах, що містять інформацію про курси, що не читаються жодним з викладачів.

ВИКЛАДАЧ

НВ	ПрВикл	ТелВикл	НК	Семестр
В1	Федоров	20-01-02	Математика	1
В2	Козлов	99-55-44	Фізика	1
В3	Іваненко	22-33-22	Хімія	2
---	----------	----------	Філософія	1

Рисунок 24

ПРАВИЛО 2. Якщо ступінь бінарного зв’язку дорівнює 1:1 і клас належності однієї сутності є обов’язковим, а іншої — необов’язковим, то необхідна побудова двох відношень. Під кожну сутність необхідно виділити одне відношення, при цьому ключ сутності повинен служити первинним ключем для відповідного відношення. Крім того, ключ сутності, для якої клас належності є необов’язковим, додається як атрибут у відношення, що виділене для сутності з обов’язковим класом належності (рисунок 25).

Скориставшись правилом 2 у ситуації, де клас належності сутності КУРС є обов'язковим, а сутності ВИКЛАДАЧ — необов'язковим, одержимо такі відношення:

ВИКЛАДАЧ (НВ, ПрВикл, ТелВикл),
КУРС (НК, Семестр, НП).

У випадку, коли ступінь бінарного зв'язку дорівнює 1:1 і клас належності жодної із сутностей не є обов'язковим, одного відношення недостатньо. При використанні тільки одного відношення можливі два шляхи виникнення пропусків (рисунок 26). Також недостатнім є використання двох відношень, тому що виникають проблеми у зв'язку з внесенням ключа однієї сутності у відношення, виділене під іншу сутність (рисунок 27). Єдине рішення полягає у виділенні трьох відношень: по одному для кожної сутності й одного для зв'язку (рисунок 28).

ВИКЛАДАЧ

НВ	ПрВикл	ТелВикл	НК
В1	Федоров	20-01-02	Математика
В2	Козлов	99-55-44	Фізика
В3	Іваненко	22-33-22	Хімія

КУРС

НК	Семестр
Математика	1
Фізика	1
Хімія	2
Філософія	1

Рисунок 25

ВИКЛАДАЧ

НВ	ПрВикл	ТелВикл	НК	Семестр
В1	Федоров	20-01-02	Математика	1
В2	Козлов	99-55-44	----------	---
В3	Іваненко	22-33-22	Фізика	1
В4	Петров	33-44-55	Хімія	2
---	----------	----------	Філософія	1

Рисунок 26

ВИКЛАДАЧ

НВ	ПрВикл	ТелВикл	НК
В1	Федоров	20-01-02	Математика
В2	Козлов	99-55-44	----------
В3	Іваненко	22-33-22	Фізика
В4	Петров	33-44-55	Хімія

КУРС

НК	Семестр	НВ
Математика	1	В1
Фізика	1	В2
Хімія	2	В4
Філософія	1	---

Рисунок 27

ВИКЛАДАЧ

НВ	ПрВикл	ТелВикл
В1	Федоров	20-01-02
В2	Козлов	99-55-44
В3	Іваненко	22-33-22
В4	Петров	33-44-55

КУРС

НК	Семестр
Математика	1
Фізика	1
Хімія	2
Філософія	1

ЧИТАЄ

НВ	НК
В1	Математика
В2	Фізика
В4	Хімія

Рисунок 28

ПРАВИЛО 3. Якщо ступінь бінарного зв’язку дорівнює 1:1 і клас належності жодної сутності не є обов'язковим, то необхідне використання трьох відношень: по одному для кожної сутності, ключі яких служать у якості первинних у відповідних відношеннях, і одного для зв’язку. Серед своїх атрибутів відношення, що виділене для зв’язку, буде містити по одному ключу сутності від кожної сутності.

Для інших ступенів бінарного зв’язку теж існують свої правила переходу від ER-діаграм до таблиць відношень. Детально про це можна дізнатися, наприклад, із літератури, список якої наведений у кінці цього конспекту. Послідовне застосування методики завдання ER-діаграм, що була розглянута вище, та правил побудови за цими діаграмами відповідних таблиць дає можливість здійснювати проектування реляційних БД будь-якої складності.

12 ДИСПЕТЧЕРСЬКА ІНФОРМАЦІЙНА СИСТЕМА ДІЛЯНКИ КИЇВ-ФАСТІВ ПІВДЕННО-ЗАХІДНОЇ ЗАЛІЗНИЦІ

Добрим прикладом інформаційної системи, що розроблена для задоволення потреб залізничного руху, є диспетчерська інформаційна система ділянки Київ-Фастів Південно-Західної залізниці. Ця система призначена виконувати широкий спектр функцій у таких напрямках: ведення поїзної моделі (графіка руху поїздів), ведення на всій ділянці вагонної моделі, організаційно-розпорядницькі функції, контроль стану технічних пристроїв, облік і аналіз різних робіт, що проводять на ділянці.

Під поїзною моделлю мається на увазі виконання задач: одержання інформації про кожний поїзд, що перебуває на ділянці або надходить із сусідньої ділянки; виявлення й реєстрація поїздів, що їдуть із відхиленнями від графіка й поза графіком; виділення поїздів, що вимагають особливих умов пропуску по ділянці; виділення поїздів окремих категорій на полігоні ділянки й графіку руху поїздів; ведення додатків до графіка руху; планування пропуску поїздів, що їдуть за графіком, і поїздів, що спізнюються; уведення й обробка інформації про поїзди, що мають підійти, кинутих, знову утворених, і поїзди, яким відмовлено в прийманні; планування й узгодження поїздоутворення збірних поїздів; планування забезпечення поїздів локомотивами й локомотивними бригадами; ведення й коректування графіка руху поїздів.

Ведення вагонної моделі містить у собі вирішення таких завдань: балансовий загальний облік місцевих вагонів і перевезених вантажів на всій ділянці й по станціях; формування вагонопотоків, що ввозяться на ділянку і що вивозяться з ділянки, по напрямках руху та у цілому; надавати інформацію про стан і хід виконання місцевої роботи на ділянці в цілому й окремо по проміжних станціях.

По організаційно-розпорядницьких функціях ця система забезпечує: реєстрацію й контроль приймання та здачі чергувань; контроль складу змін оперативно взаємодіючих працівників на всіх робочих ділянках; одержання, реєстрацію, відображення й аналіз інформації про взаємодію причетних працівників — накази, телефонограми, службові повідомлення тощо.

По контролю стану технічних пристроїв системою забезпечується: одержання інформації про технічні пристрої на станціях і перегонах, про їхній стан та оцінки можливості використання для безпечного руху поїздів; уведення інформації про стан технічних пристроїв; уведення інформації про перехід з одних засобів зв’язку за рухом поїздів на інші; уведення інформації про попередження обмеження швидкості; планування “вікон” для проведення ремонтних робіт на ділянці.

Диспетчерська інформаційна система забезпечує облік і аналіз роботи на ділянці шляхом реєстрації ведення поїзної та місцевої роботи з наступним архівуванням інформації та її

статистичною обробкою для вироблення рекомендацій щодо удосконалювання управління.

Оперативні дані в інформаційну систему надходять від двох типів джерел: через ручне уведення із клавіатури при заповненні різних електронних (комп'ютерних) журналів та форм відповідальними особами й через мережу автоматичних датчиків. Програмне забезпечення диспетчерської інформаційної системи складено алгоритмічною мовою високого рівня C++ і працює під управлінням операційної системи Linux. Комплекти апаратних засобів інформаційної системи базуються на персональних ЕОМ промислового виконання на основі процесорів Intel Pentium.

У базі даних диспетчерської інформаційної системи ділянки Київ-Фастів Південно-Західної залізниці застосовується реляційна модель організації даних. Це є розподілена БД, тобто її дані фізично не зберігаються в одному місці (на якомусь одному сервері), а розміщені в декількох вузлах комп'ютерної мережі системи.

До БД такої складної інформаційної системи надходять дані за багатьма десятками атрибутів. Тому їхні описи й вказівки функціональних залежностей між ними тут не можуть бути наведені повністю. Нижче розглядається проект тільки деякої її частини. Причому, щоб цей приклад був логічно закінченим, уводяться додаткові спрощення.

Змістовно база даних інформаційної системи, що розглядаємо, розкладається на кілька частин: відомості про стан колій і технічних засобів управління рухом; довідкові дані; розклад руху поїздів; відомості про перевезені вантажі; відомості про рух і состав поїздів; відомості про стан рухомого складу. База даних, структура якої наводиться нижче, являє собою спрощений варіант двох останніх названих частин реальної бази системи. Зв'язки з іншими частинами тут подаються через наявність окремих відповідних атрибутів. З розгляду повністю виключені пасажирські перевезення, а облік проведених робіт з рухомим складом показаний лише частково. Аналіз структури цієї бази зроблений щодо нормалізації за допомогою методу нормальних форм.

Назви атрибутів

Нижче наводяться назви атрибутів, які входять до бази даних, та їхні скорочення, що використовуються в описах функціональних залежностей. Назва кожного атрибута супроводжується коротким поясненням або характерним прикладом його значення.

Назва станції, де знаходиться парк — СтаПрк (Київ, Фастів і т.д.).

Номер поїзда — НомПзд (102 і т.д.).

Номер локомотива — НомЛок (000251 і т.д.).

Машиніст локомотива поїзда — МашПзд (прізвище, ім’я, по батькові або табельний номер).

Назва станції формування поїзда — СтаФор (Київ, Фастів і т.д.).

Назва станції призначення поїзда — СтаНаз (Київ, Фастів і т.д.).

Порядок відносної нумерації вагонів у поїзді — НумВаг (з голови, із хвоста).

Умовна довжина поїзда в момент формування — УДлПзд (у вагонах).

Номер вагона — НомВаг (багатоцифровий заводський номер).

Номер вагона один за одним у поїзді — НоВПзд (6, 10 і т.д.).

Вага вантажу вагона — ВесГрз (у тоннах).

Назва станції призначення вагона — СтаНВа (Київ, Фастів і т.д.).

Одержувач вантажу — ПолГрз (код одержувача або ім’я одержувача).

Код вантажу — КодГрз (ідентифікаційний код вантажу).

Тип вагона — ТипВаг (вантажний відкритий, пасажирський і т.д.).

Власник вагона — СобВаг (код власника або ім’я власника).

Момент приєднання вагона до поїзда — МоПриВ (12.09.07-19:00 і т.д.).

Момент зміни стану вагона у парку — МСмСоВ (12.09.07-19:00 і т.д.).

Номер парку — НомПрк (ідентифікаційний номер парку на станції).

Номер колії у парку — НомПут (4, 12, 15 і т.д.).

Завантаженість вагона — ЗагВаг (завантажений, порожній).

Чи закріплений вагон на коліях — ЗакВаг (закріплений, незакріплений).

До якого типу парку належить вагон — ТипПрк (робочий, неробочий).

Момент формування поїзда — МоФПзд (мається на увазі дата й час відправлення поїзда зі станції формування).

Назва станції прибуття поїзда — СтаПри (Київ, Фастів і т.д.).

Назва станції від'їзду поїзда — СтаОтб (Київ, Фастів і т.д.).

Момент прибуття поїзда — МоПриб (12.09.07-19:00 і т.д.).

Момент від'їзду поїзда — МомОтб (12.09.07-19:00 і т.д.).

Номер основної колії, займаний поїздом — НмОсПт (4, 6, 10 і т.д.).

Умовна довжина поїзда при від'їзді зі станції — ДлПзОт (у вагонах).

Момент відходу вагона із зони спостереження системи — МоУход (12.09.07-19:00 і т.д.).

Напрямок відходу вагона — НаУход (назва станції прямування поїзда).

Код станції — КодСта (ідентифікаційний код станції в таблиці бази даних).

Функціональні залежності

Наведемо функціональні залежності, які враховувалися при проектуванні бази даних:

НомВаг, МсмСоВ → ТипПрк
НомВаг, МсмСоВ → СтаПрк
НомВаг, МсмСоВ → НомПут
НомВаг, МсмСоВ → НомПрк
НомВаг, МсмСоВ → ЗагВаг
НомВаг, МсмСоВ → ЗакВаг
НомВаг → СобВаг
НомВаг → ТипВаг
НомВаг → інші постійні характеристики вагона
НомПзд, МоФПзд → СтаФор
НомПзд, МоФПзд → НумВаг
НомПзд, МоФПзд → СтаНаз

НомПзд, МоФПзд → НомЛок
НомЛок → інші постійні характеристики локомотива
НомПзд, МоФПзд → МашПзд
МашПзд → інші дані про працівника залізниці
НомПзд, МоФПзд → ДлПзОт
НомПзд, МоФПзд, СтаПри → НмОсПт
НомПзд, МоФПзд, СтаПри → МомОтб
НомПзд, МоФПзд, СтаПри → МоПриб
НомПзд, МоФПзд, СтаПри → СтаОтб
НомПзд, МоФПзд, СтаПри → ДлПзОт
НомПзд, МоФПзд, НомВаг → СтаНВа
НомПзд, МоФПзд, НомВаг → МоПриВ
НомПзд, МоФПзд, НомВаг → ВесГрз
НомПзд, МоФПзд, НомВаг → НоВПзд
НомПзд, МоФПзд, НомВаг → КодГрз
НомПзд, МоФПзд, НомВаг → ПолГрз
НомВаг, МоУход → НаУход
СтаОтб → КодСта
СтаПрк → КодСта
СтаФор → КодСта
СтаНаз → КодСта
СтаНВа → КодСта
СтаПри → Код ста

Схеми функціональних залежностей і зв'язків між таблицями бази

На рисунку 29 показана схема функціональних залежностей, список яких був поданий вище. Схема показує, що в базі доцільно утворити таблиці, що дозволяють відокремити атрибути шести об'єктів: вагонів, локомотивів, поїздів, руху поїздів від станції до станції, станцій і машиністів поїздів (працівників залізниці). Причому стани вагонів, що перебувають під контролем інформаційної системи, варто описувати декількома таблицями (див. рисунок 30). Це необхідно, оскільки кожний зі станів визначається різними атрибутами. Об'єднання їх в одній таблиці неминуче призвело б до появи порожніх полів у записах що не можна вважати раціональним рішенням побудови БД.

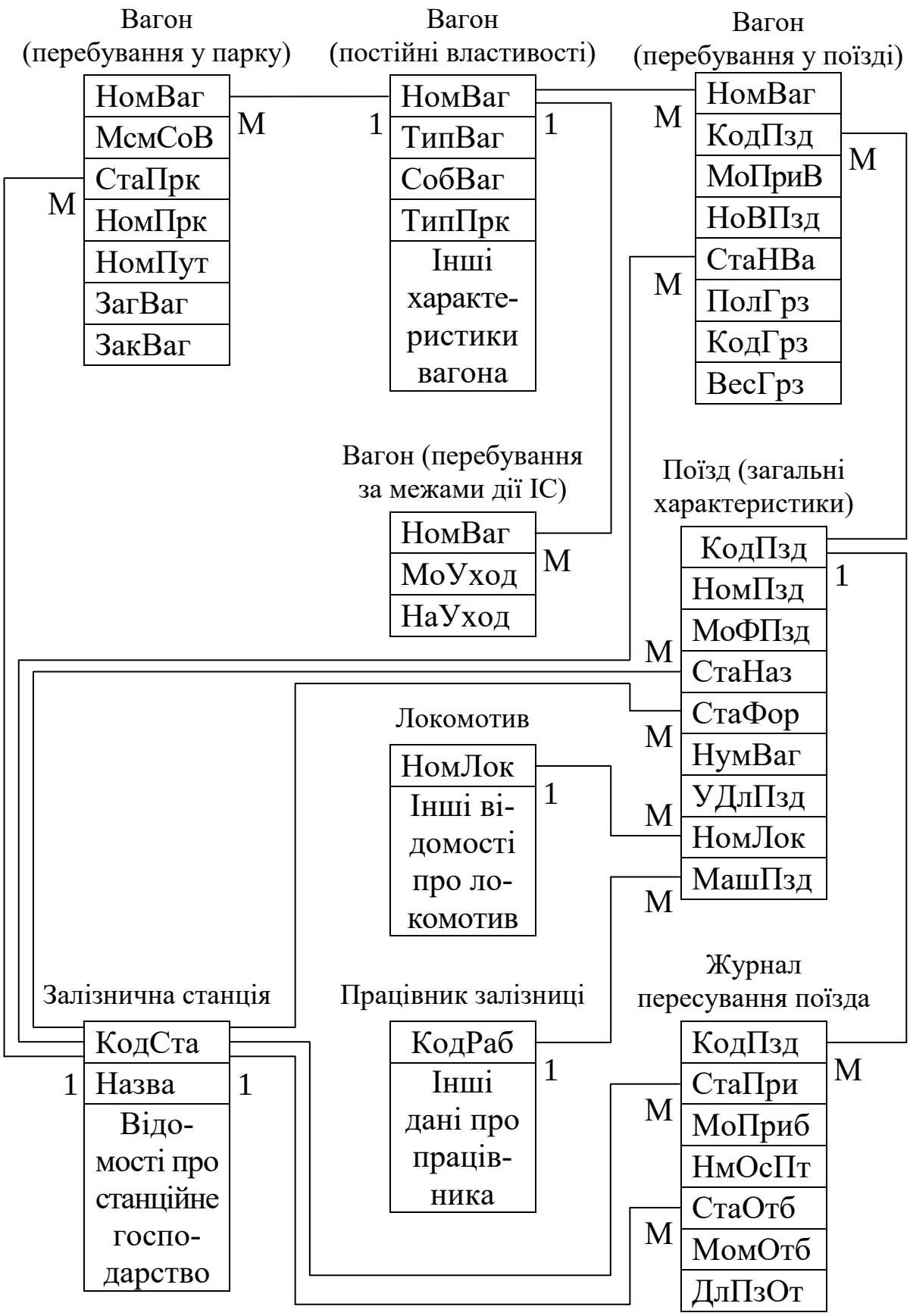

Вагон (перебування у парку)
НомВаг
МсмСоВ
СтаПрк
НомПрк
НомПут
ЗагВаг
ЗакВаг
Вагон (постійні властивості)
НомВаг
ТипВаг
СобВаг
ТипПрк
Інші характе-ристики вагона
Вагон (перебування у поїзді)
НомВаг
КодПзд
МоПриВ
НоВПзд
СтаНВа
ПолГрз
КодГрз
ВесГрз
Вагон (перебування за межами дії ІС)
НомВаг
МоУход
НаУход
Поїзд (загальні характеристики)
КодПзд
НомПзд
МоФПзд
СтаНаз
СтаФор
НумВаг
УДлПзд
НомЛок
МашПзд
Локомотив
НомЛок
Інші ві-домості про ло-комотив
Залізнична станція
КодСта
Назва
Відо-мості про станційне госпо-дарство
Працівник залізниці
КодРаб
Інші дані про праців-ника
Журнал пересування поїзда
КодПзд
СтаПри
МоПриб
НмОсПт
СтаОтб
МомОтб
ДлПзОт
1
М

Рисунок 30

Система дозволяє фіксувати будь-яку зміну у станах кожного контрольованого нею об’єкта. Події будь-якого відрізка часу можуть бути нею відтворені в хронологічному порядку на екрані монітора перед користувачем.

Обробка таблиць розглянутого тут фрагмента бази даних диспетчерської інформаційної системи пов’язана із частими порівняннями значень атрибутів, що обумовлюють моменти зміни станів об’єктів предметної області. Тому для управління даними в системі широко використовується апарат індексів. Наприклад, щоб виконати запит про місцезнаходження на ділянці вагона із заданим номером, необхідно виконати перебирання записів із трьох таблиць із метою визначення такої, де значення моменту зміни стану шуканого вагона буде найближчим до поточного моменту.

Однак разом із наведеними таблицями система автоматично може вести індексну таблицю, кожний запис якої складається зі значень усього чотирьох атрибутів: номера вагона, моменту зміни стану вагона, ідентифікатора нового стану вагона й коду (номера) запису опису цього стану вагона у відповідній таблиці. Тоді виконання запиту пошуку зводиться до перебирання системою записів індексу й звернення безпосередньо до потрібних даних у якійсь одній із трьох згаданих таблиць, що значно скорочує кількість операцій, вироблених системою.

Для розглянутої частини бази даних ІС вхідні дані надходять із клієнтських автоматизованих робочих місць (АРМ) поїзного диспетчера та чергових по станціях. Оскільки система повинна зберігати у своїй базі повну картину динаміки подій на контрольованій нею ділянці, то зміни значень полів у вже існуючих записах таблиць бази майже не виконуються. Нові дані, що поставляються із клієнтських АРМ, постійно збільшують потужність таблиць. Коли кількість записів у таблицях доходить до деякого порога, проводиться архівація й видалення з таблиць тих записів, у яких значення моментів зміни стану об’єкта найбільше віддалені за часом від сучасного моменту.

Зі схеми функціональних залежностей (рисунок 29) видно, що частина таблиць у базі мають складені ключі. Користуватися такими ключами не зручно, тому в практиці їх замінюють простими штучними ключами. Такі заміни були зроблені й у

структурі таблиць бази даних диспетчерської інформаційної системи. У списку атрибутів і на схемі функціональних залежностей зазначені тільки ті штучні ключі, без яких схема втрачає важливі залежності. На схемі зв’язків між таблицями бази (рисунок 30) вони усі присутні.

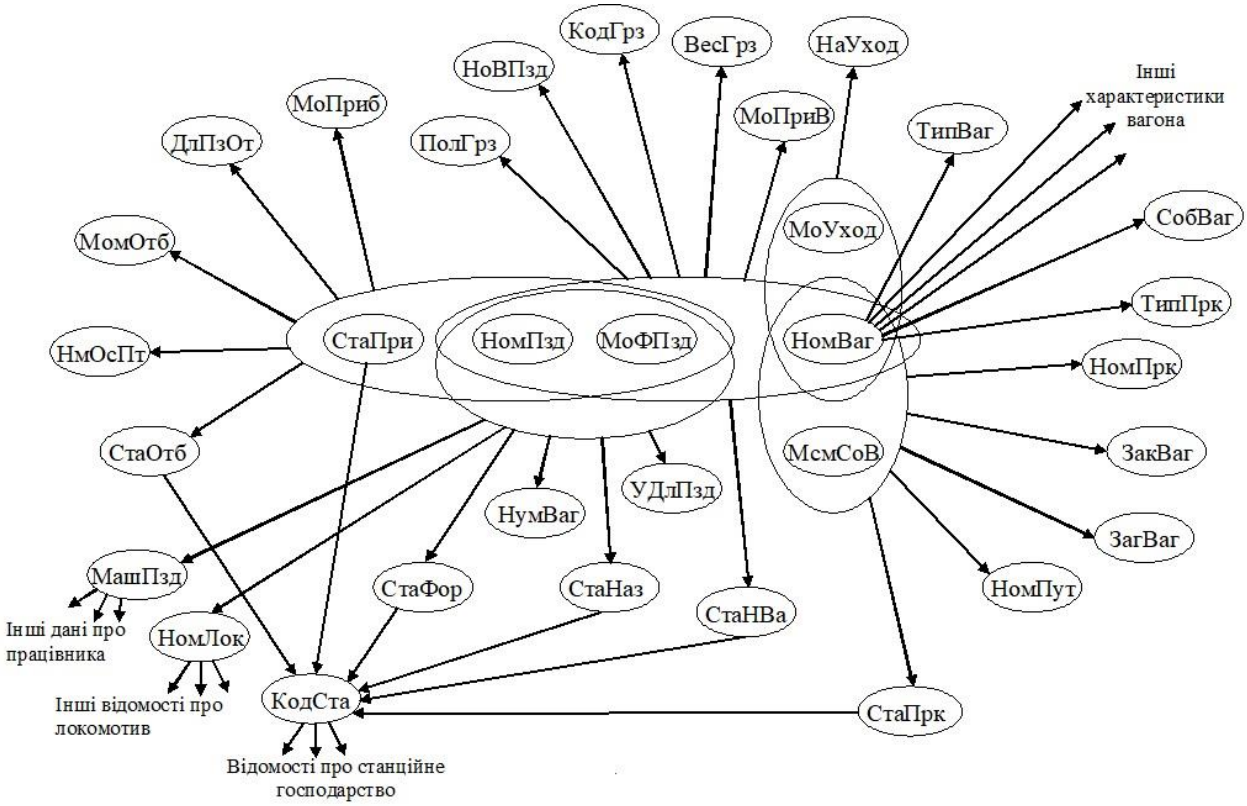


Рисунок 29

СПИСОК ЛІТЕРАТУРИ

1 Избачков Ю.С., Петров В.Н. Информационные системы: Учебник для вузов. – 2-е изд. – С.Пб.: Питер, 2005.
2 Хоменко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для вузов. – 4-е изд. доп. и переработ. – С.Пб.: Корона принт, 2004.
3 Голицына О.Л., Максимов Н.В., Попов И.И. Базы данных: Учебное пособие. – М.: Форум, 2004.
4 Кузнецов С.Д. Основы современных баз данных. – www.citforum.ru, 2002.
5 Дейт К. Дж. Введение в системы баз данных. – 6-е изд. – К., М., С.Пб.: Вильямс, 2000.
6 Хэлворсон М., Янг М. Эффективная работа с Microsoft Office 2000. – С.Пб.: Питер, 2001.

С.Є. Бантюков, В.М. Бутенко, В.Г. Пчолін

КОНСПЕКТ ЛЕКЦІЙ

з дисципліни

“ІНФОРМАЦІЙНІ СИСТЕМИ НА ЗАЛІЗНИЧНОМУ ТРАНСПОРТІ”

Частина 1

Відповідальний за випуск Бантюков С.Є.

Редактор Буранова Н.В.

Підписано до друку 16.04.07 р.

Формат паперу 60х84 1/16 . Папір писальний.

Умовн.-друк.арк. 3,5. Обл.-вид.арк. 3,75.

Замовлення № Тираж 100. Ціна

Видавництво УкрДАЗТу, свідоцтво ДК № 2874 від. 12.06.2007 р.
Друкарня УкрДАЗТу,
61050, Харків - 50, пл. Фейєрбаха, 7