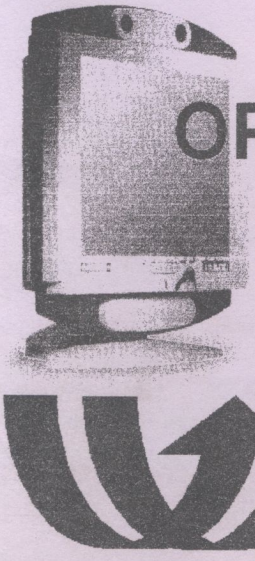
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВІННИЦЬКИЙ ДЕРЖАВНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

О.Н.Романюк, Т.О.Савчук



ОРГАНІЗАЦІЯ БАЗ ДАНИХ І ЗНАНЬ

Вінниця ВДТУ 2001

Міністерство освіти і науки України
Вінницький державний технічний університет

Олександр РОМАНЮК, Тамара САВЧУК

ОРГАНІЗАЦІЯ БАЗ ДАНИХ І ЗНАНЬ

ЧАСТИНА І

Затверджено Ученою радою Вінницького державного технічного університету як навчальний посібник для студентів спеціальності"Інтелектуальні системи прийняття рішень" денної та заочної форм навчання. Протокол № 4 від 30 листопада 2000р.

Вінниця ВДТУ 2001

УДК 681.31
Р 69

Рецензенти:
В.О.Поджаренко, доктор технічних наук, професор
В.Є.Качуровський, кандидат технічних наук, доцент
В.П.Кожем'яко, доктор технічних наук, професор

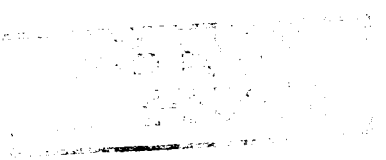
Рекомендовано до видання Ученою радою Вінницького державного технічного університету Міністерства освіти і науки України

О.Н.Романюк, Т.О.Савчук
Р 69 Організація баз даних і знань. Частина 1. Навчальний посібник. - Вінниця: ВДТУ, 2001. - 123 с.

В навчальному посібнику розглянуто основи організації та керування базами даних та знань, викладені основні положення, наведені приклади і рекомендації по їх засвоєнню, приведені рекомендації по розширеному опануванню. Велика увага приділена сучасним засобам керування базами даних. Приведені контрольні запитання, задачі для самостійної роботи, що можуть бути використані при проведенні як практичних, так і лабораторних занять.

Навчальний посібник призначено для студентів спеціальностей "Програмне забезпечення автоматизованих систем" та "Інтелектуальні системи прийняття рішень" денної та заочної форм навчання.

УДК 681.31



ЗМІСТ

ВСТУП

Основні ідеї сучасної інформаційної технології базуються на концепції баз даних (БД). Відповідно до цієї концепції, основою інформаційної технології є дані, які організовані в БД із метою адекватного відображення реального світу і задоволення інформаційних потреб користувачів.

Збільшення об'єму і структурної складності збережених даних, розширення кола користувачів інформаційних систем висунуло вимогу створення зручних загальносистемних засобів інтеграції збережених даних і керування ними. Це привело до появи промислових систем керування базами даних (СКБД) - спеціалізованих програмних засобів, призначених для організації і ведення БД. Систему, яка забезпечує створення, ведення і застосування баз знань, можна розглядати як інструментальну систему або як прикладну систему з конкретною прикладною базою знань. Існує тісний взаємозв'язок між технологією БД і систем БД - з одного боку, і технологією систем баз знань з іншого. Виникла тенденція "інтелектуалізації" систем БД. На зовнішньому рівні їх архітектури реалізують різноманітні семантичні моделі даних, створюють "дружні" інтерфейси для користувачів, хоча традиційні СКБД є необхідною складовою частиною інструментарію керування даними в системах баз знань.

Наявна література з баз даних і знань не в повній мірі задовольняє потреби студентів вузів, інженерно-технічних працівників і інших спеціалістів, які займаються програмуванням та обробкою даних. Відчувається потреба в навчальному посібнику, в якому б з єдиних методологічних позицій викладались основи баз даних і знань від організації їх структур до систем керування.

В навчальному посібнику приведені основи організації баз даних і знань на концептуальному, фізичному та логічному рівнях, а також особливості керування ними.

1 ОСНОВНІ ПОНЯТТЯ ПРО БАНКИ ДАНИХ

1.1 Визначення і класифікація інформаційних систем

Трудова діяльність людини постійно пов'язана зі сприйняттям і накопиченням інформації про навколишнє середовище, відбором і обробкою інформації при розв'язуванні різних задач, обміном нею з іншими людьми. З часом комплекс цих операцій, методи і засоби їхньої реалізації послужили основою для створення інформаційних систем, основне призначення яких - інформаційне забезпечення користувача, тобто надання йому необхідних даних із визначеної предметної області. Завдяки появі ЕОМ стало можливим створення автоматизованих інформаційних систем (АІС).

У розвитку АІС намітилися два покоління:

1-е покоління - інформаційні системи, які базуються на автономних файлах. Це системи з простою архітектурою й обмеженим набором можливостей. Вони складаються із набору автономних файлів і комплексу прикладних програм, призначених для обробки цих файлів і видачі документів. Такі системи мають ряд серйозних недоліків, що обмежують їхнє широке застосування: високу надлишковість даних, складність ведення і спільної обробки файлів, залежність програм від даних і ін.

2-е покоління - банки даних. Це системи з високим ступенем інтеграції даних і автоматизації керування ними. Вони орієнтовані на колективне користування й, в основному, позбавлені недоліків, властивих АІС 1-го покоління.

Функціонування АІС пов'язано з накопиченням і обробкою інформації. Під інформацією розуміється сукупність знань про фактичні дані і залежності між ними. У ЕОМ поняття інформації і даних часто ототожнюються. Але якщо бути точними, то дані - це інформація, подана у

формі, необхідній для введення її в ЕОМ, збереження, обробки і видачі споживачам.

Інформація, яка вводиться в АІС і видається системою користувачу, представляється у вигляді документів. Документ - це матеріальний об'єкт, який містить інформацію, що має відповідно до чинного законодавства правове значення, і призначений для передачі і використання. Джерелом інформації в АІС є люди і датчики, споживачами - люди (користувачі).

Звертання користувачів до АІС здійснюється у вигляді запитів. Запит - це формалізоване повідомлення, що надходить на вхід системи. Він включає умову на пошук даних, а також вказівку про те, що необхідно проробити зі знайденими даними.

Інтерпретація введених запитів, виконання дій, зазначених у них, формування і виведення повідомлень і документів складають основні етапи роботи АІС. У цілому під автоматизованою інформаційною системою розуміється сукупність інформаційних масивів, технічних, програмних і мовних засобів, призначених для збору, збереження, пошуку, обробки і видачі даних за запитами користувачів.

Використання АІС може здійснюватися одним із двох способів, суть яких це:

1. Автономне функціонування системи, при якому АІС не входить до складу інших систем і використовується самостійно. Прикладом можуть служити такі АІС, як документальні (бібліотечні) інформаційно-пошукові системи, а також системи резервування авіа- і залізничних квитків типу "Сирена" і "Експрес", у яких відповіддю на запит пасажира є документ у вигляді квитка або повідомлення про відсутність вільних місць.

2. Використання АІС в якості складової частини іншої автоматизованої системи. У цьому випадку вихідні дані можуть використовуватися не тільки кінцевими користувачами, але й іншими користувачами цієї автоматизованої системи з метою подальшої обробки і застосування у виробничому процесі. Так, у навчальних системах АІС

містить досліджуваний матеріал, набір питань, задач і відповідей, у САПР - нормативно-довідкову інформацію, зведення про ДСТ і інші дані, в АСУ - всю інформацію, необхідну для керування підприємством, тобто для аналізу, оцінки, прогнозування, виробітку рішень, планування, контролю виконання.

Інформаційні системи можна класифікувати за рядом ознак. В основу класифікації, приведеної на рис.1.1, покладені найбільш істотні ознаки, що характеризують можливості й особливості сучасних АІС.

Документальні інформаційно-пошукові системи (ДІПС) призначені для збереження і обробки документальних даних - адрес збереження документів. Такі дані подаються в неструктурованому вигляді. Прикладом ДІПС є бібліотечні, бібліографічні АІС. На відміну від систем цього класу фактографічні інформаційно-пошукові системи (ФІПС) зберігають і оброблюють фактографічну інформацію - структуровані дані у вигляді чисел і текстів. Над такими даними можна виконувати різні операції. Більшість АІС являють собою системи класу ФІПС.

Друга ознака класифікації поділяє інформаційні системи на дві групи: до першої відносяться інформаційно-довідкові системи (ІДС), які виконують пошук і виведення інформації без її обробки. Автоматизовані інформаційні системи обробки даних (АІСОД, ІСОД), що відносяться до другої групи об'єднують у собі інформаційно-довідкову систему і систему обробки даних. Обробка знайдених даних виконується комплексом прикладних програм. Більшість АІС побудована за принципом ІСОД.

Ступінь інтеграції даних і автоматизації керування ними є найважливішою ознакою класифікації АІС. У ранніх системах - АІС на автономних файлах (АІС АФ) - принцип інтеграції даних практично не використовувався, а рівень автоматизації керування файлами був порівняно низьким. Такі системи застосовуються і в даний час; вони ефективні у випадку вузького, спеціалізованого використання невеликим колом осіб. Високий ступінь інтеграції мають банки даних (БнД).

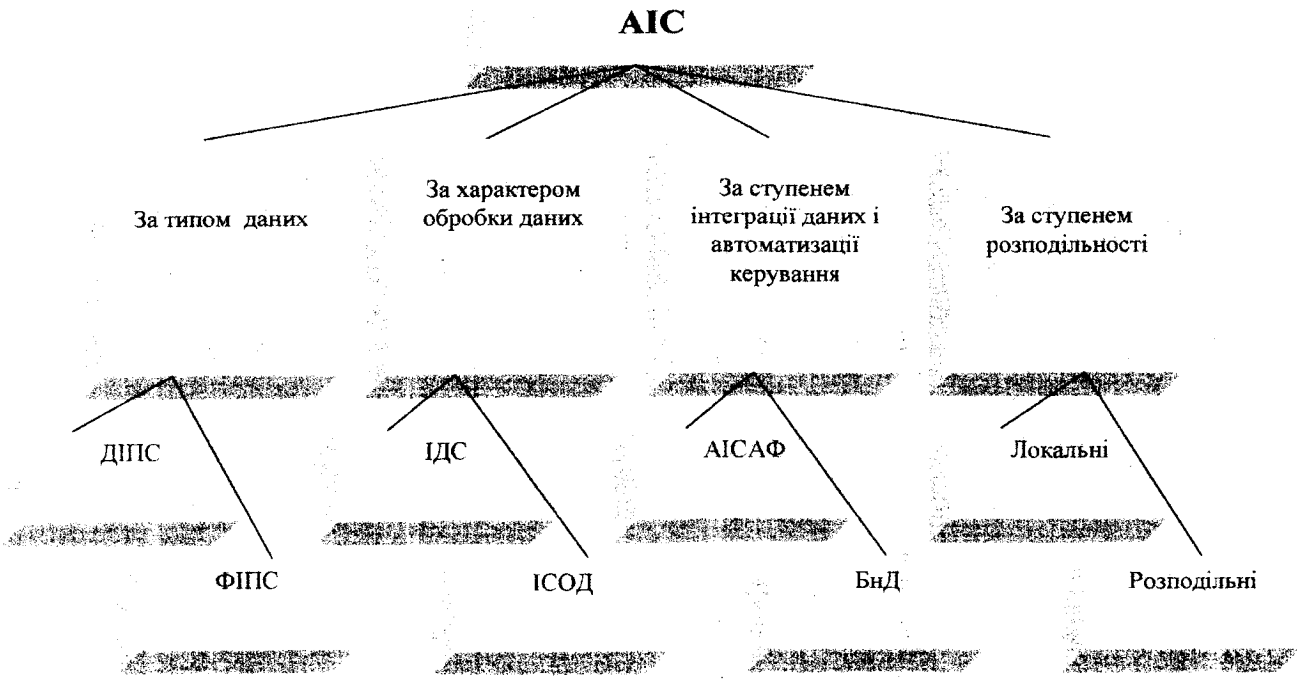


Рисунок 1.1 - Класифікація автоматизованих інформаційних систем

У порівнянні з АІС на автономних файлах у БнД збережена інформація зосереджена в єдиному інформаційному масиві - базі даних (БД), а процес маніпулювання даними автоматизований.

Останній із приведених ознак класифікації враховує розподільність компонентів АІС: локальна система розміщена на одній ЕОМ, у той час як розподілена система функціонує в середовищі обчислювальної мережі і розподілена по її вузлах (серверах і робочих станціях).

Контрольні запитання

1. Дайте визначення автоматизованої інформаційної системи.
2. За якими ознаками класифікують автоматизовані інформаційні системи?
3. Які недоліки мають автоматизовані інформаційні системи на автономних файлах?
4. Які автоматизовані інформаційні системи мають найбільшу ступінь інтеграції?

1.2 Автоматизовані банки даних

Автоматизований банк даних (БнД) - це організаційно-технічна (людино-машинна) система, що представляє собою сукупність інформаційної бази, колективу фахівців і комплексу програмних і технічних засобів забезпечення її функціонування, призначена для збереження, пошуку і видачі інформації у вигляді, зручному для користувачів.

На загальній схемі структури автоматизованого банку даних (рис.1.2) прийняті такі позначення: БД - база даних; ПП - прикладні програми користувачів банку даних; СКБД - система керування базою даних, тобто комплекс програмних засобів, який забезпечує завантаження інформації в базу даних, реорганізацію та ведення бази, пошук та перетворення інформації для забезпечення роботи програм користувачів банку даних.

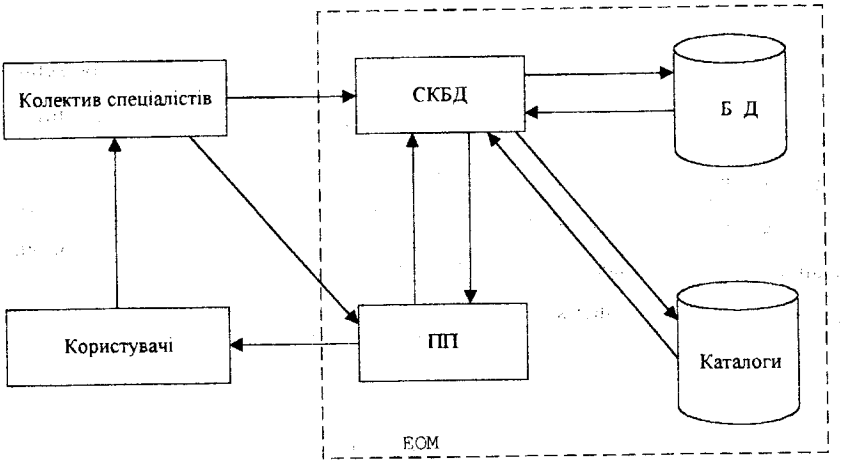


Рисунок 1.2 – Структура автоматизованого банку даних

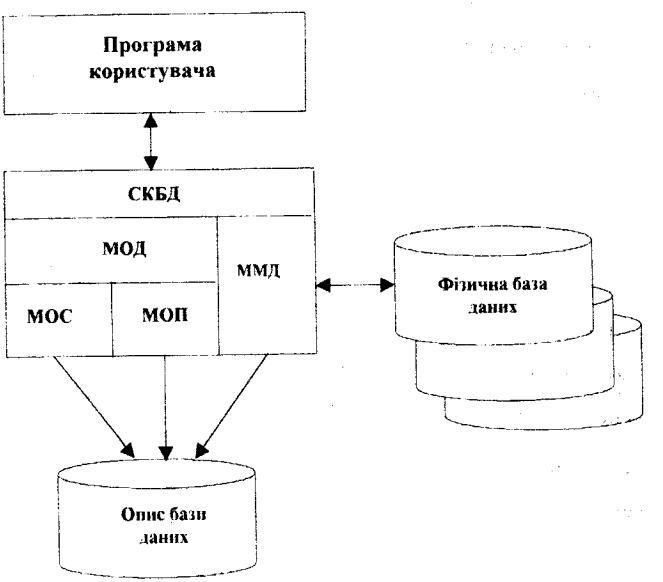


Рисунок 1.3 – Архітектура СКБД

більш детально структурні складові частини цього автоматизованого банку даних.

База даних. Відомі два підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але характеризується рядом недоліків.

Характерна риса файлового підходу - вузька спеціалізація як обробних програм, так і файлів даних, що служить причиною великої надлишковості, тому що ті самі елементи даних зберігаються в різних системах. Поскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше області. Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів подання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зв'язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань. Вказані причини обумовили появу баз даних.

База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні "файл-програма", вона явним чином декларується в базі.

База даних орієнтована на інтегровані запити, а не на одну програму, як у випадку файлового підходу, і використовується для інформаційних

потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою СКБД.

СКБД - це складна програмна система накопичення та з наступним маніпулюванням даними, що представляють інтерес для користувача. Кожній прикладній програмі СКБД надає інтерфейс з базою даних та має засоби безпосереднього доступу до неї. Таким чином, СКБД відіграє центральну роль в функціонуванні автоматизованого банку даних.

Архітектурно СКБД складається з двох великих компонент (рис.1.3). За допомогою мови опису даних (МОД) створюються описи елементів, груп та записів даних, а також взаємозв'язки між ними які, як правило, задаються у вигляді таблиць. В залежності від конкретної реалізації СКБД мову опису даних підрозділяються на мову опису схеми бази даних (МОС) та мову опису підсхем бази даних (МОП). Слід особливо зазначити, що МОД дозволяє створити не саму базу даних, а лише її опис.

Для виконання операцій з базою даних в прикладних програмах використовується мова маніпулювання даними (ММД). Фактична структура фізичного зберігання даних відома тільки СКБД.

З метою забезпечення зв'язків між програмами користувачів і СКБД (що особливо важливо при мультипрограмному режимі роботи операційної системи) в СКБД виділяють особливу складову - резидентний модуль системи керування базами даних. Цей модуль значно менший від всієї СКБД, тому на час функціонування автоматизованого банку інформації він може постійно знаходитись в основній пам'яті ЕОМ та забезпечувати взаємодію всіх складових СКБД і програм, які до неї звертаються.

Приведена структура притаманна усім СКБД, котрі розрізняються обмеженнями та можливостями по виконанню відповідних функцій. Отже, процес порівняння і оцінки таких систем для одного конкретного

застосування зводиться до співставлення можливостей наявних СУБД з вимогами користувачів.

Будь-який аналіз, необхідний для проведення оцінки і вибору СКБД, повинен починатись з ретельного вивчення потреб користувачів. При цьому виконується опис зв'язків між елементами даних в базі даних, а також для оцінки експлуатаційних характеристик визначаються вимоги до часу виконання кожної транзакції. Під транзакцією розуміється повідомлення, яке передається від прикладної програми до СКБД. Транзакція ініціює в останній роботу певного виду чи окрему операцію по обробці даних.

За допомогою мови опису даних адміністратор, а іноді і інші програмісти описують для СКБД вміст та структуру бази даних. При розгляданні МОД слід з'ясувати її власні характеристики (наприклад, простоту використання та наочність), а також проаналізувати обмеження СКБД на дані (наприклад, типи даних чи можливості по обмеженню доступу).

Засоби маніпулювання даними визначають методи доступу до бази даних та мову (мови), за допомогою якої відбувається цей доступ. Мова маніпулювання даними є засобом, який застосовується користувачами чи прикладними програмістами для виконання операцій над базою даних. При порівнянні можливостей СКБД з сучасними принципами обробки даних важливе значення має зв'язок ММД з існуючими мовами програмування. Простота її вивчення і використання розширює можливості розробки банку даних з конкретною СКБД. Степінь процедурності ММД визначає міру незалежності програм від даних. Чим менш процедурна мова, тим менша ймовірність зміни програм написаних з її використанням, при включенні нових методів доступу і навіть нових структур даних в СКБД.

Нарешті, засоби маніпулювання даними визначають можливості паралельної обробки. Для збереження цілісності бази даних в умовах її одночасного використання декількома прикладними програмами звичайно вводяться обмеження на доступ для всіх, крім одного, з процесів, які

виконуються одночасно. Наприклад, якщо програма поновлює запис бази даних, то їй може надаватись доступ до поля, що змінюється, до запису чи до всього фізичного файлу, в який цей запис входить.

Основна мета застосування баз даних – забезпечити незалежність логічної бази даних та прикладних програм від методів зберігання фізичної бази даних. При цьому способи доступу значно впливають на експлуатаційні характеристики банку даних.

Не менш важливу роль відіграють засоби копіювання та відновлення бази даних. Найбільш суттєвим тут є наявність автоматичного режиму ведення журналу фіксування роботи з базою даних. Наявність стандартних програм СКБД таких, як програма перезавантаження бази даних і програма обробки журналу, спрощують процес відновлення бази даних після апаратурних збоїв.

Серед інших стандартних програм слід відзначити програми завантаження бази даних, трасування роботи з базою даних для полегшення налагодження, а також накопичення і аналізу статистики по експлуатаційних характеристиках. Якщо база даних призначена для використання прикладними програмами, які функціонують в діалоговому режимі, то особливу роль в складі СКБД відіграють засоби передачі даних.

Концепція баз даних припускає інтеграцію даних, що раніше зберігалися окремо. Незалежно від того, охоплює чи не охоплює така інтеграція централізацію фізичних даних, вона припускає ріст спільного використання даних різними прикладними програмами та зменшення надлишковості зберігання одних і тих же даних. Для створення цим процесам сприятливих умов потрібне централізоване керування вмістом баз даних. Об'єктами централізованого керування є форма елементів даних і структури баз даних, а не власне значення самих елементів даних. Функція керування формою та вмістом бази даних називається адмініструванням даних.

Керування вмістом бази даних відбувається шляхом збору і ведення точної та повної інформації про дані. Ця інформація, яку часто називають метаданими, включає опис смислу елементів даних, методів їх використання, джерел, фізичних характеристик, а також різних правил та обмежень. Метадані дозволяють проводити аналіз запитів по нових даних, проектування і програмування нових прикладних систем, супроводження існуючих систем та документування всіх етапів розвитку бази даних.

Дані про базу даних, чи метадані, можливо розбити на три класи: семантична інформація, фізичні характеристики та інформація про використання. Засобами автоматизації формування та використання метаданих являються словники даних (системи словників-довідників даних). Перерахуємо їхні основні функції:

- встановлення зв'язку між користувачами БД;
- здійснення простого та ефективного керування елементами даних при вводі в систему як нових елементів, так і при зміні опису існуючих;
- зменшення надлишковості;
- усунення протиріччя даних;
- централізація керування елемсітами даних з метою спрощення проектування БД та її розширення.

Для створення ефективного і зручного словника даних необхідно при зборі інформації про дані встановити правила присвоєння елементам імен, добитися однозначного тлумачення різними користувачами призначення джерел і угод по присвоєнню імен, сформулювати прийнятні для усіх користувачів описи елементів даних і виявити синоніми, усунути багатозначність (омонімію, полісемію). Вказаний процес виконується ітеративно і зв'язаний з усуненням конфліктних ситуацій.

В процесі роботи з словником даних є можливість отримати в алфавітному порядку лістинг найменувань типів всіх статей (з зазначенням частоти їх використання в системі, статусу кожної статті та числа структур кожного виду).

Словник даних використовується кінцевими користувачами при роботі з системою на мові запитів, прикладними програмістами – при написанні програм, системними програмістами – в процесі розвитку системи. Словник в умовах організації інформації у вигляді баз даних вводиться до складу опису баз даних та використовується СКБД при роботі компілювальних і інтерпретувальних програм.

З розвитком системи необхідно проводити контроль логічності та повноти даних, які оброблюються в системі. Цей контроль полягає в перевірці за допомогою словника даних відповідності потоків і елементів даних, потоків і джерел даних, процесів обробки та елементів даних.

Особливо важливий словник даних при взаємодії декількох систем обробки даних, при побудові розподілених банків даних, при використанні програм, які виконані в інших організаціях. В останньому випадку словарні статті вилучають з написаних програм, встановлюють синонімічні зв'язки їх зі статтями словника системи і переводять їх в формат, принятий в системі.

Колектив спеціалістів, який забезпечує функціонування автоматизованого банку даних, складається з адміністратора, аналітиків, системних та прикладних програмістів. Взаємодія їх між собою та кінцевими користувачами показана на рис.1.4. Розглянемо детальніше функції, покладені на кожного з перерахованих спеціалістів.

Адміністратор – це спеціаліст, який володіє інформацією про інформаційні потреби кінцевих користувачів, працює в тісному контакті з користувачами і відповідає за визначення, завантаження, захист та ефективність експлуатації баз даних.

Необхідність включення адміністратора в колектив спеціалістів стала актуальною в той період, коли виникла необхідність в централізації обробки даних, що призвело до переходу від файлових систем до інтегрованих баз даних. Оскільки користувачі повинні обслуговуватись усіма засобами автоматизованого банку даних, то адміністратор є

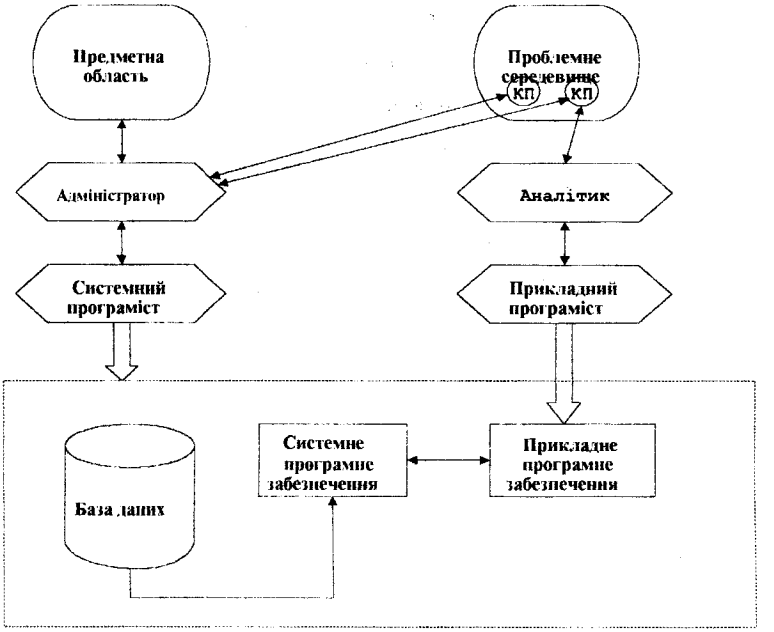


Рисунок 1.4 – Схема взаємодії колективу спеціалістів банку

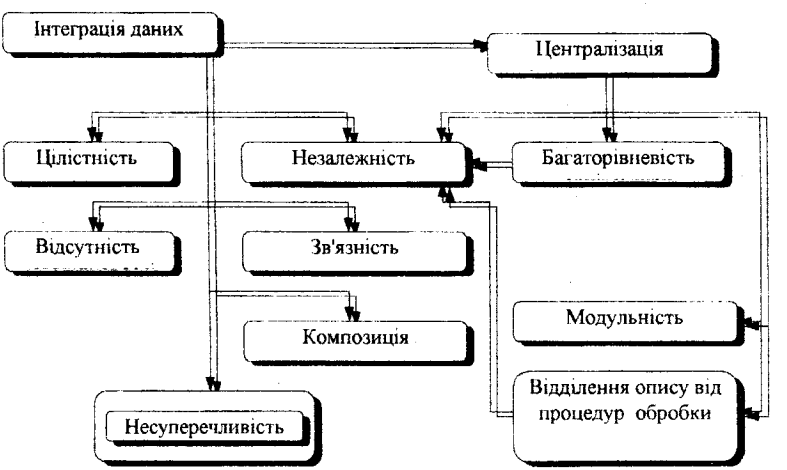


Рисунок 1.5 – Основні принципи побудови банків інформації

відповідальним за аналіз потреб користувачів, проектування бази даних, її впровадження, поновлення та реорганізацію.

Всі задачі, виконання яких покладається на адміністратора, можна розділити у відповідності з етапами розробки автоматизованих банків інформації на чотири групи: планування, проектування, експлуатація та використання.

При плануванні адміністратор бере участь у виборі програмного забезпечення, яке пов'язано з базою даних, та обладнання. Йому доводиться працювати з кінцевими користувачами, щоб встановити реальні цілі та вимоги до прикладних програм і баз даних. Адміністратор повинен гарантувати, що пріоритети розробки та експлуатації прикладних програм відповідають цілям різних категорій користувачів, приймає участь в довгостроковому плануванні, в тому числі у визначенні перспектив розширення бази даних.

При проектуванні адміністратор надає розробникам прикладних систем необхідні засоби для проектування логічної та фізичної баз даних, керує процесом логічного проектування з метою отримання повної картини ресурсів даних. При появі нових вимог до даних адміністратор банку визначає метод, за допомогою якого ці дані можна включити в склад існуючої бази даних, і керує процесом виконання необхідних змін. Він також здійснює вибір методів доступу і методів розміщення даних в фізичній пам'яті, що дозволяє забезпечити виконання вимог прикладних програм.

На етапі експлуатації в обов'язки адміністратора також входить розробка і контроль дій, які гарантують збереження цілісності бази даних, включаючи процедури її копіювання і відновлення, а також організації захисту бази даних за допомогою механізмів керування доступом і засобів СКБД.

І нарешті, адміністратору доводиться взаємодіяти з користувачами бази даних, тому він вводить стандарти на вміст і використання бази даних,

роблячи їх доступними для потенційних користувачів, супроводжує спеціальні засоби програмного забезпечення роботи з базою даних (словники даних, мови запитів). Крім того, він може консультувати користувача бази даних по застосуванню окремих елементів чи усього програмного забезпечення.

Системні програмісти займаються створенням базового математичного забезпечення для ЕОМ. Вони "генерують" операційні системи, СКБД, транслятори з різних мов, розробляють сервісні програми і інші програмні засоби, які забезпечують обробку інформації та вирішення задач на ЕОМ.

Аналітик, користуючись знаннями закономірностей певного проблемного середовища, розробляє його математичну модель, застосовує необхідні математичні методи і методи моделювання. Він оцінює альтернативні варіанти і приймає рішення в залежності від заданих вимог.

Аналітик переводить задачі кінцевого користувача (КК) в деяку вихідну формальну модель. Оскільки КК не являється математиком, то задачу він формулює на мові своєї професії, і все мистецтво аналітика полягає в умінні з цього формулювання побудувати адекватну йому математичну задачу. Результатом роботи аналітика є вихідне представлення задачі для прикладного програміста, метою котрого є перетворення продукту аналітика в програмний продукт, придатний для вводу в ЕОМ.

Контрольні запитання

1. Назвіть основні складові частини автоматизованого банку інформації.
2. Які функції виконує СКБД в банках даних?
3. Яке призначення словника даних?
4. Чим відрізняються МОД та ММД?
5. В чому проявляється інтегрований характер бази даних?
6. В чому полягає відмінність функцій прикладного програміста і аналітика?

1.3 Вимоги до банків даних

Різноманіття інформаційних потреб висуває до банків даних підвищені вимоги. До основних вимог відносяться:

- Адекватність інформації стану предметної області. БнД є інформаційною моделлю предметної області і, як відзначалося вище, інформація, яка зберігається в ньому, повинна повно і точно відображати її об'єкти, їхні властивості і відношення між об'єктами. Відступ від принципу адекватності робить систему марною і навіть небезпечною, неприпустимою для використання. У свою чергу, вимога адекватності породжує ряд нових вимог до системи таких, як необхідність постійного внесення змін у дані і періодичної зміни організації даних.
- Надійність функціонування - одна з найважливіших вимог, які висуваються до будь-якої системи.
- Швидкодія і продуктивність. Ці дві близькі одна до одної вимоги відображають часові потреби користувачів. Перша з них визначається часом відповіді (реакції) системи на запит, який відраховується з моменту введення запиту до моменту початку видачі знайдених даних. Цей час залежить не тільки від швидкодії ЕОМ, але і від способів фізичної організації даних, методів доступу, способів пошуку, складності запиту й інших чинників. Друга вимога визначається кількістю запитів, які відпрацьовуються в одиницю часу.
- Простота і зручність використання. Ця вимога висувається до БнД з боку всіх без винятку категорій користувачів, особливо кінцевих. Складність запитів, відсутність сервісу формують у психології користувача небажання працювати з інформаційною системою.
- Масовість використання. Сучасна інформаційна система повинна забезпечувати колективний доступ користувачів, при якому вони

можуть одночасно і незалежно звертатися до баз даних для одержання необхідних даних.

- Захист інформації. Система повинна забезпечувати захист збережених у ній даних і програм як від випадкових спотворень і знищення, так і від навмисних, несанкціонованих дій користувачів.
- Можливість розширення. Архітектура системи повинна допускати розширення її можливостей шляхом модифікації або заміни існуючих програмних модулів, додаванням нових компонентів, а також шляхом реорганізації інформаційних масивів.

Контрольні запитання

1. Які вимоги до БнД відображають часові потреби користувачів?
2. Як Ви розумієте адекватність інформації стану предметної області?
3. Якими шляхами може бути розширена БнД?
4. Які типи захисту інформації повинні бути передбачені в БнД?

1.4 Принципи побудови банків даних

У основі побудови БнД лежать наукові принципи, на основі яких розроблюють високоякісні системи, які відповідають сучасним вимогам. Вибір принципів побудови БнД і їхнє втілення в конкретній системі складають основу проектування.

З множини використовуваних принципів виділимо найбільш істотні (рис.1.5): принцип інтеграції даних і принцип централізації керування ними. Обидва принципи відображають суть банку даних: інтеграція є основою організації БнД, централізація керування - основою організації і функціонування системи керування базами даних (СКБД). Інші принципи в тій або іншій мірі пов'язані з першими. Окремі з них є їхнім результатом або одним із можливих шляхів реалізації. Так, інтеграція даних припускає взаємозалежність (зв'язність) даних; зв'язність, у свою чергу разом із принципом композиції дозволяє звести надлишковість даних до мінімуму.

Суть, принципу інтеграції даних полягає в об'єднанні окремих, взаємно не зв'язаних даних у єдине ціле, в ролі якого виступає база даних. В результаті вказаного користувачу і його прикладним програмам всі дані представляються єдиним інформаційним масивом. При цьому полегшуються пошук взаємозалежних даних і їхня спільна обробка, зменшується надлишковість даних, спрощується процес ведення БнД.

Інтеграцію даних необхідно розглядати на двох рівнях - логічному і фізичному. На логічному рівні множина структур даних відображається в єдину структуру даних, на фізичному рівні автономні файли об'єднуються в базу даних.

Принцип цілісності даних відображає вимогу адекватності збереженої в БнД інформації стану предметної області: у будь-який момент часу дані повинні в точності відповідати властивостям і характеристикам об'єктів. Порушення цілісності виникає внаслідок спотворення або навіть руйнації (стирання) усіх або частини даних, а також як результат запису в базу даних невірної інформації. Підтримка цілісності досягається за рахунок контролю вхідної інформації, періодичної перевірки збережених у БнД даних, застосуванням спеціальної системи відновлення даних, а також іншими заходами.

Під незалежністю даних будемо розуміти незалежність прикладних програм від збережених даних, при якій будь-які зміни в організації даних не вимагають корекції цих програм. Одним із шляхів досягнення незалежності є введення додаткових рівнів абстрагування даних (принцип багаторівневості). Замість двох традиційних рівнів, передбачених базовим програмним забезпеченням і стандартними мовами програмування, - логічного і фізичного - в архітектурі БнД використовується принцип трирівневої організації даних: логічний рівень ділиться на два - зовнішній (рівень користувача) і концептуальний (загальний системний рівень даних).

Інший шлях досягнення незалежності даних - передача СКБД частини функцій, що раніше покладалися на прикладні програми. Маються на увазі функції, зв'язані з організацією доступу до БнД. При цьому прикладна програма ніяк не зв'язана ні з БнД, ні з методом доступу. Вона лише формує і передає ядру інформацію, необхідну для пошуку даних.

Незалежність даних досягається також застосуванням і дотриманням принципу відділення опису БнД від процедур обробки даних. Нарешті, істотним чинником забезпечення незалежності варто вважати реляційний підхід до побудови БнД - розробку бази даних на основі реляційної моделі даних і використання методів і засобів реляційної алгебри в процесі обробки БнД. Найбільший ефект досягається раціональним сполученням усіх зазначених шляхів.

Відсутність надлишковості - це стан даних, коли кожний елемент присутній у БнД в єдиному екземплярі. Надлишковість може мати місце як на логічному рівні, коли в структурі даних повторюються ті самі типи даних, так і на фізичному рівні, коли дані зберігаються в двох або більше екземплярах. Принцип інтеграції дозволяє звести надлишковість до мінімуму. Під несуперечливістю розуміється смислова відповідність між даними. Це такий стан бази даних, при якому збережені в ній дані не суперечать один одному. Розрізняють два аспекти несуперечливості: смислова відповідність різнотипних даних і ідентичність (рівність) дублюючих даних.

Принцип зв'язності даних полягає в тому, що дані в БнД взаємозалежні, і зв'язки відбивають відношення між об'єктами предметної області. Множина типів даних і множина зв'язків утворять логічну структуру даних. Наявність зв'язків між записами в БнД дозволяє зменшувати надлишковість, спростити і прискорити пошук даних.

Принцип централізації керування полягає в передачі усіх функцій керування даними єдиному комплексу керуючих програм - системі керування базами даних. Як було зазначено вище, всі запити до

доступом до БнД, виконуються не прикладними програмами, а централізовано - ядром СКБД - на підставі інформації, яку отримують з цих програм. Дотримання цього принципу дозволяє автоматизувати роботу з базами даних і тим самим істотно підвищити ефект, який отримують від застосування інформаційної системи.

Відділення опису даних від процедур їхньої обробки припускає, що опис даних виключається з прикладних програм, складається і транслюється окремо від них і зберігається в базі даних (або поза нею у вигляді окремого файла). Виведення цих описів за рамки прикладної програми робить її більш незалежною від БнД, полегшує процес програмування, зменшує розміри необхідної для програми пам'яті, підвищує гнучкість маніпулювання даними.

На основі зазначених вище принципів формується архітектура БнД - концепція взаємозв'язку логічних, фізичних і програмних компонентів системи.

Контрольні запитання

1. Назвіть основні принципи побудови банків інформації.
2. Для чого необхідно забезпечити відокремлення опису даних від процедур їхньої обробки?
3. В чому полягає принцип трирівневої організації даних?
4. Покажіть взаємозв'язок основних принципів побудови банків даних.

2 ІНФОЛОГІЧНА МОДЕЛЬ ДАНИХ

2.1 Основні поняття

Проектування бази даних треба починати з аналізу предметної області і виявлення вимог до неї окремих користувачів (співробітників організації, для яких створюється база даних). Об'єднуючи власні уявлення про зміст бази даних, отримані в результаті опитування користувачів, і свої уявлення про дані, що можуть знадобитися в майбутніх додатках, створюється узагальнений неформальний опис утворюваної бази даних. Цей опис, виконаний із використанням природної мови, математичних формул, таблиць, графіків і інших засобів, зрозумілих усім людям, що працюють над проектуванням бази даних, називають інфологічною моделлю даних (рис.2.1).

Така модель цілком незалежна від фізичних параметрів середовища збереження даних. Інфологічна модель не повинна змінюватися доти, поки якісь зміни в реальному світі не приведуть до зміни в ній деякого визначення, щоб ця модель продовжувала відображувати предметну область. Інші моделі, показані на рис.1.3, є комп'ютерно-орієнтованими. З їхньою допомогою СКБД дає можливість програмам і користувачам здійснювати доступ до збережених даних лише за їхніми іменами, не турбуючись про фізичне розташування цих даних. Потрібні дані відшукуються СКБД на зовнішніх запам'ятовувальних пристроях по фізичній моделі даних. Оскільки зазначений доступ здійснюється за допомогою конкретної СКБД, то моделі повинні бути описані мовою опису даних цієї СКБД.

Такий опис, утворений по інфологічній моделі даних, називають

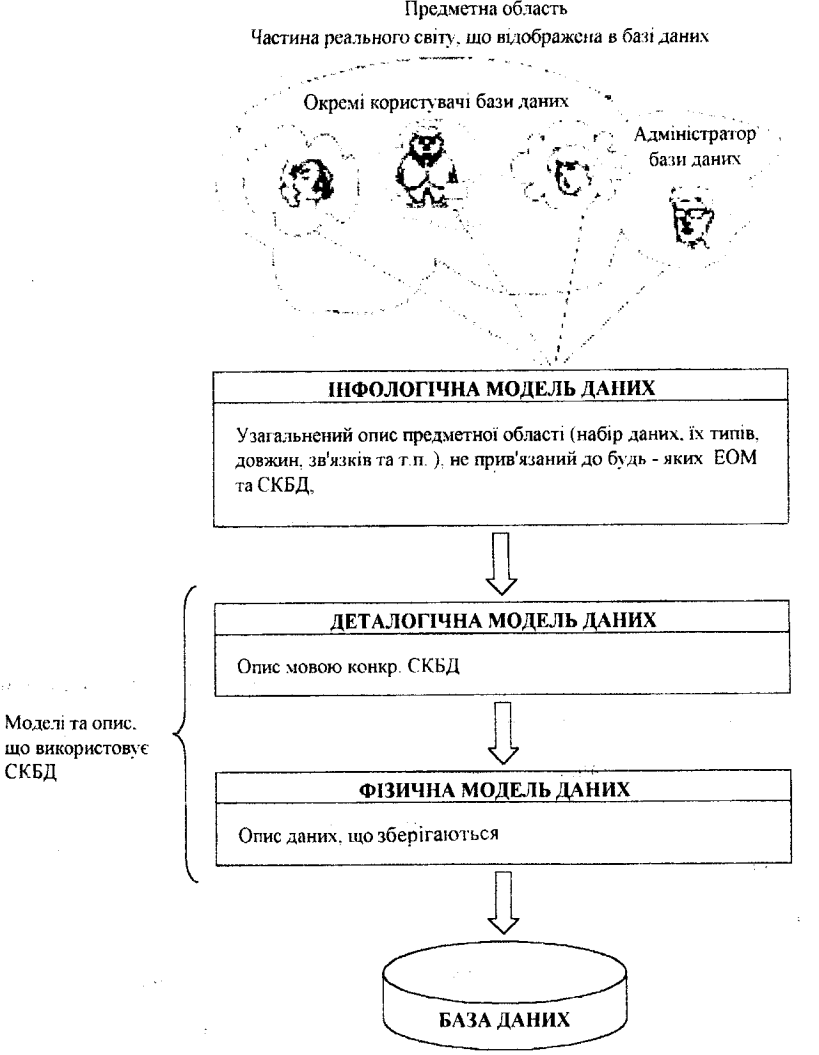


Рисунок 2.1 – Рівні моделей даних

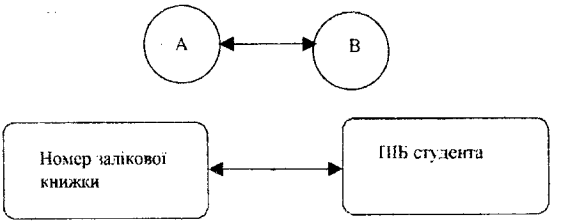


Рисунок 2.2 – Приклад відображення ОДИН-ДО-ОДНОГО

даталогічною моделлю даних.

Трирівнева архітектура (інфологічний, даталогічний і физичний рівні) дозволяє забезпечити незалежність збережених даних від програм, в яких вони використовуються. Можна при необхідності переписати збережені дані на інші носії інформації і (або) реорганізувати їхню фізичну структуру. Можна підключити до системи будь-яке число нових користувачів (нових додатків), доповнивши, якщо треба, даталогічну модель. Зазначені зміни фізичної і даталогічної моделей не будуть помічені користувачами системи, так само як не будуть помічені і нові користувачі. Отже, незалежність даних забезпечує можливість розвитку системи баз даних без руйнування існуючих додатків.

Таким чином, інфологічна модель відображає реальний світ у деякій зрозумілі людині концепції, цілком незалежній від параметрів середовища збереження даних. Існує безліч підходів до побудови таких моделей: графові моделі, семантичні мережі, модель "сутність-зв'язок" і т.д. Найбільш популярною серед них виявилася модель "сутність-зв'язок".

Мета інфологічного моделювання - забезпечення найбільш природних для людини засобів збору й подання інформації, що буде зберігатися в створюваній базі даних. Тому інфологічну модель даних намагаються будувати за аналогією з природною мовою (остання не може бути використаною у чистому вигляді через складність комп'ютерної обробки текстів і неоднозначності будь-якої природної мови). Основними конструктивними елементами інфологічних моделей є сутності, зв'язки між ними і їхні властивості (атрибути).

Сутність - будь-який помітний об'єкт (об'єкт, що ми можемо відрізнити від іншого), інформацію про який необхідно зберегти в базі даних. Сутностями можуть бути люди, місця, літаки, рейси, колір і т.д. Слід розрізняти такі поняття, як тип сутності і екземпляр сутності. Поняття типу сутності відноситься до набору однорідних особистостей, предметів, подій або ідей, що виступають як ціле. Екземпляр сутності відповідає ...

конкретної речі в наборі. Наприклад, типом сутності може бути МІСТО, а екземпляром - Москва, Київ і т.д.

Атрибут - поіменована характеристика сутності. Його найменування повинно бути унікальним для конкретного типу сутності, але може бути однаковим для різноманітного типу сутностей (наприклад, КОЛІР може бути визначений для багатьох сутностей: СОБАКА, АВТОМОБІЛЬ, ДИМ і т.д.). Атрибути використовуються для визначення того, яка інформація повинна бути зібрана про сутність. Прикладами атрибутів для сутності АВТОМОБІЛЬ є ТИП, МАРКА, НОМЕРНИЙ ЗНАК, КОЛІР і т.д. Тут також існує розходження між типом і екземпляром. Тип атрибута КОЛІР має багато екземплярів або значень: Червоний, Синій, Банановий, Біла ніч і т.д., проте кожному екземпляру сутності присвоюється тільки одне значення атрибута.

Абсолютне розходження між типами сутностей і атрибутами відсутнє. Атрибут є таким тільки в зв'язку з типом сутності. У іншому контексті атрибут може виступати як самостійна сутність. Наприклад, для автомобільного заводу КОЛІР - це тільки атрибут продукту виробництва, а для лакофарбової фабрики КОЛІР - тип сутності.

Ключ - мінімальний набір атрибутів, за значеннями яких можна однозначно знайти необхідний екземпляр сутності. Мінімальність означає, що видалення із набору будь-якого атрибута не дозволяє ідентифікувати сутність по тих атрибутах, що залишилися.

Зв'язок - асоціювання двох або більше сутностей. Якби призначенням бази даних було тільки збереження окремих, не пов'язаних між собою даних, то її структура могла б бути дуже простою. Проте одне з основних вимог до організації бази даних - це забезпечення можливості знаходження одних сутностей за значеннями інших, для чого необхідно встановити між ними визначені зв'язки. Оскільки в реальних базах даних нерідко присутні сотні або навіть тисячі сутностей, то теоретично між ними може бути

встановлено більше мільйона зв'язків. Наявність такої множини зв'язків і визначає складність інфологічних моделей.

Контрольні питання

1. Що Ви розумієте під інфологічною моделлю даних?
2. Що таке даталогічна модель даних?
3. Що Ви розумієте під фізичною моделлю даних?
4. Що є метою інфологічного моделювання?
5. Дайте визначення основних конструктивних елементів інфологічної моделі даних (сутність, екземпляр сутності, атрибут, ключ, зв'язок).

2.2 Характеристика зв'язків

Структура даних може бути описана формально. Опис глобальної логічної структури бази даних називається схемою. Схема визначає всі типи елементів даних, які зберігаються в базі даних, а також усі зв'язки між ними. Схема бази даних, як правило, дуже складна. Конкретний користувач або прикладний програміст не повинен знати про схему в цілому. Така необізнаність часто навіть необхідна з точки зору безпеки даних. Програміст або користувач повинен бути інформований тільки про множину даних і зв'язків, які орієнтовані на його конкретну область.

Частина схеми отримала назву підсхеми. По суті, підсхема - це деяка організація файлів прикладного програміста. В функції СКБД входить побудова відповідних підсхем із загальної схеми і передача даних користувачам і системним програмістам. При цьому схема даних повина бути спроектована таким чином, щоб з неї могли бути побудовані всі підсхеми за запитами користувачів або прикладних програм. Ні схема, ні підсхема не визначають методів фізичного зберігання даних.

Схеми і підсхеми представляють у вигляді діаграм, на яких зображують типи елементів даних і зв'язки між ними. Розрізняють чотири види зв'язків:

1) необов'язковий зв'язок: існування об'єктів не залежить від зв'язку;
2) можливий зв'язок: існування одного з об'єктів залежить від зв'язку;
3) умовний зв'язок: частковий вид можливого зв'язку, коли задається умова існування (наприклад, зв'язок між об'єктами СТУДЕНТ, СТИПЕНДІЯ можлива при умові відповідної успішності);
4) обов'язковий зв'язок: існування обох об'єктів залежить від зв'язку.

Односторонні зв'язки між парами елементів називаються асоціаціями, а двосторонні - відображеннями.

Між двома сутностями А й В можливі чотири типи зв'язків:

- ✓ **перший тип** - зв'язок ОДИН-ДО-ОДНОГО (1:1): за допомогою такого відображения подають такий тип зв'язку, коли в кожний момент часу кожний екземпляр елемента, від якого направлений зв'язок, ідентифікує один і тільки один екземпляр елемента, до якого направлений зв'язок, при цьому ця ідентифікація є унікальною в обох напрямках. Приклад відображения 1:1 приведено на рис.2.2. Якщо відомо значення А, то однозначно визначається і значення В. І навпаки.
- ✓ **другий тип** – зв'язок ОДИН-ДО-БАГАТЬОХ (1:Б): якщо екземпляр елемента даних, від якого направлений зв'язок, ідентифікує деяке число екземплярів елементів даних, до яких направлений зв'язок, причому ідентифікація в даному напрямку не обов'язково є унікальною, то таке відображення називається ОДИН-ДО-БАГАТЬОХ (1:Б). Прикладом такого відношення (рис.2.3) може бути НОМЕР ВІДДІЛУ - ТАБЕЛЬНІ НОМЕРИ ПРАЦІВНИКІВ. У відділі працює багато службовців, але кожний працівник відноситься тільки до одного відділу.

Оскільки між двома сутностями можливі зв'язки в обох напрямках, то існує ще **два типи зв'язків** БАГАТО-ДО-ОДНОГО (Б:1) і БАГАТО-ДО-БАГАТЬОХ (Б:Б). Відображення Б:1 є аналогічним відображенням 1:Б.

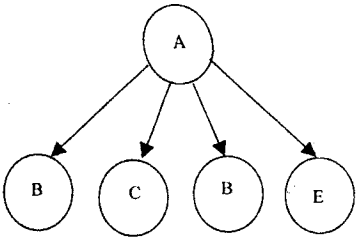


Рисунок 2.3 – Приклад відображення ОДИН-ДО-БАГАТЬОХ

Якщо екземпляр елемента даних, від якого направлений зв'язок, ідентифікує деяке число екземплярів елементів даних, до яких направлений зв'язок, і навпаки, тобто ідентифікація не є унікальною в обох напрямках, то таке відображення називається БАГАТО-ДО-БАГАТЬОХ (Б:Б).

Прикладом такого відношення (рис.2.4) є відношення ВИКЛАДАЧІ-СТУДЕНТИ. Кожний студент "пов'язаний" з багатьма викладачами і кожний викладач читає лекції різним групам студентів.

Характер зв'язків між сутностями не обмежується переліченими. Існують і більш складні зв'язки:

- множина зв'язків між одними й тими ж сутностями;

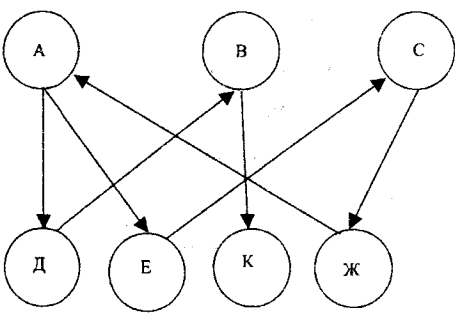


Рисунок 2.4 – Приклад відображення БАГАТО-ДО-БАГАТЬОХ

Наприклад, пацієнт, маючи одного лікаря, що лікує, може мати також декілька лікарів-консультантів; лікар може бути лікарем, що лікує декількох пацієнтів та може одночасно консультувати декількох інших пацієнтів.

- тренарні зв'язки;

Наприклад, лікар може призначити декільком пацієнтам декілька аналізів, аналіз може бути призначений декількома лікарями декільком пацієнтам, й пацієнту може бути призначено декілька аналізів декількома лікарями).

- зв'язки більш високих порядків, семантика (зміст) яких іноді дуже складна.

Існує три типа асоціацій:

– асоціація типу 1 (проста);
– асоціація типу М (складна);
– асоціація типу С (умовна).

В простій асоціації типу 1 (рис.2.5) екземпляр елемента даних, від якого направлено зв'язок, ідентифікує один і лише один екземпляр елемента даних, до якого направлено зв'язок. Ця ідентифікація є унікальною й визначає функціональну залежність.

В складній асоціації типу М (рис.2.6) екземпляр елемента даних, від якого направлено зв'язок, ідентифікує деяке число екземплярів елементів даних, до яких направлено зв'язок. Ідентифікація є багатозначною залежністю і не обов'язково унікальною. При цьому зв'язок в зворотному напрямку не розглядається.

В умовній асоціації типу С (рис.2.7) для даного екземпляра елемента даних, від якого спрямований зв'язок, може не існувати відповідного екземпляра елемента даних, до якого спрямований зв'язок. Якщо він існує, то відноситься до єдиного екземпляра елемента даних. Наприклад, ПІБ РОБІТНИКА і ДАТА ЗВІЛЬНЕННЯ.

У реальних базах даних існує велика кількість типів елементів даних.

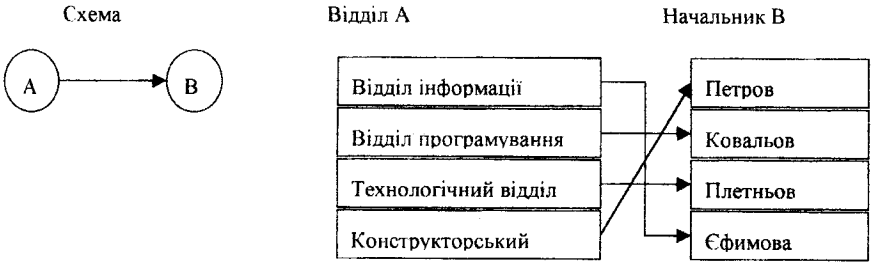


Рисунок 2.5 – Приклад простої асоціації типу 1

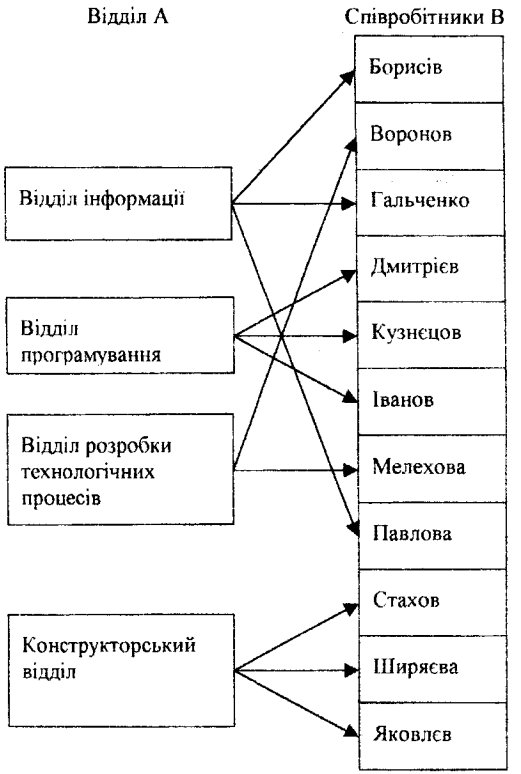


Рисунок 2.6 - Приклад складної асоціації типу М

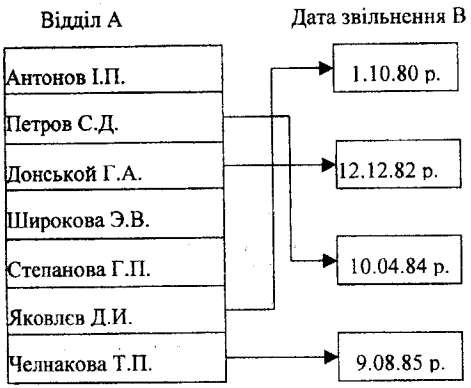


Рисунок 2.7 – Приклад умовної асоціації типу С

Для зменшення кількості зв'язків елементи об'єднують у групи. Таке групування значно зменшує кількість записів. Об'єднання елементів у групи повинно бути аргументованим і продуманим.

Для підвищення ілюстративності аналізованих зв'язків застосовується мова інфологічного моделювання (МІМ), у якій сутності й асоціації подають пропозиціями виду:

СУТНІСТЬ (атрибут 1, атрибут 2 , ... , атрибут n)

АСОЦІАЦІЯ[СУТНІСТЬ S1, СУТНІСТЬ S2, ...](атрибут 1, атрибут2, ... , атрибут n)

де *S* - ступінь зв'язку, а атрибути, що входять у ключ, повинні бути відмічені підкресленням.

Для виявлення зв'язків між сутностями необхідно, як мінімум, визначити самі сутності. Але це не проста задача, тому що в різних предметних областях один і той же об'єкт може бути сутністю, атрибутом або асоціацією.

Контрольні питання

1. Що Ви розумієте під схемою бази даних?
2. Що таке підсхема бази даних?

3. Перелічіть основні типи зв'язків між елементами даних.
4. Які види зв'язків між сутностями Вам відомі?
5. Визначте основні типи асоціацій.

2.3 Класифікація сутностей

К.Дейт визначає три основні класи сутностей: стрижневі, асоціативні і характеристичні, а також підклас асоціативних сутностей - позначення.

Стрижнева сутність (стрижень) - це незалежна сутність.

Асоціативна сутність (*асоціація*) - це зв'язок типу Б:Б між двома або більше сутностями або екземплярами сутності. Асоціації розглядаються як повноправні сутності:

- вони можуть брати участь у других асоціаціях і позначеннях так само, як стрижневі сутності;
- можуть мати властивості, тобто мати не тільки набір ключових атрибутів, необхідних для вказівки зв'язків, але і будь-яке число інших атрибутів, що характеризують зв'язок.

Характеристична сутність (*характеристика*) - це зв'язок типу Б:1 або 1:1 між двома сутностями (окремий випадок асоціації). Єдиною метою характеристики в рамках аналізованої предметної області є опис або уточнення деякої іншої сутності. Необхідність у них виникає в зв'язку з тим, що сутності реального світу мають іноді багатозначні властивості. Наприклад, книга може мати декілька характеристик перевидання (доповнене, перероблене, ...) і т.д. Існування характеристики цілком залежить від сутності, що характеризується. Для опису характеристики використовується нова пропозиція MIM, що має в загальному випадку вигляд:

ХАРАКТЕРИСТИКА (атрибут 1, атрибут 2, ...) {СПИСОК СУТНОСТЕЙ, ЩО ХАРАКТЕРИЗУЮТЬСЯ }.

Позначення - це зв'язок типу Б:1 або 1:1 між двома сутностями і відрізняється від характеристики тим, що не залежить від сутності, яку він позначає.

Розглянемо приклад, пов'язаний із зарахуванням співробітників у різні відділи організації. При відсутності жорстких правил (співробітник може одночасно зараховуватися в декілька відділів або не зараховуватися ні в один відділ) необхідно створити опис з асоціацією ЗАРАХУВАННЯ:

Відділи (Номер відділу, Назва відділу, ...)

Службовці (Табельний номер, Прізвище, ...)

Зарахування [Відділи M, Службовці N](Номер відділу, Табельний номер, Дата зарахування).

Проте, за умови, що кожний із співробітників повинний бути обов'язково зарахований в один із відділів, можна створити опис із позначенням СЛУЖБОВЦІ:

Відділи (Номер відділу, Назва відділу, ...)

Службовці (Табельний номер, Прізвище, ... , Номер відділу, Дата зарахування) [Відділи]

У даному прикладі службовці мають незалежне існування (якщо ліквідується відділ, то з цього не випливає, що також повинні бути скорочені службовці такого відділу). Тому вони не можуть бути характеристиками відділів і названі позначеннями. Позначення використовують для збереження повторюваних значень великих текстових атрибутів: "кодифікатори" досліджуваних студентами дисциплін, найменувань організацій і їхніх відділів, переліків товарів і т.п. Опис позначення зовнішньо відрізняється від опису характеристики тільки тим, що сутності, які позначаються, беруться не у фігурні дужки, а в квадратні: *ПОЗНАЧЕННЯ (атрибут 1, атрибут 2, ...)[СПИСОК СУТНОСТЕЙ, ЩО ПОЗНАЧАЮТЬСЯ].*

Як правило, позначення не розглядаються як повноправні сутності, хоча це не призвело б до помилки.

Позначення і характеристики не є цілком незалежними сутностями, оскільки вони припускають наявність деякої іншої сутності, що буде "позначатися" або "характеризуватися". Проте вони все ж являють собою окремі випадки сутності і можуть, звичайно, мати властивості, можуть брати участь в асоціаціях, позначеннях і мати свої власні (більш низького рівня) характеристики. Підкреслимо також, що всі екземпляри характеристики повинні бути обов'язково пов'язані з яким-небудь екземпляром сутності, що характеризується. Проте припускається, що деякі екземпляри сутності, що характеризуються, не мали зв'язків. Перевизначимо тепер стрижневу сутність як сутність, що не є ні асоціацією, ні позначенням, ні характеристикою. Такі сутності мають незалежне існування, хоча вони і можуть позначати інші сутності, як, наприклад, співробітники позначають відділи.

Контрольні питання

1. Дайте визначення стрижневої сутності.
2. Що таке асоціативна сутність?
3. Що Ви розумієте під характеристичною сутністю?
4. Як використовують МІМ для опису характеристичної сутності?
5. Дайте визначення позначення.

2.4 Аналіз предметної області

Першим етапом проектування бази даних будь-якого типу є аналіз предметної області, що закінчується побудовою інформаційної структури (концептуальної схеми). На даному етапі аналізуються запити користувачів, вибираються інформаційні об'єкти та їх характеристики і на основі проведеного аналізу формується структура предметної області, яка не залежить від програмного та технічного середовища, в якому буде реа-

лізовуватися база даних. Аналіз предметної області доцільно розбити на три фази:

- аналіз концептуальних вимог та інформаційних потреб;
- виявлення інформаційних об'єктів та зв'язків між ними;
- побудова концептуальної моделі предметної області та проектування концептуальної схеми бази даних.

На етапі аналізу концептуальних вимог та інформаційних потреб необхідно вирішити такі задачі:

- аналіз вимог користувача до бази даних (концептуальних вимог);
- виявлення задач, що мають місце, при обробці інформації, яка повинна бути представлена у базі даних (аналіз додатків);
- виявлення перспективних задач (перспективних додатків);
- документування результатів аналізу.

Вимогами користувачів до бази даних, що розробляється є, в загальному випадку, список запитів з вказанням їх інтенсивності та об'ємів даних. Ці вказівки опрацьовуються в діалозі з майбутнім користувачем бази даних. Тут же з'ясовуються вимоги до вводу, відновлення та корегування інформації. Вимоги користувачів уточнюються та доповнюються при аналізі перспективних додатків, що мають місце.

Так, виходячи із специфіки діяльності читального залу, необхідно забезпечити облік книг, що є в наявності, виконувати швидкий пошук творів, що входять до складу тих чи інших книг. Крім того читальному залу потрібна картотека користувачів, що дозволяло б оперативно здійснювати з ними зв'язок, а також для виявлення користувачів, які порушили строки повернення книги, повинна бути картотека творів, що є в наявності в читальному залі. Тоді в базу даних доцільно включити інформацію про: книги, що є в читальному залі; твори, що входять до складу тих чи інших книг; користувачів, що користуються послугами

читального залу. При цьому, розроблювана база даних повинна забезпечити такі функції:

1.Ведення картотеки користувачів.

В інформацію про користувачів доцільно включити такі дані:

- прізвище, ім'я та по батькові;
- адреса;
- номер телефону;
- паспортні дані;
- освіта;
- професія.

2. Облік книг, що є в читальному залі чи якими в даний момент користуються повинен враховувати такі дані:

- назва книги;
- автор книги;
- рік видання;
- видавництво;
- кількість сторінок;
- предметна область.

3. Ведення картотеки творів, які містяться в даній книзі, включає таку інформацію:

- назва;
- автор;
- дата написання.

4. Виявлення боржників, що не повернули книгу протягом дня.

5. Формування даних про повернення книги.

Контрольні питання

1. Охарактеризуйте аналіз предметної області як перший етап проектування бази даних.
2. Назвіть три фази аналізу предметної області.

3. Перелічіть задачі етапу аналізу концептуальних вимог та інформаційних потреб.
4. Назвіть задачі етапу виявлення інформаційних об'єктів та зв'язків між ними.

2.5 Розробка універсального відношення

Друга фаза аналізу предметної області складається з вибору інформаційних об'єктів, задання необхідних властивостей для кожного об'єкта, виявлення зв'язків між об'єктами, виявлення обмежень, що накладаються на інформаційні об'єкти, типи зв'язків між ними, характеристики інформаційних об'єктів.

При виборі інформаційних об'єктів бажано намагатися відповісти на такі питання:

1. На які класи можна розбити дані, що підлягають збереженню у базі даних?
2. Яке ім'я можна присвоїти кожному класу даних?
3. Які найбільш цікаві характеристики (з точки зору користувача) кожного класу даних можно виділити?
4. Які імена можна присвоїти вибраним наборам характеристик?

Виділення інформаційних об'єктів - процес ітеративний. Він здійснюється на основі аналізу інформаційних потоків та інтерв'ювання споживачів. Характеристики інформаційних об'єктів визначаються тими ж методами.

Введемо ряд позначень, які будуть використовуватися у ході подальшого викладення матеріалу.

R - є відношення над множинами *D1, D2,...Dn*, якщо воно є множиною упорядкованих *n*-кортежів вигляду *<d1,d2,...dn>*. *D1,D2,...Dn* називаються доменами відношення *R*.

Відношення може бути подане у вигляді файла або таблиці, стовпцями яких є елементи доменів, а рядками - кортежі. Кожен кортеж відображає один екземпляр інформаційного об'єкта. Імена стовпців (поле запису) – називаються атрибутами, а індивідуальні значення елементів - значеннями атрибутів. Кожен атрибут відображає відповідну характеристику інформаційного об'єкта. Число стовпців у відношенні називається ступенем відношення, а число кортежів - потужністю відношення. У процесі експлуатації бази даних ступінь відношення змінюється значно рідше, ніж його потужність.

Атрибут, або набір атрибутів, який можна використати для однозначності ідентифікації конкретного кортежа, називається первинним ключем (у випадку набору атрибутів - складений ключ).

Можливі випадки, коли відношення може вміщувати декілька унікальних ключів. Тоді один з них вибирається в якості первинного, а інші отримують назву можливих ключів.

Атрибути, що представляють копії ключів інших відношень, називаються зовнішніми ключами.

Атрибут, або набір атрибутів, що використовується для більш швидкого пошуку називається другорядним індексом.

Універсальним називається відношення, що вміщує в себе всі атрибути, які будуть використовуватися в базі даних. Для невеликих баз даних універсальне відношення може служити відправною точкою при їх проектуванні.

Розглянемо порядок роботи універсального віднощення при створенні бази даних для читального залу.

Враховуючи аналіз предметної області, в універсальне відношення потрібно включити атрибути, що описують такі інформаційні об'єкти: *КНИГА, ТВІР, КОРИСТУВАЧ, РОЗДІЛ.*

Складемо перелік найбільш суттєвих характеристик кожного інформаційного об`єкта.

КНИГА (шифр книги, рік видання; видавництво, що надрукувало книгу; кількість сторінок, що включає книга).

ТВІР (назва твору, що входить до складу книги; автор твору; дата створення).

КОРИСТУВАЧ (код користувача, що служить для його швидкої ідентифікації; прізвище, ім'я та по батькові користувача; домашня адреса; номер телефону, домашнього або службового; паспортні дані; освіта; професія).

РОЗДІЛ (код предметної області, назва предметної області).

Перерахунок вибраних для універсального відношення атрибутів приведено в таблиці 2.1.

Оскільки всі перераховані в таблиці атрибути є незалежними, тобто значення одних з них не можуть бути обчислені за значеннями інших, то всі вони можуть бути включені в склад універсального відношення, яке при цьому приймає вигляд:

R (Cod_book , Name, Author, Cod_rozdil, Name_rozdil, Data_write, Publisher, Year_publish, Page, Cod_client, FIO, Address, Telephone, № pasp, Culture, Profession).

Контрольні питання

1. Визначте принципи вибору інформаційних об'єктів.
2. Дайте визначення понять: домен, елемент домену, кортеж відношення.
3. У чому полягає різниця між ступенем відношення та його потужністю?
4. Що таке універсальне відношення?
5. Наведіть приклад роботи універсального відношення при створенні бази даних.

2.6 Розробка *ER*–моделі предметної області

Заключна фаза аналізу предметної області складається з розробки її інформаційної структури (або концептуальної схеми).

Таблиця 2.1 – **Початковий перелік атрибутів для формування універсального відношення бази даних читального залу**

Шифр книги	Cod_book	Кожна книга має унікальний шифр
Код розділу	Cod_rozdil	Кожен предметний розділ має унікальний шифр
Назва розділу	Name_rozdil	Назва предметного розділу
Назва твору	Name	Кожен твір має назву
Автор	Author	Письменник, що написав твір
Дата створення	Data_write	Дата, коли письменник закінчив писати твір
Дата видання	Year_publish	Дата, коли книгу надрукували
Видавництво	Publisher	Видавництво, що надрукувало книгу
Кількість сторінок	Page	Кількість сторінок, що займає книга
Код користувача	Cod_client	Код користувача книги
Користувач	FIO	Прізвище, ім'я та по батькові
Адреса	Address	Адреса користувача
Телефон	Telephone	Номер телефона користувача
Паспорт	№_pasp	Паспортні дані користувача
Освіта	Culture	Освіта, яку має користувач
Професія	Profession	Професія користувача

В звичайних випадках для побудови концептуальної схеми використовуються традиційні методи агрегації та узагальнення. При агрегації декілька інформаційних об'єктів (елементів даних) об'єднується в один у відповідності з семантичними зв'язками між об'єктами. При побудові інфологічних моделей можна використовувати мову *ER*-діаграм (від англ. Entity-Relationship, "сутність-зв'язок"). В них сутності зображуються позначеними прямокутниками, асоціації - позначеними ромбами або шестикутниками, атрибути - позначеними овалами, а зв'язки між ними - ненаправленими ребрами, над якими може проставлятися ступінь зв'язку (1 або буква, що заміняє слово "БАГАТО") і необхідне пояснення.

Розглянемо деякі риси моделі "сутність-зв'язок" (*ER*-моделі). На використанні різновидів *ER*-моделі реалізується більшість сучасних підходів до проектування баз даних. Модель була запропонована Ченом (Chen) у 1976 р. Моделювання предметної області базується на використанні графічних діаграм, що включають невелике число різнорідних компонентів. У зв'язку з наочністю уявлення концептуальних схем баз даних *ER*-моделі знайшли широке застосування в системах CASE, що підтримують автоматизоване проектування реляційних баз даних.

Головними поняттями *ER*-моделі є сутність, зв'язок і атрибут. У діаграмах *ER*-моделі сутність подається у вигляді прямокутника, що містить ім'я сутності. При цьому ім'я сутності - це ім'я типу, а не деякого конкретного екземпляра цього типу. Для кращого розуміння ім'я сутності може супроводжуватися прикладами конкретних об'єктів цього типу. Сутність АЕРОПОРТ із об'єктами Шереметьєво і Хітроу зображена на рис.2.8.

Кожний екземпляр сутності повинен відрізнятися від будь-якого іншого екземпляра тієї ж сутності (ця вимога до певної міри аналогічна вимозі відсутності кортежів-дублікатів у реляційних таблицях).

Зв'язок - це асоціація, що графічно зображується та установлюється між двома сутностями. Ця асоціація завжди є бінарною і може існувати

Рисунок 2.8 – Зображення сутності АЕРОПОРТ із об'єктами Шереметьєво і Хітроу

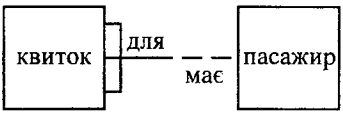


Рисунок 2.9 - Зображення зв'язку між сутностями КВИТОК та ПАСАЖИР

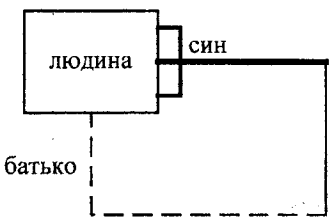


Рисунок 2.10 - Зображення рекурсивного зв'язку

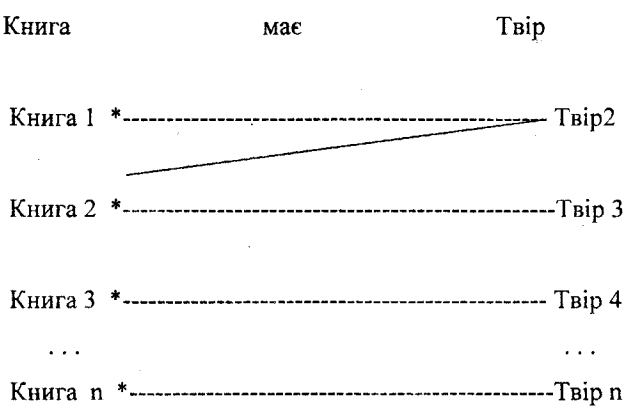


Рисунок 2.11 – Зображення зв'язку КНИГА – ТВІР

між двома різними сутностями або між сутністю і нею ж самою (рекурсивний зв'язок). У будь-якому зв'язку виділяються два кінці (відповідно до існуючої пари сутностей, що зв'язуються), на кожному з яких вказується ім'я кінця зв'язку, ступінь кінця зв'язку (скільки екземплярів даної сутності зв'язується), обов'язковість зв'язку (тобто чи будь-який екземпляр даної сутності повинен брати участь у даному зв'язку). Зв'язок подається у вигляді лінії, що зв'язує дві сутності або веде від сутності до неї ж самої. Про це в місці "стикування" зв'язку із сутністю вказує триточковий вхід у прямокутник сутності, якщо для цієї сутності в зв'язку можуть використовуватися багато (many) екземплярів сутності, і одноточковий вхід, якщо в зв'язку може брати участь тільки один екземпляр сутності. Обов'язковий кінець зв'язку зображується суцільною лінією, а необов'язковий - переривчастою лінією.

Розглянемо приклад зв'язку між сутностями КВИТОК і ПАСАЖИР (рис.2.9). При цьому кінець сутності з ім'ям "для" дозволяє зв'язувати з одним пасажиром більше одного квитка, причому кожний квиток повинен бути пов'язаний із яким-небудь пасажиром. Кінець сутності з ім'ям "має" означає, що кожний квиток може належати тільки одному пасажиру, причому пасажир зобов'язаний мати не менше одного квитка. Трактування зображеної діаграми (рис.2.9) таке:

- Кожний КВИТОК призначений для одного і тільки одного ПАСАЖИРА;
- Кожний ПАСАЖИР може мати один або більше КВИТКІВ.

На рис.2.10 приведено приклад зображення рекурсивного зв'язку, що зв'язує сутність ЛЮДИНА з нею ж самою.

Кінець зв'язку з ім'ям СИН визначає той факт, що в одного батька може бути більше ніж один син. Кінець зв'язку з ім'ям БАТЬКО означає, що не в кожної людини можуть бути СИНИ.

Трактування зображеної діаграми (рис.2.10) таке:

- Кожна ЛЮДИНА є сином одного і тільки одного ЧОЛОВІКА;

- Кожна ЛЮДИНА може бути батьком для одного або більше ЛЮДЕЙ.

Атрибутом сутності є будь-яка деталь, що служить для уточнення, ідентифікації, класифікації, числової характеристики або вираження стану сутності. Імена атрибутів заносяться в прямокутник, що зображує сутність, під ім'ям сутності і зображуються малими буквами, можливо, із прикладами. Наприклад, унікальним ідентифікатором сутності є атрибут, комбінація атрибутів, комбінація зв'язків або комбінація зв'язків і атрибутів, що унікально відрізняє будь-який екземпляр сутності від інших екземплярів сутності того ж типу.

Таким чином, зв'язки можуть представлятися різними способами, з яких ми будем використовувати тільки два: діаграма *ER*-екземплярів і діаграма *ER*-типів. Діаграми *ER*-екземплярів відображають зв'язки між екземплярами сутностей. Діаграми *ER*-типів відображають зв'язки між типами сутностей.

Отже, першою задачою, яку необхідно розв'язати при розробці *ER*-моделі, є формування сутностей, що необхідні для описання предметної області. Іншими словами, необхідно вказати ті типи об'єктів (тобто набори подібних об'єктів), про які в системі повинна накопичуватися інформація. Це означає, що за підсумками аналізу предметної області потрібно мати повну або достатньо повну уяву про реалізовані в системі запити. Разом з тим необхідно врахувати, що при кваліфікованій експлуатації системи у більшості випадків у користувача виникає бажання розширити систему запитів. У зв'язку з цим, при розробці *ER*-схеми, з одного боку зручно твердо прив'язатися до обраної в результаті аналізу предметної області системи запитів, а з іншого боку, бажано розглядати задачу в більш широкому плані, з врахуванням перспектив подальшого нарощування можливостей системи. Приведемо приклад для таких об'єктів: КНИГА, ТВІР, КОРИСТУВАЧ, РОЗДІЛ. Для кожної сутності необхідно обрати атрибут, що її однозначно ідентифікує або сукупність атрибутів, які

пати ключем відповідного відношення. Необхідно врахувати, що ключ повинен не тільки виконувати задачу ідентифікації, але і по можливості, включати в свій склад мінімальне число атрибутів. У зв'язку з цим в процесі проектування вибрані в якості ключів атрибути можуть переглядатися. Наприклад, виберемо в якості ключових такі атрибути:

КНИГА --> Cod_book (шифр книги);

ТВІР --> Cod_book (шифр книги), Name_tvir (назва твору);

РОЗДІЛ --> Cod_rozdil (код предметної області).

Необхідність введення складеного ключа для сутності ТВІР обумовлено тим, що в читальному залі може бути кілька однакових творів, написаних одним автором, але входять вони до складу різних книг. Тепер виявимо залежності, що існують між визначеними нами сутностями, а також визначимо характеристики кожного зв'язку. Зв'язок КНИГА – ТВІР приведено на рис.2.11.

З рис.2.11 видно, що клас належності обох сутностей є обов'язковим, оскільки книги та твори, яких немає в читальному залі не повинні заноситись в базу даних.

Тип зв'язку – Б:Б: декілька творів можуть входити до складу однієї книги, та один і той самий твір може знаходитись в декількох книгах. Зв'язок КНИГА – РОЗДІЛ приведено на рис.2.12.

З рис.2.12 видно, що клас належності обох сутностей є обов'язковим, оскільки книги, яких немає в читальному залі не повинні заноситись в базу даних, та розділи з предметних областей, що не мають книг, не заносять до бази даних.

Тип зв'язку – Б:1: декілька творів можуть входити до складу одного предметного розділу, але одна книга не може міститися в більше ніж одному розділі.

У реальному житті часто зустрічаються ситуації, які неможливо описати бінарними зв'язками. Тобто, в цих випадках зв'язок об'єднує одночасно не дві, а більше сутностей, як правило - три. *ER*– діаграма, що ілюструє

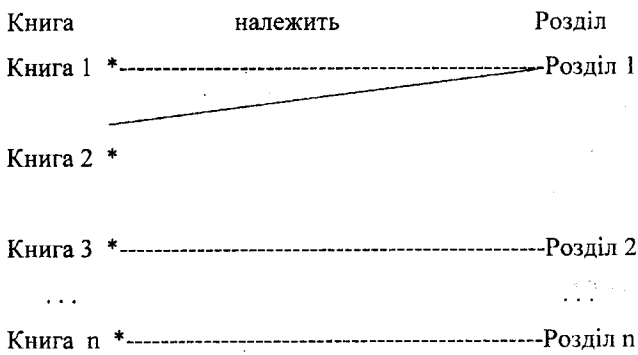


Рисунок 2.12 – Зображення зв'язку КНИГА – РОЗДІЛ

тристоронні зв'язки РОЗДІЛ–КНИГА–ТВІР приведена на рис.2.13). Розмірковуючи аналогічним чином побудуємо *ER*-модель предметної області "Читальний зал". В якості сутностей оберемо такі об'єкти предметної області (з перерахуванням ключових атрибутів кожної сутності) :

КНИГА →<Cod_book(шифр книги)>,Cod_client,Publisher,Year_publish, Page.

ТВІР → <Cod_book (шифр книги), Name_tvir (назва твору)>, Author, Data_write;

РОЗДІЛ → Cod_rozdil (код предметної області), Name_rozdil.

КОРИСТУВАЧ → Cod_client (шифр користувача), FIO, Address, Telephone, №_pasp, Culture, Profession.

Характеристики зв'язків виділених сутностей приведені в табл.2.2, а *ER*-модель предметної області "Читальний зал", що побудована на їх основі, показана на рис.2.14.

До числа більш складних елементів моделі відносяться такі:

Таблиця 2.2 – **Характеристики зв'язків предметної області "Читальний зал"**

Ім'я сутності 1	Ім'я сутності 2	Ім'я зв'язку	Тип зв'язку	Клас належності
Книга	Розділ	Належить	N : 1	Обов'язк., Обов'язк.
Книга	Твір	Має	N : M	Обов'язк., Обов'язк.
Книга	Користувач	Читає	N : 1	Необов'язк., Необов'язк.

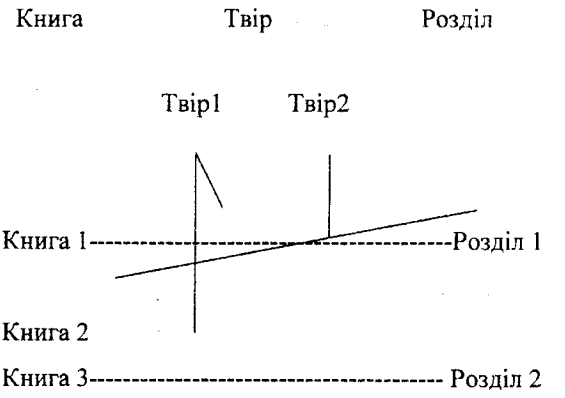


Рисунок 2.13 – *ER*–діаграма, що ілюструє тристоронні зв'язки (РОЗДІЛ – КНИГА – ТВІР)

- Підтипи і супертипи сутностей. Як у мовах програмування з розвинутими типовими системами (наприклад, у мовах об'єктно-орієнтованого програмування), вводиться можливість спадкування типу сутності, виходячи з одного або декількох супертипів.

- Зв'язки "many-to-many". Іноді буває необхідно зв'язувати сутності таким чином, що з обох кінців зв'язку можуть бути присутніми декілька екземплярів сутності (наприклад, усі члени кооперативу спільно володіють майном кооперативу). Для цього вводиться різновид зв'язку БАГАТО-З-БАГАТЬМА.
- Ступені зв'язку, що уточнюються. Іноді буває корисно визначити можливу кількість екземплярів сутності, що беруть участь у даному зв'язку (наприклад, службовцю дозволяється брати участь не більш, ніж у трьох проектах одночасно). Для вираження цього семантичного обмеження дозволяється вказувати на кінці зв'язку її максимальний або обов'язковий ступінь.
- Каскадні видалення екземплярів сутностей. Деякі зв'язки бувають настільки сильними (звичайно, у випадку зв'язку "ОДИН-ДО-БАГАТЬОХ"), що при видаленні опорного екземпляра сутності (відповідає кінцю зв'язку "ОДИН") потрібно видалити і всі екземпляри сутності, що відповідають кінцю зв'язку "БАГАТО". Відповідну вимогу "каскадного видалення" можна сформулювати при визначенні сутності.
- Домени. Як і у випадку реляційної моделі даних буває корисна можливість визначення потенційно припустимої множини значень атрибута сутності (домена).

Ці й інші більш складні елементи моделі даних "сутність-зв'язок" роблять її більш потужною, але одночасно в певній мірі ускладнюють її використання. Звичайно, при реальному використанні *ER*-діаграм для проектування баз даних необхідно ознайомитися з усіма можливостями.

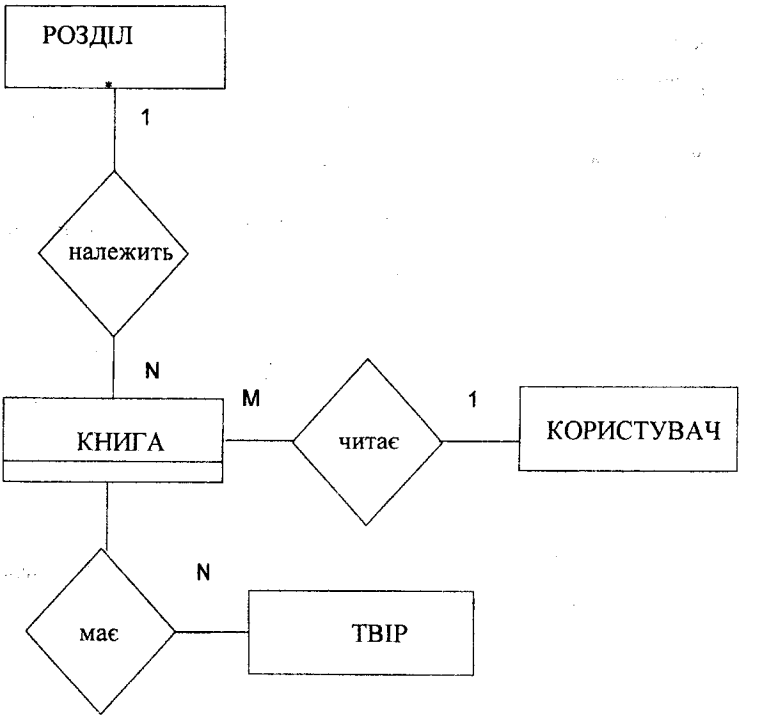


Рисунок 2.14 - *ER*-модель предметної області "Читальний зал"

Контрольні питання

1. Як використовують мову *ER*-діаграм при побудові інфологічних моделей?
2. Як використовують *ER*-діаграми при моделюванні предметної області?
3. Визначте головні поняття *ER*-моделі (сутність, зв'язок, атрибут).
4. Наведіть приклад рекурсивного зв'язку сутностей.
5. Як Ви розумієте поняття "атрибут сутності"?
6. Назвіть основні етапи розробки *ER*-моделі.
7. Які Вам відомі складні елементи моделі?

3 МОДЕЛІ ДАНИХ

Відомі три основних типи моделей даних: ієрархічна, мережна та реляційна. Перші дві з них використовують графові моделі для представлення інформації про об'єкти.

3.1 Ієрархічна модель даних

Деревоподібна (ієрархічна) структура (рис.3.1), або дерево - це зв'язний неорієнтований граф, що не містить циклів, тобто петель з замкнутих шляхів.

Як правило, при роботі з деревом виділяють будь-яку конкретну верхівку (початок) та визначають її як *коріння* дерева. В цю верхівку не заходить жодне ребро. В цьому випадку дерево стає орієнтованим. Орієнтація на кореневому дереві визначається або від коріння, або до коріння.

Кореневе дерево можна визначити таким чином:

1) є єдиний особливий вузол, який називається корінням, в який не заходить жодне ребро;
2) в усі інші вузли заходить тільки одне ребро, а виходить довільна (0, 1, 2,..., *n)* кількість ребер;
3) не існує циклів.

В програмуванні використовується інше визначення дерева, яке дозволяє розглядати дерево як рекурсивну структуру.

Рекурсивне дерево визначається як кінцева множина T, яка складається з одного або більше вузлів таких, що:

1) існує один спеціально виділений вузол, який називається корінням дерева;
2) інші вузли розбиті на m>0 підмножин $T_1, T_2, \ldots, T_m$, що не перетинаються, кожна з яких в свою чергу є деревом. $T_1, T_2, \ldots, T_m$

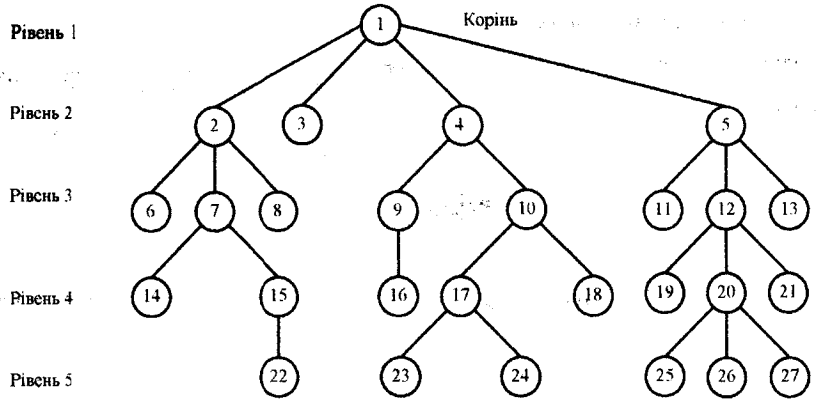
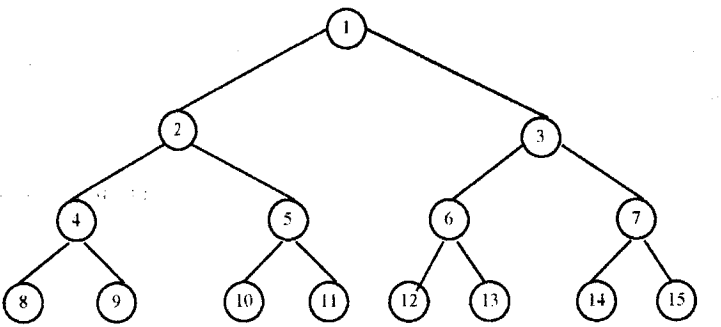
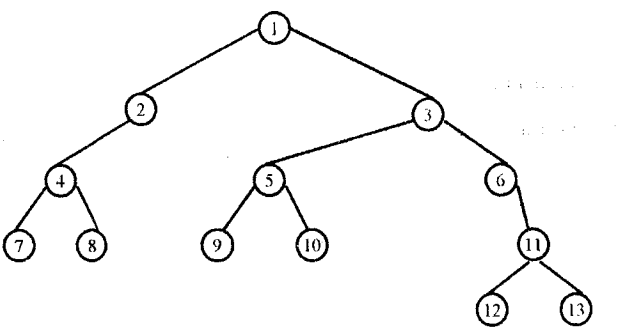


Рисунок 3.1 – Ієрархічна структура даних



а)



б)

Рисунок 3.2 – Приклади дерев

називаються піддеревами.

З визначення випливає, що будь-який вузол дерева є корінням деякого піддерева, що міститься в повному дереві. Число піддерев вузла називають ступенем вузла. Вузол називається кінцевим, якщо він має нульову ступінь. Інколи кінцеві вузли називають листками, а ребра - гілками. Кожний вузол, крім кореневого, зв'язаний з одним вузлом на більш високому рівні ієрархії і називається вихідним. Кожний вузол може бути зв'язаний з одним або декількома вузлами на більш низькому рівні і називається породженим.

Якщо кожний вузол має однакову кількість гілок, причому процес включення нових гілок іде зверху вниз, а на кожному рівні дерева - зліва направо, то таке дерево називається збалансованим (рис. 3.2,а). Для збалансованих дерев фізична організація даних суттєво спрощується. До особливої категорії дерев відносять двійкове (бінарне) дерево. Це дерево має не більше як дві гілки, які виходять з одного вузла. Двійкові дерева можуть бути як збалансованими, так і незбалансованими (рис. 3.2,б).

Прикладом простого ієрархічного подання може служити адміністративна структура вищого учбового закладу (рис.3.3): університет - відділення – факультет – група (студентська).

Пошук даних у ієрархічній структурі виконується завжди по одній із гілок, починаючи з кореневого елемента, тобто необхідно зазначити повний шлях руху по гілках. Так, для пошуку і вибірки одного або декількох екземплярів запису типу СТУДЕНТ (див. рис.3.4) необхідно вказати кореневий елемент ФАКУЛЬТЕТ і елементи КУРС, ГРУПА. В операційній системі MS-DOS для пошуку файла використовується такий же принцип - вказуються послідовно ім'я диска, ім'я каталогу, ім'я підкаталогів, ім'я файла.

На рис.3.5 приведений приклад типу набору, представленого у вигляді діаграми Бахмана. Діаграму назвали за іменем вченого, який вперше їх застосував для опису відношень між даними при розробці

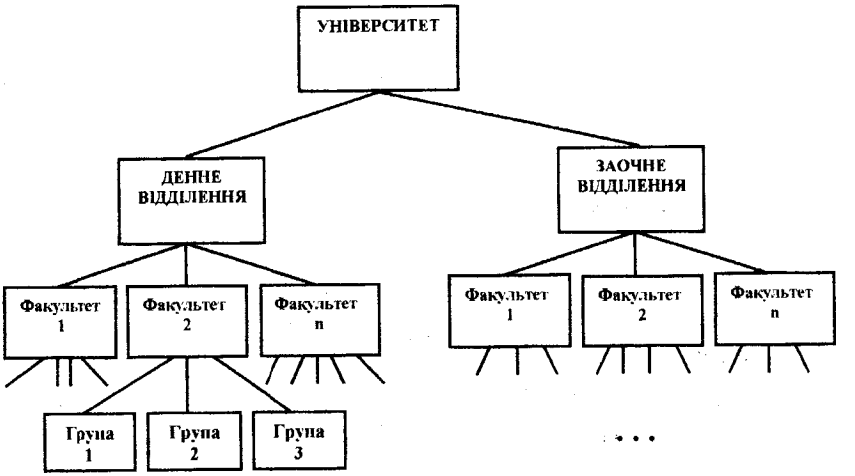


Рисунок 3.3 – Ієрархічне подання адміністративної структури вищого учбового закладу

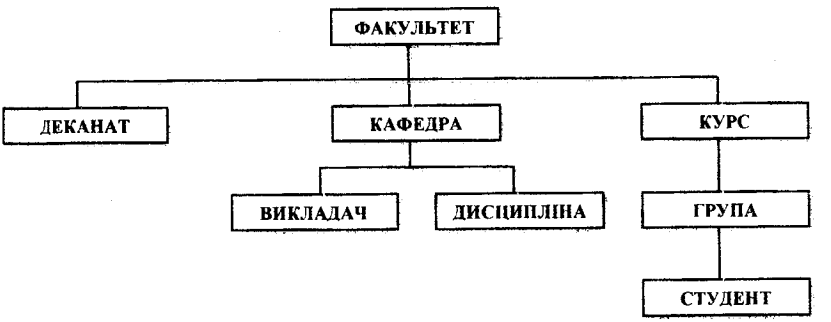


Рисунок 3.4 – Приклад ієрархічної структури

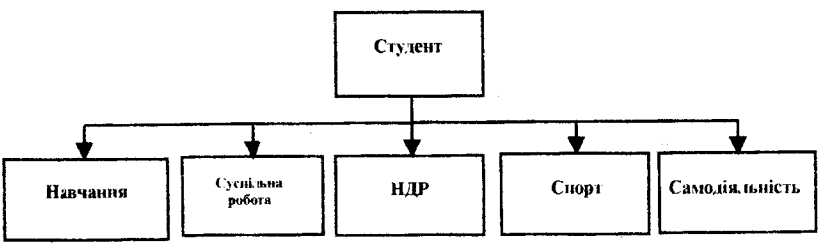


Рисунок 3.5 – Приклад типу набору у вигляді діаграми Бахмана

СКБД IDS.

На такій діаграмі кожний прямокутник представляє собою тип запису, а стрілка - відношення «один до багатьох» між типами запису. У прикладі на рис.3.5 тип запису СТУДЕНТ є записом-власником, а типи записів НАВЧАННЯ, СУСПІЛЬНА РОБОТА, НДР, СПОРТ, САМОДІЯЛЬНІСТЬ - записами-членами. Тип набору названий ім'ям СНСНСС по перших літерах імен усіх типів запису, що беруть участь у наборі (наборові можна було надати і будь-яке інше ім'я). У цілому приведений тип набору призначений для того, щоб відобразити зв'язок між загальними даними про студента, що знаходяться в типі запису СТУДЕНТ, і даними, що характеризують різні сторони діяльності студента у вузі. При традиційному підході всі ці данні можна було б помістити в один загальний запис. Оскільки не кожний студент бере участь, наприклад, у спорті або самодіяльності, то прийшлося б вибрати запис з змінною довжиною або запис із фіксованою довжиною, причому в останньому випадку частина пам'яті витрачалася б даремно. Ієрархічна структура усуває труднощі, що виникають при цьому, тому що в будь-якому екземплярі типу набору з записом-власником можна асоціювати стільки записів-членів, скільки необхідно для конкретного екземпляра.

Повна схема бази даних формується в загальному випадку з множини різних типів набору і типів запису.

Ієрархічна деревоподібна структура, що орієнтована від коріння, задовольняє такі умови:

1) ієрархія завжди починається з кореневого вузла;
2) на першому рівні може знаходитися тільки один вузол - кореневий;
3) на нижніх рівнях знаходяться породжені (залежні) вузли;
4) кожний породжений вузол, який знаходиться на рівні *i*, зв'язаний тільки з одним вхідним вузлом, який знаходиться на рівні *(i-1)* ієрархії дерева;

5) кожний вхідний вузол може мати один або декілька породжених вузлів, які називаються подібними;

6) доступ до кожного породженого вузла виконується через відповідний йому вхідний вузол;

7) існує єдиний ієрархічний шлях доступу до будь-якого вузла, починаючи від кореневого вузла дерева.

Прикладами типових операторів маніпулювання ієрархічно-організованими даними можуть бути такі:

- пошук заданого дерева БД;
- перехід від одного дерева до іншого;
- перехід від одного запису до іншого в середині дерева;
- перехід від одного запису до іншого в порядку обходу ієрархії;
- установлення нового запису в зазначену позицію;
- видалення поточного запису.

Перевагами деревовидної моделі є наявність функціональних систем керування базами даних, які підтримують дану модель; простота сприйняття користувачами принципу ієрархії; забезпечення деякого рівня незалежності даних; простота оцінки операційних характеристик системи завдяки апріорно заданим взаємозв'язкам.

До недоліків ієрархічних структур відносять надлишковість зберігання інформації, оскільки ієрархічні структури не підтримують взамозв'язки Б:Б; строгу ієрархічну впорядкованість, яка ускладнює процедури включення та вилучення записів; вилучення вихідних вузлів призводить до вилучення відповідних їм породжених, що вимагає особливої обережності; ускладнюється доступ до даних, які лежать на більш низьких рівнях ієрархії, оскільки кореневий вузол завжди є головним, а доступ до будь-якого породженого вузла може здійснюватись через вихідний.

Розглянемо в якості прикладу задачу. Нехай необхідно побудувати ієрархічну базу даних, до якої входить інформація про викладачів, студентів та дисципліни в таких взаємозв'язках:

О - Б
ВИКЛАДАЧ ⟷ ДИСЦИПЛІНА
О - Б
СТУДЕНТ ⟷ ДИСЦИПЛІНА,

О - Б
де ⟷ - відношення ОДИН – ДО –БАГАТЬОХ.

Ця інформація в ієрархічній моделі може бути представлена різними способами. Один з варіантів показаний на рис.3.6. Кореневим вузлом є об'єкт СТУДЕНТ. Для кожного студента при даному представленні є екземпляр кореневого вузла.

Об'єкти ВИКЛАДАЧ, ДИСЦИПЛІНА об'єднані в породжуваний вузол ДИСЦИПЛІНА + ВИКЛАДАЧ. В кожний момент часу база даних, що задана моделлю рис. 3.6, може мати записи для N студентів. На рис.3.7 показаний екземпляр запису бази даних для студента Собка Н.Ю.

Побудована модель дозволяє виконати такі операції, як оперативна видача інформації про здачу іспитів студентами з різних дисциплін (отримати кількісну та якісну оцінки). В той же час, якщо необхідно отримати інформацію про те, які викладачі з дисципліни "Схемотехніка" приймають іспити, то прийдеться переглянути всі записи породжуваних вузлів всіх екземплярів кореневого вузла. При такій постановці задачі більше підходить модель, яка представлена на рис.3.8, де кореневим вузлом є об'єкт ВИКЛАДАЧ, а об'єкти СТУДЕНТ та ДИСЦИПЛІНА об'єднані в породжуваний вузол СТУДЕНТ+ ДИСЦИПЛІНА. Крім того, можна відмітити, що дані про викладачів та назви дисциплін в базі даних (див. рис.3.6) повторюються в екземплярі породжуваного вузла [illegible]

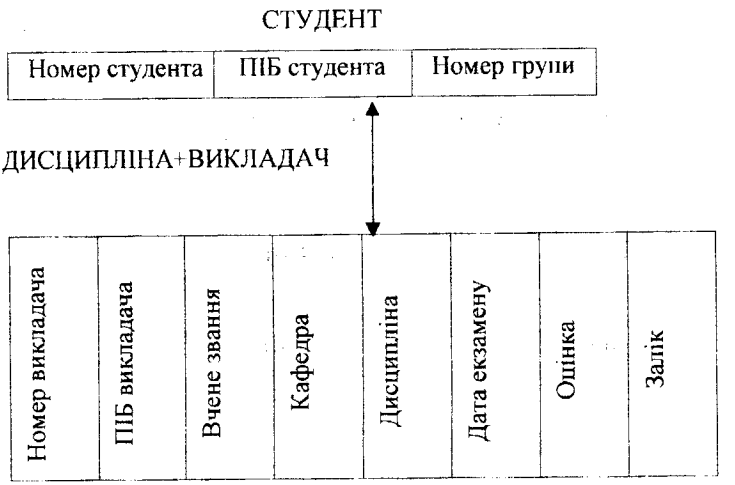


Рисунок 3.6 – Представлення бази даних (СТУДЕНТ-ДИСЦИПЛІНА) за допомогою ієрархічної моделі

1212	Собко Н.Ю.	832

		121	Колос Ю.П.	Доцент	Кафедра філосо-фії	Філосо-фія	23.05.99	Відм.	Залік
	053	Ковко В.І.	Доцент	Кафедра вищої математики	Вища математика	25.05.99	Відм.	Залік	
096	Сухов А.П.	Про-фе-сор	Кафедра обчислювальної техніки	Схемо-техніка	30.05.99	Відм.	Залік		

Рисунок 3.7 – Екземпляр запису ієрархічної бази даних СТУДЕНТ – ДИСЦИПЛІНА

разів, скільки студентів навчається у даного викладача. В моделі на рис.3.8 даними, що дублюються, є відомості про студентів.

На прикладах (див. рис.3.6 та рис.3.8) проілюструємо проблеми, які пов'язані з операціями включення та вилучення даних.

З принципу ієрархії випливає, що екземпляр породжуваного вузла не може існувати при відсутності відповідного йому екземпляра вихідного вузла. Таким чином, неможливо без використання будь-яких інших методів включити в базу даних, представлену на рис.3.6, відомості про викладача, який не приймає іспити або заліки. Одним з методів реалізації такого включення є формування "пустих" екземплярів, які приводять до додаткових затрат зовнішньої пам'яті ЕОМ і до необхідності строгого обліку

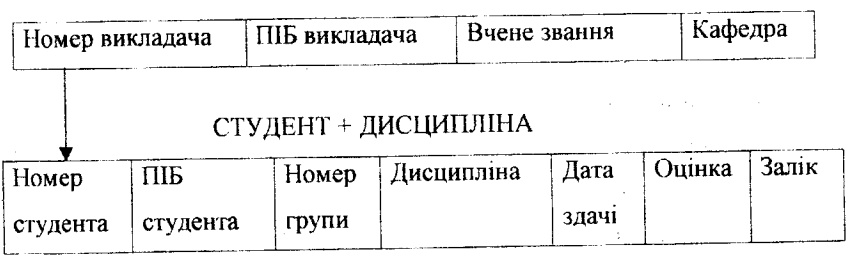


Рисунок 3.8 – Альтернативне представлення бази даних СТУДЕНТ-ДИСЦИПЛІНА

таких записів. При вилученні екземплярів вихідного вузла автоматично вилучаються всі екземпляри породжені вузлом. Наприклад, в базі даних, модель якої приведена на рис.3.8, вилучення екземпляра ВИКЛАДАЧ потягне за собою і вилучення всіх екземплярів про студентів, які навчаються в даного викладача. Вказане повністю неприпустиме при розв'язку, наприклад, задачі оцінки якості навчання студентів.

В ієрархічних базах даних проблеми, пов'язані з операціями включення нових записів і вилучення старих, а також проблеми частково

дублювання інформації виникають в результаті того, що відношення БАГАТО-ДО-БАГАТЬОХ безпосередньо не підтримується, що і є основним недоліком ієрархічних моделей.

Контрольні запитання

1. В чому полягають особливості структурної організації ієрархічних моделей даних?
2. Чим обумовлено широке використання збалансованих дерев?
3. Перерахуйте типові оператори маніпулювання ієрархічними даними.
4. Перерахуйте основні переваги та недоліки ієрархічних моделей даних.

3.2 Мережна модель даних

Більш широкі можливості для користувача забезпечує мережна модель бази даних, яка є узагальненням ієрархічної моделі і дозволяє відображати відношення між типами записів виду «багато до багатьох». У мережній моделі кожний тип запису може бути членом більш ніж одного типу набору. В результаті можна сформувати модель бази даних з довільними зв'язками між різними типами записів. Крім того, окремі типи записів можна не включати ні в які типи набору, що забезпечує додаткові можливості для ряду задач обробки даних в СКБД. Відзначимо, що СКБД, в основі якої використовується мережна модель бази даних, називається СУБД мережного типу. В 1971 році був опублікований офіційний стандарт мережних баз даних, який відомий як модель CODASYL.

В основі мережної моделі даних лежать мережні структури. Розглянемо мережні структури. Припустимо, що нам необхідно графічно представити відношення між об'єктами СТУДЕНТСЬКИЙ_КОЛЕКТИВ та УЧБОВА_ГРУПА, КІМНАТА_В_ ГУРТОЖИТКУ та СТУДЕНТ. Взаємозв'язок між цими об'єктами показаний на рис.3.9, звідки можна побачити, що дана схема не є ієрархічною, оскільки породжений елемент

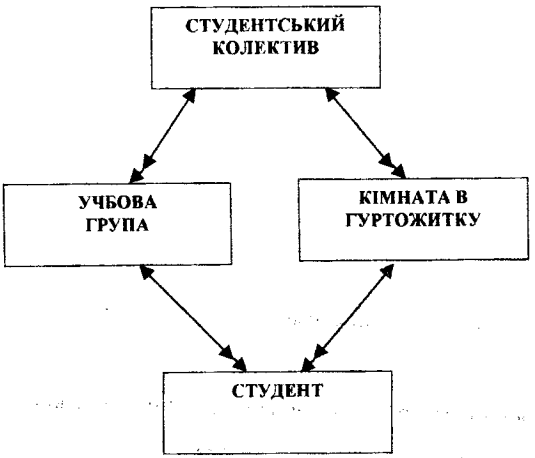


Рисунок 3. 9 – Приклад взаємозв'язків між об'єктами мережної структури

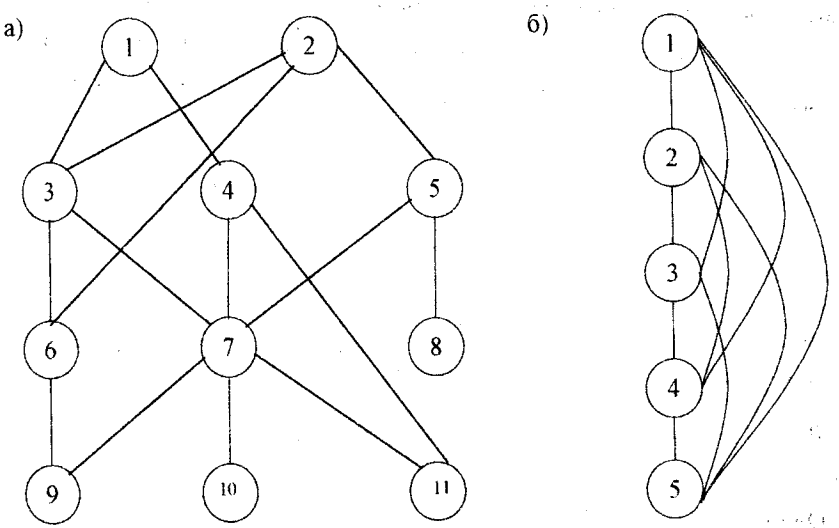


Рисунок 3.10 – Приклади мережних структур

СТУДЕНТ має два вихідних. Мережна структура відрізняється від ієрархічної тим, що в ній будь-який елемент може зв'язуватися з будь-яким іншим елементом.

На рис.3.10 наводяться приклади більш складних мережних структур, де для зручності цифрами 1,2,..., 11 позначені елементи (об'єкти). Мережну структуру можна описати за допомогою вихідних та породжених елементів та зобразити її таким чином, щоб породжені елементи знаходилися нижче вихідних, що і зроблено на рис 3.10. При розгляді деяких мережних структур природно розглядати рівні, як і при деревовидних структурах. Структура на рис.3.9 має три рівні, а мережні структури на рис.3.10 - відповідно чотири та п'ять рівнів.

Розглянемо як подаються в мережній структурі взаємозв'язки між об'єктами. На рисунку 3.9 представлена мережна структура, в якій між двома об'єктами присутні два види взаємозв'язку: ОДИН-ДО-БАГАТЬОХ (наприклад, між об'єктами НАВЧАЛЬНА_ГРУПА - СТУДЕНТ) та БАГАТО-ДО-ОДНОГО (СТУДЕНТ - КІМНАТА_В_ГУРТОЖИТКУ). Цей вид зв'язку закладено в ієрархічних структурах при умові, що дані зв'язки існують відповідно між вихідними та породженими, а зв'язок БАГАТО-ДО-ОДНОГО – між породженими та вихідними вузлами. Виконання цієї умови для відповідних вузлів мережної схеми говорить про просту мережну структуру. Складною мережною структурою називають схему, в якій присутній хоча б один зв'язок БАГАТО-ДО-БАГАТЬОХ.

На рис.3.11 показаний приклад складної мережної структури та можливі взаємозв'язки між об'єктами. В цьому прикладі вузол ВИКЛАДАЧ може мати декілька породжених, тому що викладач веде заняття більш ніж з одним студентом. А кожен студент навчається більш ніж у одного викладача.

Розділення мережних структур на два типи (складні та прості) необхідно хоча б тому, що структури, побудовані з використанням зв'язку БАГАТО-ДО-БАГАТЬОХ, вимагають для їх реалізації використання більш

складних методів. Крім того, деякі системи керування базами даних можуть підтримувати прості мережні структури, але не можуть підтримувати складні. Наприклад, СКБД DMS, DBMS, СЕКТОР дозволяють описувати прості мережні структури. Реалізація складних мережних структур можлива і в цих системах керування базами даних шляхом приведення їх до простішого вигляду.

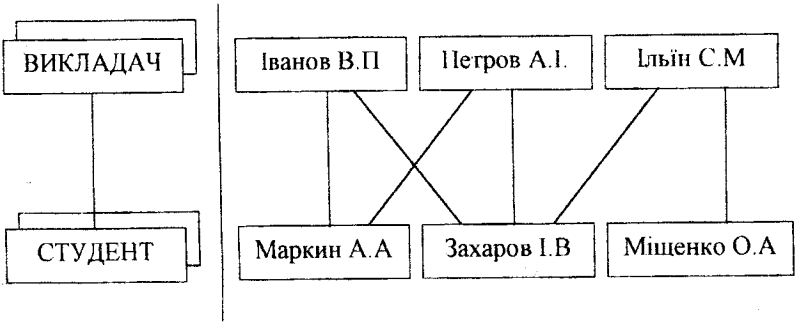


Рисунок 3.11 – Складна мережна структура

Будь-яку мережну структуру можливо привести до простішого вигляду, якщо ввести надлишковість (рис.3.12). Якщо надлишковість, яка при цьому виникає, є допустимою, то такий шлях дозволяє підтримувати мережні структури даних за допомогою СКБД, які орієнтовані на ієрархічну організацію даних.

Прикладами типових операторів маніпулювання мережними даними можуть бути:

- пошук конкретного запису в наборі однотипних записів;
- перехід від предка до першого нащадка за деяким зв'язком;
- перехід до наступного нащадка в деякому зв'язку;
- перехід від нащадка до предка за деяким зв'язком;
- створення нового запису;
- знищення запису;
- модифікація запису;
- включення в зв'язок;

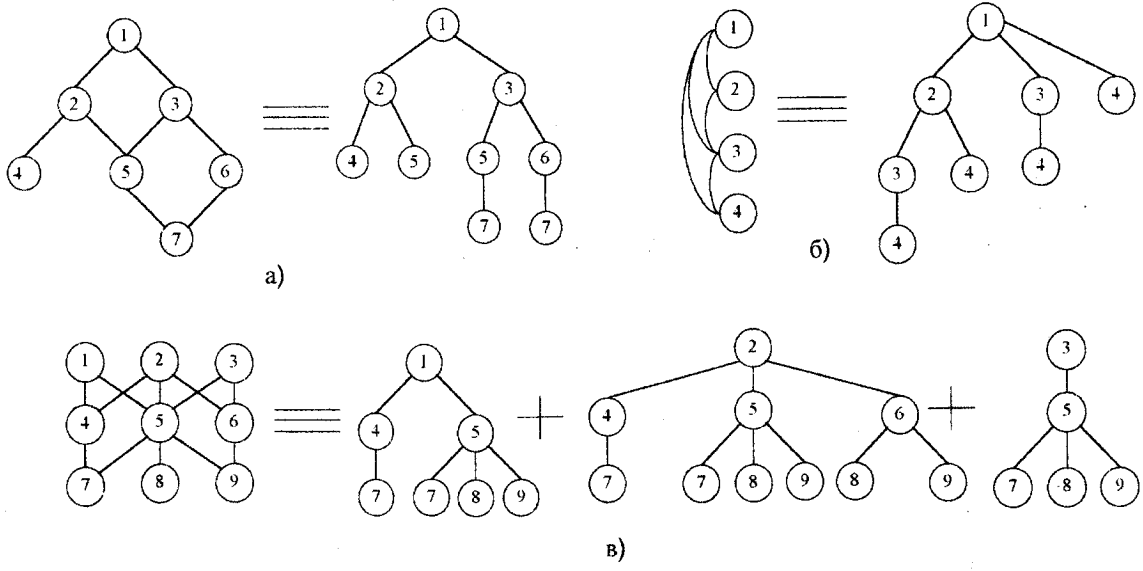


Рисунок 3.12 – Представлення простої мережної структури:
а, б — у вигляді одного дерева;
в — у вигляді суми дерев

- виключення зі зв'язку;
- перестановка в інший зв'язок і т.д.

Бази даних, які описуються мережною моделлю, складаються з декількох областей (рис.3.13). Кожна область складається з записів, які в свою чергу складаються з полів. Об'єднання записів в логічну структуру можливе не тільки по областях, але й за допомогою так званих наборів. Термін набір є основною конструкцією мови системи баз даних КОДАСІЛ. Набір – це пойменоване дворівневе дерево, яке дозволяє формувати багаторівневі дерева та прості мережні структури. Таким чином, база даних КОДАСІЛ складається з деякої кількості наборів. Використовуючи дворівневі зв'язки, спеціаліст з аналізу системи може конструювати достатньо складні структури даних. Набір – це екземпляр пойменованої сукупності записів. Для кожного типу набору тип запису може бути оголошений його власником. Кожний набір повинен мати один екземпляр запису та будь-яку кількість екземплярів кожного типу запису – членів набору. Наприклад, набір можна використовувати для об'єднання записів про студентів одної групи (див. рис. 3.10).

Перерахуємо властивості, що притаманні набору:

а) набір – це пойменована сукупність зв'язаних записів;

б) в кожному екземплярі набору присутній тільки один екземпляр власника;

в) екземпляр набору може мати нуль, один або декілька записів-членів;

г) набір вважається пустим, якщо жодний з екземплярів запису-члена не має зв'язків з відповідним екземпляром запису-володаря;

д) екземпляр набору існує після запам'ятовування запису-володаря;

е) тип набору є логічний взаємозв'язок ОДИН-ДО-БАГАТЬОХ між володарем та членом набору (рис.3.14);

ж) кожному типу набору надається ім'я, що дозволяє одній і тій самій парі типів об'єктів брати участь у декількох взаємозв'язках.

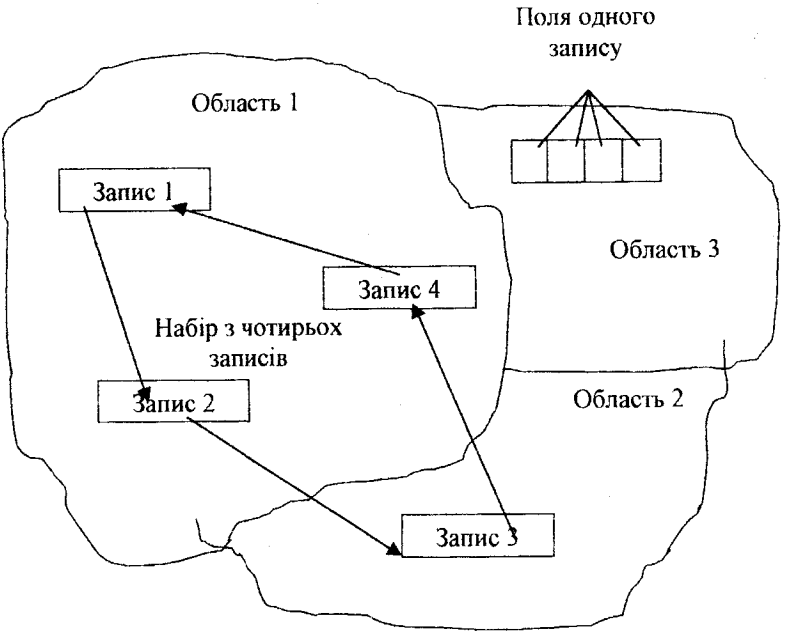


Рисунок 3.13 – Структура мережної бази даних

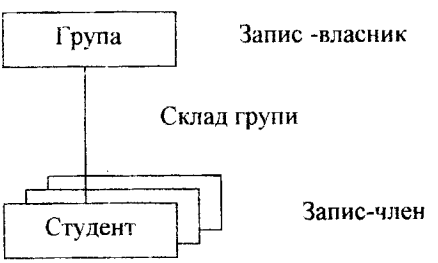


Рисунок 3.14 – Тип набору "Склад групи"

Необхідно розрізняти тип та екземпляр набору. Але спочатку пояснимо різницю між поняттями тип та екземпляр запису. СТУДЕНТ є типом запису, а рядок символів "Захаров Іван Викторович, член профкому, кімн. 638" – екземпляром типу запису СТУДЕНТ. Іншими словами, в базі даних можуть зберігатися один чи декілька екземплярів запису деякого типу. Подібне відношення існує і між типом набору та екземпляру. У прикладі на рис.3.14 наведений тип набору СКЛАД ГРУПИ, а на рис.3.15 – його екземпляр, який вміщує екземпляр типу запису-володаря ГРУПА "832 група, куратор Іванов І.П., каф. кібернетики" та N екземплярів типу запису-члена: "1013 Захаров І.П., староста групи", "201К Кирилова О.Б.", ..., "426Х Харламов О.М."

Таким чином, екземпляр типу набору складається з одного екземпляра типу запису-володаря, нуля чи більше екземплярів типу запису-членів даного типу набору. Між екземпляром типу запису-володаря та екземпляром типу запису-члена існує взаємозв'язок ОДИН-ДО-БАГАТЬОХ. Певний екземпляр типу запису-члена в екземплярі даного типу набору не може одночасно належати більш ніж одному екземпляру типу запису-володаря. Іншими словами, унікальність володаря типу набору є обов'язковою. В моделі даних, що являє собою взаємозв'язок ОДИН-ДО-БАГАТЬОХ, тип запису-володаря має від 0 до N екземплярів типу запису-члена. В свою чергу тип запису-члена в іншому наборі може мати роль типу запису-володаря. Запис-володар даного набору може грати дану роль у декількох наборах. Така структура є ієрархію. Отже, ієрархічна модель даних є окремим випадком мережної моделі.

Суттєва відмінність між мережною та ієрархічною моделями даних полягає в тому, що в мережній моделі кожний запис може брати участь у будь-якій кількості наборів. На рис.3.16 в мережній моделі, що представляється двома типами наборів ВЧИТЕЛЬ_ ВИКЛАДАЄ_ ДИСЦИПЛІНУ та СТУДЕНТ_НАВЧАЄТЬСЯ _ДИСЦИПЛІНІ, запис-член ДИСЦИПЛІНА входить до обох типів наборів та є зв'язкою цих наборів

832 група куратор Іванов І.П. каф. кібернетики	1013 Захаров І.П. староста групи	201К Кирилова О.Б.	426Х Харламов О.М.

Рисунок 3.15 – Екземпляр набору СКЛАД_ГРУПИ

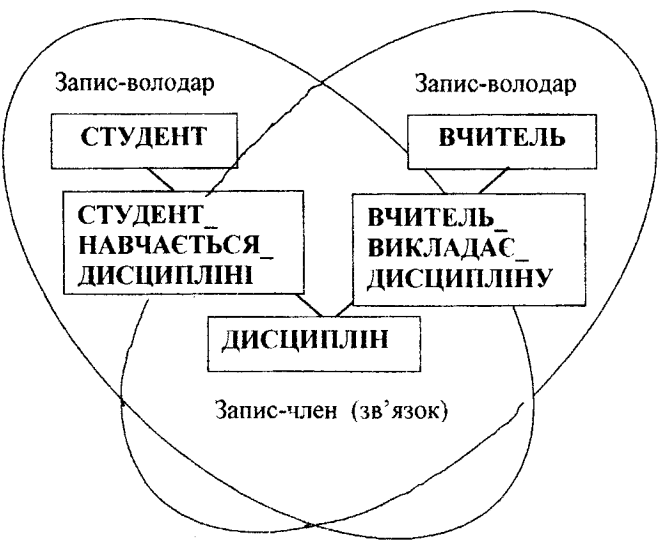


Рисунок 3.16 – Представлення відношення M:N у мережній моделі

Крім того, будь-який запис мережної моделі може мати роль як володаря, так і члена набору.

До переваг мережних структур слід віднести наявність СКБД, які успішно реалізують таку організацію, а також простоту реалізації зв'язків "багато до багатьом", які часто зустрічаються в реальному світі. Недолік таких структур полягає в їх складності по відношенню до ієрархічних структур. Прикладному програмісту часто необхідно знати логічну структуру бази даних. При реорганізації бази даних не виключається в ряді випадків втрата незалежності даних.

Представлення даних ієрархічними та мережними структурами, в загальному випадку, перешкоджає внесенню багатьох змін, пов'язаних з розширенням бази даних. Це може призвести до порушення логічного представлення даних, схем, підсхем, а, отже, до зміни в прикладних програмах.

Контрольні запитання

1. Які типи зв'язків використовують в мережних структурах?
2. В чому відмінність простої і складної мережних структур?
3. В чому полягають переваги та недоліки мережних структур?
4. Яким чином мережні структури подаються ієрархічними?

3.3 Реляційна модель даних

В деяких випадках зростання ієрархічної чи мережної бази даних може привести до порушення логічної організації даних. Такі ситуації виникають при появі нових користувачів, нових застосувань та видів запитів, при обліку інших логічних зв'язків між елементами даних.

Недоліки ієрархічної і мережної моделей привели до появи нової, реляційної моделі даних, створеної Коддом у 1970 році. Реляційна модель була спробою спростити структуру бази даних. У ній були відсутні явні

покажчики на предків і нащадків, а всі дані були представлені у вигляді простих таблиць, розбитих на рядки і стовпці.

Реляційною називається база даних, у якій усі дані, доступні користувачу, організовані у вигляді таблиць, а всі операції над даними зводяться до операцій над цими таблицями. Для представлення реляційних баз даних розроблена формальна теорія баз даних, теоретичну основу якої складає алгебра та математична логіка.

У реляційній базі даних інформація організована у вигляді таблиць, розділених на рядки і стовпці, на перетині яких містяться значення даних. У кожної таблиці є унікальне ім'я, що описує її вміст. Масив значень, що можуть міститися в стовпці, називається доменом цього стовпця.

Двовимірні таблиці в математиці отримали назву *відношення* (relation (англ.)) .

У кожного стовпця в таблиці є своє ім'я, що звичайно служить заголовком стовпця. Всі стовпці в одній таблиці повинні мати унікальні імена, однак дозволяється привласнювати однакові імена стовпцям, розташованим в різних таблицях.

Стовпці таблиці упорядковані зліва направо, і їхній порядок визначається при формуванні таблиці. У будь-якій таблиці завжди є як мінімум один стовпець.

Як правило, не вказується максимально допустиме число стовпців у таблиці, однак майже у всіх комерційних СКБД ця межа існує і, як правило, складає приблизно 255 стовпців.

На відміну від стовпців, рядки таблиці не мають визначеного порядку. Це значить, що якщо послідовно виконати два однакових запити для відображення вмісту таблиці, то немає гарантії, що обидва рази рядки будуть перераховані в тому самому порядку.

Рядки таблиці утворюють данні різного формату і різного типу, тобто можна стверджувати, що рядки таблиці с кортежами.

У таблиці може міститися будь-яка кількість рядків. Цілком

припустиме існування таблиці з нульовою кількістю рядків. Така таблиця називається порожньою. Порожня таблиця зберігає структуру, визначену її стовпцями, просто в ній не містяться дані. Стандарти реляційних баз даних не накладають обмежень на кількість рядків у таблиці, і в багатьох СКБД розмір таблиць обмежений лише вільним дисковим простором комп'ютера.

Як правило, в сучасних реляційних БД допускається збереження символьних, числових даних, бітових рядків, спеціалізованих числових даних (таких як "гроші"), а також спеціальних "темпоральных" даних (дата, час, часовий інтервал).

Найменша одиниця даних реляційної моделі – це окреме атомарне (неподільне) для даної моделі значення даних. Так, в одній предметній області прізвище, ім'я і по-батькові можуть розглядатися як єдине значення, а в інший – як три різних значення.

Опис кожного відношення складається з імені відношення (підмет), за яким в круглих дужках перераховується список атрибутів (присудок). Цей опис називають інтенсіоналом або схемою відношення. Під описом розуміють деяке заповнення кортежів відношення, яке називають екстенсіоналом.

Відношення називаються еквівалентними, якщо вони відрізняються тільки порядком чергування атрибутів.

Кодд у 1985 році сформулював 12 правил, які повинна задовольняти, будь-як база даних, що претендує на тип реляційної. З того часу дванадцять правил Кодда вважаються визначенням реляційної СУБД.

Дванадцять правил Кодда, яким повинна відповідати реляційна СКБД:

1. **Правило інформації.** Вся інформація в базі даних повинна бути надана винятково на логічному рівні і тільки одним способом - у вигляді значень, що містяться в таблицях.
2. **Правило гарантованого доступу.** Логічний доступ до всіх і кожного елемента даних (атомарного значення) у реляційній базі даних

повинний забезпечуватися шляхом використання комбінації імені таблиці, первинного ключа та імені стовпця.

3. **Правило підтримки недійсних значень.** У реляційній базі даних повинна бути реалізована підтримка недійсних значень, що відрізняються від рядка символів нульової довжини, рядка символів пропусків, будь-якого іншого числа і використовуватися для представлення відсутніх даних незалежно від типу цих даних.
4. **Правило динамічного каталогу, основаного на реляційній моделі.** Опис бази даних на логічному рівні необхідно представити в тому ж вигляді, що й основні дані, щоб користувачі, які мають відповідні права, могли працювати з нею за допомогою тієї ж реляційної мови, яку вони застосовують для роботи з основними даними.
5. **Правило вичерпної підмови даних.** Реляційна система може підтримувати різні мови і режими взаємодії з користувачем (наприклад, режим питань і відповідей). Однак повинна існувати принаймні одна мова, оператори якої підтримують такі елементи:
   - визначення даних;
   - визначення представлень;
   - обробку даних (інтерактивну і програмну);
   - умови цілісності;
   - ідентифікацію прав доступу;
   - границі транзакцій (початок, завершення і скасування).
6. **Правило відновлення представлень.** Усі представлення, які теоретично можна обновити, повинні бути доступні для відновлення.
7. **Правило доповнення, відновлення і вилучення.** Можливість працювати з відношенням як з одним операндом повинна існувати не тільки при читанні даних, але і при доповненні, відновленні і вилученні даних.
8. **Правило незалежності фізичних даних.** Прикладні програми й утиліти для роботи з даними повинні на логічному рівні залишатися

недоторканими при будь-яких змінах методів збереження даних чи методів доступу до них.

9. **Правило незалежності логічних даних.** Прикладні програми й утиліти для роботи з даними повинні на логічному рівні залишатися недоторканими при внесенні в базові таблиці будь-яких змін, які теоретично дозволяють зберегти недоторканими дані, що містяться в цих таблицях.
10. **Правило незалежності умов цілісності.** Повинна існувати можливість визначати умови цілісності, специфічні для конкретної реляційної бази даних, на підмові реляційної бази даних і зберігати їх у каталозі, а не в прикладній програмі.
11. **Правило незалежності поширення.** Реляційна СКБД не повинна залежати від потреб конкретного клієнта.
12. **Правило одиничності.** Якщо в реляційній системі є низькорівнева мова (що обробляє один запис за один раз), то повинна бути відсутня можливість використання її для того, щоб обійти правила й умови цілісності, виражені на реляційній мові високого рівня (що обробляє кілька записів за один раз).

Правило 2 вказує на роль первинних ключів при пошуку інформації в базі даних. Ім'я таблиці дозволяє знайти необхідну таблицю, ім'я стовпця дозволяє знайти необхідний стовпець, а первинний ключ дозволяє знайти рядок, який містить шуканий елемент даних.

Правило 3 вимагає, щоб відсутні дані можна було представити за допомогою недійсних значень (NULL).

Правило 4 говорить, що реляційна база даних повинна сама себе описувати. Іншими словами, база даних повинна містити набір системних таблиць, що описують структуру самої бази даних.

Правило 5 вимагає, щоб СКБД використовувала мову реляційної бази даних, наприклад SQL, хоча явно SQL у правилі не згадують. Така мова повинна підтримувати всі основні функції СКБД — створення бази даних,

читання і введення даних, реалізацію захисту бази даних і т.д.

Правило 6 дозволяє показувати різним користувачам різні фрагменти структури бази даних.

Правило 7 акцентує увагу на тому, що бази даних за своєю природою орієнтовані на множини. Воно вимагає, щоб операції доповнення, вилучення і відновлення можна було виконувати над множинами рядків. Це правило призначене для того, щоб заборонити реалізації, у яких підтримуються тільки операції над одним рядком.

Правила 8 і 9 стверджують, що конкретні способи реалізації чи збереження доступу, які використовуються в СКБД, і навіть зміни структури таблиць бази даних не повинні впливати на можливість користувача працювати з даними.

Правило 10 вимагає, щоб мова бази даних підтримувала обмежувальні умови, які накладаються на дані, що вводяться, і дії, що можуть бути виконані над даними.

Правило 11 говорить, що мова бази даних повинна забезпечувати можливість роботи з розподіленими даними, розташованими на інших комп'ютерних системах.

І, нарешті, правило 12 запобігає використанню інших можливостей для роботи з базою даних, крім мови бази даних, оскільки це може порушити її цілісність.

Згідно Дейту реляційна модель складається з трьох частин, що описують різні аспекти реляційного підходу: структурної частини, маніпуляційної частини та цілісної частини.

У структурній частині моделі фіксується, що єдиною структурою даних, яка використовується в реляційних БД, є нормалізоване n-арне відношення.

В маніпуляційній частині моделі стверджується два фундаментальних механізми маніпулювання реляційними БД - реляційна алгебра і реляційне числення. Перший механізм базується в основному на класичній теорії

множин (з деякими уточненнями), а другий - на класичному логічному апараті числення предикатів першого порядку.

У цілісній частині реляційної моделі даних фіксуються дві базові вимоги цілісності, що повинні підтримуватися в будь-якій реляційній СКБД. Перша вимога називається вимогою цілісності сутностей. Об'єкту або сутності реального світу в реляційній БД відповідають кортежі відношень. Конкретно вимога полягає в тому, що будь-який кортеж будь-якого відношення відрізняється від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення повинно мати первинний ключ. Друга вимога називається вимогою цілісності по посиланнях. Очевидно, що при дотриманні нормалізованності відношень складні сутності реального світу представляються в реляційній БД у вигляді декількох кортежів декількох відношень.Вимога цілісності по посиланнях, або вимога зовнішнього ключа полягає в тому, що для кожного значення зовнішнього ключа, який з'являється у відношенні, до якого посилаються, повинний існувати кортеж із таким же значенням первинного ключа. Так, наприклад, якщо для співробітника зазначений номер відділу, то цей відділ повинний існувати.

Степінь відношення - це число його атрибутів. Відношення степеня один називають унітарним, степеня два - бінарним, степеня три - тернарним, ... , а степеня n - n-арним.

Кардинальне число або потужність відношення - це число його кортежів. Кардинальне число відношення змінюється в часі на відміну від його степеня.

До переваг реляційної бази можна віднести незалежність від фізичного рівня представлення, зручність і розуміння організації даних користувачами, максимальна гнучкість при обробці непередбачених запитів, можливість розширення бази приєднанням нових елементів й записів без зміни при цьому існуючих підсхем та прикладних програм.

Контрольні запитання

1. Назвіть 12 правил Кодда, які повинна задовольняти реляційна БД.
2. Як визначається степінь та кардинальне число відношення?
3. Чи існують в сучасних СКБД обмеження на розміри відношень?
4. Перерахуйте переваги та недоліки реляційної моделей даних.

4 ОСНОВИ ПРОЕКТУВАННЯ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

4.1 Поняття ключа, основні типи ключів

Якщо кожному стовпцю таблиці присвоїти ім'я, то розміщення стовпців буде несуттєвим. Список імен атрибутів одного відношення називається схемою заміщення; кожне відношення, як правило, має свою назву (ім'я). Якщо, наприклад, відношення (ШИФР ГРУПИ, КІЛЬКІСТЬ СТУДЕНТІВ, ФАКУЛЬТЕТ, СТАРОСТА, КУРАТОР) назвемо ГРУПА, то схема буде мати атрибути ШИФР ГРУПИ, КІЛЬКІСТЬ СТУДЕНТІВ, ФАКУЛЬТЕТ, СТАРОСТА, КУРАТОР. Формально схема відношення описується таким чином:

ІМ'Я_ВІДНОШЕННЯ (A_1, A_2, A_3, ..., A_K),

де A_1, A_2, ..., A_K – імена атрибутів.

Стовпець чи ряд стовпців, значення яких однозначно ідентифікують кожний рядок таблиці, називають ключем. Так, в розглянутому прикладі можливими ключами можуть бути атрибути: ШИФР ГРУПИ, СТАРОСТА. А такі атрибути, як КІЛЬКІСТЬ СТУДЕНТІВ, ФАКУЛЬТЕТ, не можуть бути ключами, оскільки різні студентські групи можуть мати однакову кількість студентів і на одному факультеті вчиться декілька груп.

Якщо відношення має більше одного можливого ключа, тоді має сенс виділити один ключ, який називають первинним (ШИФР ГРУПИ). І навпаки, якщо відношення не має жодного атрибута, який би повністю визначив об'єкт (рядок, кортеж), тоді приходиться мати справу зі складеним ключем, який схематично відображають подвійною горизонтальною рискою. Наприклад, в відношенні СТУДЕНТ-УСПІШНІСТЬ (табл.4.1) відсутній атрибут, який би однозначно ідентифікував кортеж, оскільки екзамени з одного навчального курсу (фізика) можуть здаватися в декількох семестрах.

Таблиця 4.1 - **Відношення СТУДЕНТ-УСПІШІСТЬ зі складеним ключем**

Прізвище, ім'я, по батькові студента	Назва Дисципліни	Дата здачі екзамену	Оцінка	Залік
Іванов П.В.	фізика	13.12.98	4	Залік
Іванов П.В.	фізика	17.12.98	5	Залік
Іванов П.В.	математичний аналіз	13.12.98	3	Залік
Петров А.С	фізика	3.01.99	3	Залік

Позначимо атрибути відношення СТУДЕНТ-УСПІШНІСТЬ відповідно:

ПІБ СТУДЕНТА - F;

НАЗВА ДИСЦИПЛІНИ - N;

ДАТА ЗДАЧІ ЕКЗАМЕНУ - D;

ОЦІНКА - C;

ЗАЛІК - Z.

Тоді відношення СТУДЕНТ-УСПІШНІСТЬ, що містить атрибути F, N, D, C, Z, можна представити у вигляді схеми (рис. 4.1).

СТУДЕНТ-УСПІШНІСТЬ

F	D	N	C	Z

Рисунок 4.1 – Схематичне представлення відношення СТУДЕНТ-УСПІШНІСТЬ

Щоб виділити вказаним способом складений ключ відношення СТУДЕНТ-УСПІШНІСТЬ, потрібно представити домени відношення таким чином, щоб мінімальна кількість атрибутів, які однозначно визначають кортеж, були розміщені на початку схеми. Відношення СТУДЕНТ-УСПІШНІСТЬ прийме вигляд (рис.4.2):

СТУДЕНТ-УСПІШНІСТЬ

F	D	N	C	Z

Рисунок 4.2 – Складений ключ

Атрибути F і D визначають кожний рядок даної таблиці. Якщо припустити, що можлива здача двох і більше екзаменів одним студентом в один день, тоді атрибути F, D вже не будуть однозначно ідентифікувати кортеж, оскільки даним екземплярам атрибутів F, D може відповідати в таких випадках два і більше кортежі. При цих умовах складений ключ буде містити три атрибути, що схематично зображено на рис 4.3.

СТУДЕНТ-УСПІШНІСТЬ

F	D	N	C	Z

Рисунок 4.3 – Складений ключ, який включає три атрибути

Можливі випадки, коли складений ключ включає всі атрибути відношення.

Розрізняють наступні типи ключів: простий ключ, повністю складений ключ, напівскладений ключ.

Простим називається ключ, що складається з одного атрибута, причому атрибут повинен бути атомарним, а екземпляри даного атрибута – унікальні.

Атомарним є ключ, значення якого сприймається програмою (СКБД) як неподільний елемент даних, навіть якщо він створений з декількох об'єктів. Наприклад, атомарність атрибута ДАТА ЗДАЧІ ЕКЗАМЕНА в відношенні означає неможливість виділення з дати окремо числа, місяця та року.

Повністю складеним називається ключ, що містить декілька атрибутів, причому між цими атрибутами існує відображення (залежність) БАГАТО-ДО-БАГАТЬОХ. Атрибути, що складають такий ключ, не залежать один від одного. Приклад повністю складеного ключа в відношенні СТУДЕНТ-ВИКЛАДАЧ приведено в табл. 4.2.

Таблиця 4.2 - **Приклад відношення з повністю складеним ключем СТУДЕНТ-ВИКЛАДАЧ**

Студент	Викладач	Дисципліна
Іванов	Харитонов	фізика
Петров	Харитонов	фізика
Іванов	Міщенко	інформатика
Петров	Борисов	інформатика

Напівскладеним називається ключ, що містить декілька атрибутів, між якими на відміну від повністю складеного ключа існує залежність Б:1 (БАГАТО-ДО-ОДНОГО). Тут атрибути в ключі впорядковуються за принципом: кожний наступний уточнює попередній. У відношенні СТУДЕНТ-УСПІШНІСТЬ використовувався напівскладений ключ F, D .

Взаємозв'язок таблиць є найважливішим елементом реляційної моделі даних. Вона підтримується зовнішніми ключами (*foreign key*). Розглянемо приклад, у якому база даних зберігає інформацію про рядових співробітників (таблиця СПІВРОБІТНИК) і керівників (таблиця КЕРІВНИК) у деякій організації (рис.4.4.). Первинний ключ таблиці КЕРІВНИК - стовпець НОМЕР (наприклад, табельний номер). Стовпець ПРІЗВИЩЕ не може виконувати роль первинного ключа, тому що в одній організації можуть працювати два керівники з однаковими прізвищами. Будь-який співробітник підпорядковується єдиному керівнику, що повинно бути відображено в

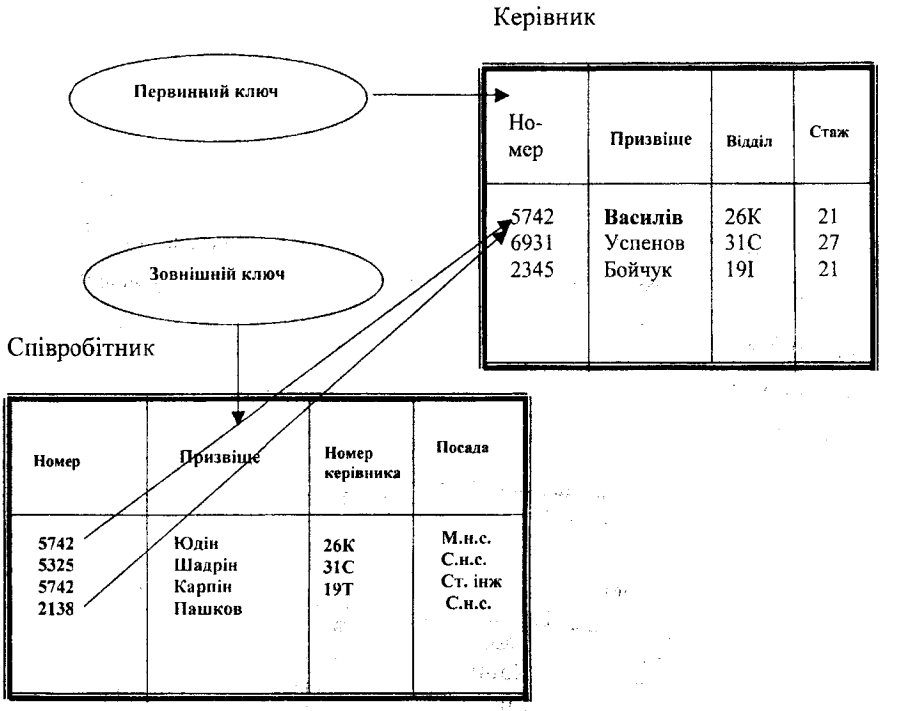


Рисунок 4.4 – Схематичне представлення зовнішнього ключа

в базі даних. Таблиця СПІВРОБІТНИК містить стовпець НОМЕР керівника, і значення в цьому стовпці вибираються зі стовпця НОМЕР таблиці КЕРІВНИК (див. рис.4.4). Стовпець НОМЕР КЕРІВНИКА є зовнішнім ключем для таблиці СПІВРОБІТНИК.

Контрольні запитання

1. Які основні вимоги висуваються до ключів?
2. Які типи відношень між атрибутами характерні для різних типів ключів?
3. В яких випадках застосовується зовнішні ключі?
4. Коли ключ включає всі атрибути відношення?

4.2 Основи реляційної алгебри

Реляційна модель баз даних надає можливість маніпулювати над доменами відношень. Для цих цілей існує два види апаратів маніпулювання відношеннями: реляційна алгебра (алгебра відношень) і реляційне обчислення (обчислення відношень). Алгеброю відношень називають систему операцій маніпулювання відношеннями, кожний оператор якого в якості операнда (операндів) використовує одне чи більше відношень і утворює нове відношення за попередньо обумовленим правилом.

Реляційне обчислення дозволяє шляхом використання обчислення предикатів та кванторів змінних описувати відношення та операції над ними в вигляді аналітичного виразу або формули.

В реляційній алгебрі використовують п'ять основних операцій: об'єднання, різниця, декартовий добуток, проекція і селекція.

Проекція. Суть цієї операції полягає в тому, що береться відношення R, видаляються деякі з його компонентів і перевпорядковуються компоненти, що залишились. Якщо в результаті проекції з'являються однакові кортежі, то вони з результуючого відношення вилучаються.

Операція проекції полягає в виділенні необхідних стовпців (доменів) з відношення. Нехай дано відношення СТУДЕНТ-УСПІШНІСТЬ (табл. 4.3).

Таблиця 4.3 - **СТУДЕНТ-УСПІШНІСТЬ**

ПІБ студента	Назва дисципліни	Дата здачі екзамену	Оцінка
Іванов П.В.	Фізика	13.01.2000	4
Іванов П.В.	Фізика	14.06.2000	5
Іванов П.В.	Мат.аналіз	10.01.2000	3
Петров А.С.	Фізика	3.01.2000	3

В результаті виконання операції проекції отримуємо нове відношення, яке представлено в табл.4.4.

Таблиця 4.4 - **Приклад операції "проекція"**

ПІБ студента	Назва дисципліни	Оцінка
Іванов П.В.	Фізика	4
Іванов П.В.	Фізика	5
Іванов П.В.	Мат. аналіз	3
Петров А.С.	Фізика	3

Об'єднання. Об'єднання відношень R і S (позначається R∪S) представляє собою множину кортежів, які належать R чи S або їм обом. Оператор об'єднання застосовується тільки до відношень однакової арності. Якщо в результаті об'єднання відношень мають місце однакові кортежі, то вони заміняються одним.

Нехай задано два відношення, представлені таблицями 4.5, 4.6. Виконаємо над ними операцію об'єднання. В результаті об'єднання відношень отримуємо результуюче відношення, яке представлено в табл.4.7.

Таблиця 4.5 – **ВИКЛАДАЧІ – ДИСЦИПЛІНИ**

Викладачі	Дисципліни
Майданюк В.П.	Мікропроцесорні системи
Власюк В.Х.	Основи програмування
Романюк О.Н.	Засоби машинної графіки

Таблиця 4.6 – **ВИКЛАДАЧІ – ДИСЦИПЛІНИ**

Викладачі	Дисципліни
Арапов С.М.	Експертні системи
Войтко В.В.	Мережі ЕОМ
Обідник Д.Т.	Схемотехніка ЕОМ

Таблиця 4.7 – **ВИКЛАДАЧІ – ДИСЦИПЛІНИ**

Викладачі	Дисципліни
Арапов С.М.	Експертні системи
Войтко В.В.	Мережі ЕОМ
Обідник Д.Т.	Схемотехніка ЕОМ
Майданюк В.П.	Мікропроцесорні системи
Власюк В.Х.	Основи програмування
Романюк О.Н.	Засоби машинної графіки

6. **Різниця.** Різницею відношень R і S (позначається як R-S), називається множина кортежів, які належать R, але не належать S (рис.4.5). При реалізації різниці необхідно, щоб R і S мали одну і ту ж саму арність.

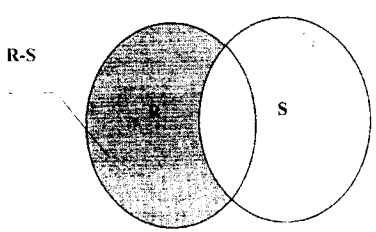


Рисунок 4.5 – Графічна ілюстрація операції різниці відношень

Якщо A - відношення про жителів мікрорайону, B- відношення про тих, хто пройшов медичний огляд, то відношення (A-B) буде містити дані про тих жителів мікрорайону, хто не пройшов медичний огляд.

Декартовий добуток. Нехай R і S – відношення арності k_1 і k_2 відповідно. Тоді декартовим добутком відношень R і S називається множина кортежів довжини $(k_1 + k_2)$, перші k_1 компонентів яких утворюють кортежі, які належать R, а останні k_2 – кортежі, що належать S.

Наприклад :

$$\begin{bmatrix} X & A \\ Y & A \\ Z & A \\ W & B \end{bmatrix} \otimes \begin{bmatrix} 5 & A \\ 1 & B \end{bmatrix} = \begin{bmatrix} X & A & 5 & A \\ X & A & 1 & B \\ Y & A & 5 & A \\ Y & A & 1 & B \\ Z & A & 5 & A \\ Z & A & 1 & B \\ W & B & 5 & A \\ W & B & 1 & B \end{bmatrix}$$

Результатом декартового добутку відношень СТУДЕНТИ (табл. 4.8) та ЕКЗАМЕНИ (табл.4.9) буде відношення ЕКЗАМЕНАЦІЙНА ВІДОМІСТЬ (табл.4.10).

Таблиця 4.8 – **СТУДЕНТИ**

Студенти
Гаврилюк
Піддубчак
Скрипник
Ящук

Таблиця 4.9 – **ЕКЗАМЕНИ**

Дисципліна	Дата	Оцінка
Математика	10.03.2000	
Фізика	15.03.2000	

Таблиця 4.10 – **ЕКЗАМЕНАЦІЙНА ВІДОМІСТЬ**

Студенти	Дисципліна	Дата	Оцінка
Гаврилюк	Математика	10.03.2000	
Гаврилюк	Фізика	15.03.2000	
Піддубчак	Математика	10.03.2000	
Піддубчак	Фізика	15.03.2000	
Скрипник	Математика	10.03.2000	
Скрипник	Фізика	15.03.2000	
Ящук	Математика	10.03.2000	
Ящук	Фізика	15.03.2000	

Селекція. Нехай F – формула, яка може бути утворена такими засобами: а) операндами, які є константами чи номерами компонентів; б) арифметичними операторами порівняння $<, =, >, \neq, \geq, \leq$; в) логічними операторами $\wedge$(І), $\vee$(АБО), $\neg$(НІ).

В цьому випадку $\delta_f(R)$ є множина кортежів t, які належать R, таких, що при підстановці *i*–го компонента t замість будь-якого входження номера *i* в формулу F для всіх *i* вона стане істиною. Наприклад, $\delta_{2>3}(R)$ означає множину кортежів, що належать R, другий компонент яких більше третього компонента.

При реалізації селекції відношення, приведеного в табл.4.11, згідно з

Таблиця 4.11 - **Відношення**

ПІБ студента	Вік
Гаврилюк	21
Піддубчак	20
Скрипник	23
Ящук	19

ознакою (вік>20), отримуємо відношення (табл.4.12):

Таблиця 4.12 – **Відношення, над яким виконано операцію селекції**

ПІБ студента	Вік
Гаврилюк	21
Скрипник	23

Крім перерахованих операцій існують і інші, але їх можна отримати з п'яти основних.

Розглянемо неосновні операції перетину та ділення.

Перетин R∩S двох відношень R та S знаходиться згідно формул (R-(R-S)). Нехай R та S є відношення арності r і s відповідно, де (r > s) і $S \neq \{\otimes\}$. Тоді **частка** (R÷S) є множина кортежів *t* довжини (r-s) таких, що для всіх кортежів *u* довжини s, які належать S, кортеж *tu* належить R. Виконаємо операцію ділення над відношеннями, які представлені відповідно таблицями 4.13 та 4.14. В результаті отримуємо частку (табл.4.15).

Схематично операції реляційної алгебри представлені на рис.4.6.

Розроблені мови маніпулювання даними, що дозволяють реалізувати всі операції реляційної алгебри і практично будь-які їх сполучення. Серед

Таблиця 4.13 - **Екзаменаційна відомість**

Студенти	Дисципліна	Дата	Оцінка
Гаврилюк	Математика	10.03.2000	3
Гаврилюк	Фізика	15.03.2000	5
Піддубчак	Математика	10.03.2000	4
Піддубчак	Фізика	15.03.2000	5
Скрипник	Математика	10.03.2000	5
Скрипник	Фізика	15.03.2000	4
Ящук	Математика	10.03.2000	4
Ящук	Фізика	15.03.2000	5

Таблиця 4.14 – **Відношення**

Дисципліна	Оцінка
Математика	4
Фізика	5

Таблиця 4.15 – **Відношення – частка**

Студенти
Піддубчак
Ящук

них найбільше поширені SQL (Structured Query Language - структуризована мова запитів) і QBE (Quere-By-Example - запити за зразком) Обидві відносяться до мов дуже високого рівня, за допомогою яких користувач вказує, які дані необхідно одержати, не уточнюючи процедуру їхнього формування. За допомогою єдиного запиту на будь-якій із цих мов можна з'єднати декілька таблиць у тимчасову таблицю і вирізувати з неї необхідні рядки і стовпці (селекція і проекція).

Контрольні запитання

1. Скільки основних операції використовується в реляційній алгебрі?
2. Чи може відношення мати два однакових кортежі після виконання операцій проекції чи об'єднання?
3. Які основні операції використовуються при знаходженні частки?
4. Як змінюється потужність відношень при виконанні операцій реляційної алгебри?

4.3 Нормалізація схем баз даних

Нормалізація - це розбивка таблиці на дві або більше, які характеризуються кращими властивостями при доповненні, зміні і вилученні даних. Кінцева мета нормалізації зводиться до отримання такого проекту бази даних, у якому кожний факт з'являється лише в одному місці, тобто виключена надлишковість інформації. Це робиться не стільки з метою економії пам'яті, скільки для виключення можливої суперечливості збережених даних.

Кожна таблиця в реляційній БД задовольняє умову, у відповідності з якою у позиції на перетині кожного рядка і стовпця таблиці завжди знаходиться єдине атомарне значення і ніколи не може бути множини таких значень. Будь-яка таблиця, що задовольняє цю умову, називається нормалізованою.

Кожній нормальній формі відповідає деякий визначений набір обмежень. Відношення знаходиться в деякій нормальній формі, якщо задовольняється властивий їй набір обмежень.

Кожна нормальна форма є більш обмеженою і більш бажаною, ніж попередня. Це пов'язано з тим, що в (N+1)-ій нормальній формі вилучаються деякі небажані властивості, які характерні N-ій нормальній формі. Теорія нормалізації грунтується на наявності тієї або іншої залежності між полями таблиці.

Основні властивості нормальних форм:

- кожна наступна нормальна форма в деякому змісті краща попередньої;
- при переході до наступної нормальної форми властивості попередніх нормальних властивостей зберігаються.

Найбільше важливі нормальні форми відношень грунтуються на фундаментальному в теорії реляційних баз даних понятті функціональної

залежності.

Визначення 1. Функціональна залежність.

У відношенні R атрибут Y функціонально залежить від атрибута X (X і Y можуть бути складовими) у тому і тільки в тому випадку, якщо кожному значенню X відповідає в точності одне значення Y: $X \rightarrow Y$.

Визначення 2. Повна функціональна залежність.

Функціональна залежність $X \rightarrow Y$ називається повною, якщо атрибут Y не залежить функціонально від будь-якої підмножини X.

Визначення 3. Транзитивна функціональна залежність.

Функціональна залежність називається транзитивною, якщо з функціональних залежностей $X \rightarrow Y$ та $Y \rightarrow Z$ випливає, що $X \rightarrow Z$.

Наприклад, Вінниця входить до Поділля, а Поділля - до України. Для даного прикладу має місце транзитивна залежність ВІННИЦЯ $\rightarrow$ УКРАЇНА.

Визначення 4. Неключовий атрибут.

Неключовим атрибутом називається будь-який атрибут відношення, що не входить до складу первинного ключа.

Визначення 5. Взаємно незалежні атрибути.

Два або більше атрибути взаємно незалежні, якщо жодний із цих атрибутів не є функціонально залежним від інших.

Відношення R задано в першій нормальній формі, якщо воно задано у вигляді множини своїх кортежів, які не повторюються.

Для того, щоб представити відношення в першій нормальній формі необхідно над його кортежами виконати операцію проекції для видалення рядків, які повторюються.

Відношення R задано в другій нормальній формі, якщо воно, по-перше, є відношенням у першій нормальній формі і, по-друге, кожний його атрибут, який не є основним атрибутом, функціонально повно залежить від будь-якого можливого ключа цього відношення.

Нехай задано відношення ФАКУЛЬТЕТ (НАЙМЕНУВАННЯ, ПІБ ДЕКАНА, ТЕЛЕФОН).

Відношення ФАКУЛЬТЕТ задано в другій нормальній формі, тому що у відношенні ФАКУЛЬТЕТ атрибут ТЕЛЕФОН, який не є основним, повністю залежить від будь-якого можливого ключа: НАЙМЕНУВАННЯ, ПІБ ДЕКАНА.

У загальному випадку, якщо всі можливі ключі відношення містять по одному атрибуту, то це відношення задане в другій нормальній формі, тому що в цьому випадку всі атрибути, які не є основними, функціонально повно залежать від можливих ключів. Однак це твердження не завжди справедливе, якщо ключ відношення R є складовим.

Розглянемо відношення:

СТУДЕНТ – КУРС_ПРОЕКТ (НОМЕР_ЗАЛІКОВОЇ_КНИЖКИ, КОД_ПРЕДМЕТУ, ПРІЗВИЩЕ_СТУДЕНТА, НОМЕР_ГРУПИ, ВИКЛАДАЧ, ПРОЦЕНТ_ВИКОНАННЯ).

Припустимо, що в одній групі можуть навчаються однофамільці. Тоді для цього відношення можливий тільки один ключ: НОМЕР_ЗАЛІКОВОЇ КНИЖКИ, КОД_ПРЕДМЕТУ. Виходячи з прийнятого припущення, атрибут ПРІЗВИЩЕ_СТУДЕНТА не входить у ключ. Тоді атрибут НОМЕР_ ЗАЛІКОВОЇ_КНИЖКИ не визначається значенням атрибута ПРІЗВИЩЕ_ СТУДЕНТА, тобто атрибути ПРІЗВИЩЕ_СТУДЕНТА і НОМЕР_ГРУПИ не є основними, але функціонально залежать від основного атрибута НОМЕР_ЗАЛІКОВОЇ_КНИЖКИ, що входить у складовий ключ. Функціональні залежності між атрибутами цього відношення показані на рис. 4.7.

Розщепивши вихідне відношення на два нових у другій нормальній формі, можна усунути надлишковість (рис.4.8). При виконанні цієї операції розбивки на два відношення враховано те, що атрибути, які функціонально залежать від одного основного атрибута разом із ним

утворять одне відношення з єдиним ключем НОМЕР_ЗАЛІКОВОЇ_КНИЖКИ, а інші атрибути, які функціонально повно залежать від складового ключа, залишено у вихідній схемі.

Розглянемо приклад схеми відношення:

СПІВРОБІТНИКИ - ВІДДІЛИ - ПРОЕКТИ

(СПІВРОБ_НОМЕР,СПІВРОБ_ЗАРП, ВІДДІЛ_НОМЕР, ПРО_НОМЕР, СПІВРОБ_ЗАВДАННЯ).

У відношенні використані скорочення: СПІВРОБ - співробітник, ЗАРП - зарплата, ПРО - проект.

Первинний ключ:

СПІВРОБ_НОМЕР, ПРО_НОМЕР.

Функціональні залежності:

СПІВРОБ_НОМЕР -> СПІВРОБ_ЗАРП

СПІВРОБ_НОМЕР -> ВІДДІЛ_НОМЕР

ВІДДІЛ_НОМЕР -> СПІВРОБ_ЗАРП

СПІВРОБ_НОМЕР, ПРО_НОМЕР -> СПІВРОБ_ЗАВДАННЯ.

Хоча первинним ключем є складовий атрибут СПІВРОБ_НОМЕР, ПРО_НОМЕР, атрибути СПІВРОБ_ЗАРП і ВІДДІЛ_НОМЕР функціонально залежать від частини первинного ключа, тобто атрибута СПІВРОБ_НОМЕР. В результаті, неможливо вставити у відношення СПІВРОБІТНИКИ-ВІДДІЛИ-ПРОЕКТИ кортеж, що описує співробітника, який ще не виконує ніякого проекту (первинний ключ не може містити невизначене значення). При видаленні кортежу не тільки руйнується зв'язок даного співробітника з даним проектом, але втрачається інформація про те, в якому відділі він працює. При переводі співробітника в інший відділ необхідно модифікувати всі кортежі, які описують цього співробітника, або одержимо неузгоджений результат. Такі неприємні явища називаються аномаліями схеми відношення. Вони усуваються шляхом нормалізації.

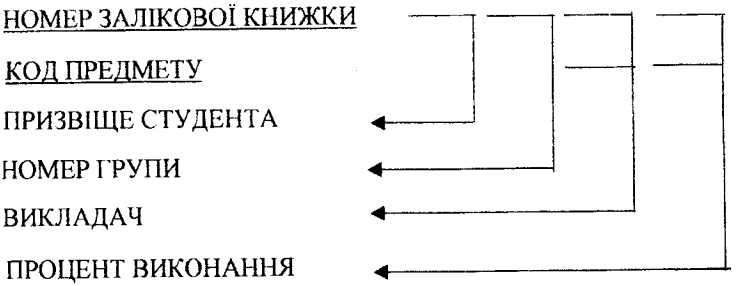


Рисунок 4.7 – Функціональна залежність

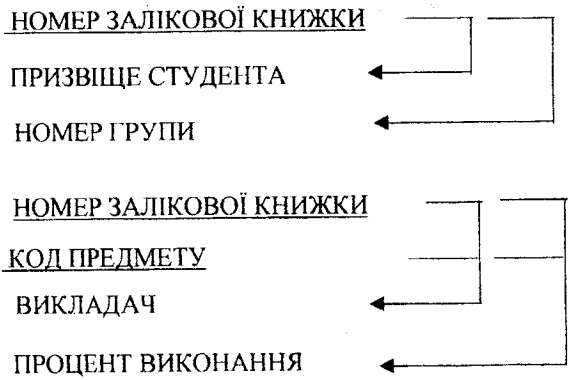


Рисунок 4.8 – Приклад усунення функціональної залежності

Виконаємо декомпозицію відношення СПІВРОБІТНИКИ-ВІДДІЛИ в два відношення СПІВРОБІТНИКИ-ВІДДІЛИ і СПІВРОБІТНИКИ-ПРОЕКТИ:

СПІВРОБІТНИКИ-ВІДДІЛИ (СПІВРОБ_НОМЕР, СПІВРОБ_ЗАРП, ВІДДІЛ_НОМЕР)

Первинний ключ:

СПІВРОБ_НОМЕР

Функціональні залежності:

СПІВРОБ_НОМЕР -> СПІВРОБ_ЗАРП

СПІВРОБ НОМЕР -> ВІДДІЛ НОМЕР

ВІДДІЛ НОМЕР -> СПІВРОБ ЗАРП

СПІВРОБІТНИКИ-ПРОЕКТИ (СПІВРОБ_НОМЕР, ПРО_НОМЕР, СПІВРОБ_ЗАВДАННЯ)

Первинний ключ:

СПІВРОБ_НОМЕР, ПРО_НОМЕР

Функціональна залежність:

СПІВРОБ_НОМЕР, ПРО_НОМЕР -> СПІВРОБ_ЗАВДАННЯ

Кожне з цих двох відношень знаходиться в 2НФ і в них усунуті відзначені вище аномалії .

Відношення R знаходиться в третій нормальній формі (3НФ) у тому і тільки в тому випадку, якщо знаходиться в 2НФ і кожний неключовий атрибут нетранзитивно залежить від первинного ключа.

Наприклад, відношення:

ГУРТОЖИТОК (ПІБ_СТУДЕНТА, НОМЕР_ГРУПИ, НОМЕР_КІМНАТИ, СТАРОСТА_КІМНАТИ) знаходиться в другій нормальній формі, але не в третій, тому що атрибут СТАРОСТА_КІМНАТИ залежить від атрибута НОМЕР_КІМНАТИ, який у свою чергу залежить від атрибута ПІБ_СТУДЕНТА і, отже, СТАРОСТА_КІМНАТИ транзитивно залежить від ПІБ_СТУДЕНТА. Це відношення можна привести до необхідної форми шляхом його розщеплення на два:

СТУДЕНТ-ГУРТОЖИТОК (ПІБ_СТУДЕНТА, НОМЕР_ГРУПИ, НОМЕР_КІМНАТИ) та КІМНАТА-ГУРТОЖИТОК (НОМЕР_КІМНАТИ, СТАРОСТА_КІМНАТИ).

Залежності між атрибутами вихідного й отриманих відношень подані на рис.4.9, звідки видно, що отримані відношення більш доцільніші від вихідного. Так, інформація про старосту кімнати може знадобитися незалежно від інформації про студентів, що проживають у цій кімнаті.

Нехай маємо відношення (ФІРМА, СКЛАД, ОБ'ЄМ).

Для даного відношення характерні такі аномалії:

1. Якщо в даний момент відсутня фірма, яка отримує товар зі складу, то в базу даних неможливо ввести інформацію про об'єм складу.
2. Якщо фірма перестає отримувати товар зі складу, то данні про склад та його об'єм не можна зберігати в базі даних.
3. Якщо об'єм складу змінився, то необхідно переглянути всі рядки відношення і змінити кортежі для форм, пов'язаних зі складом.

Причиною аномалій для даного відношення є наявність транзитивного зв'язку між атрибутами.

Для усунення аномалій розіб'ємо вихідне відношення на два:
ЗБЕРІГАННЯ (ФІРМА, СКЛАД) та ОБ'ЄМ (СКЛАД, ОБ'ЄМ).

На практиці в більшості випадків три нормальні форми схем відношень є достатніми і приведенням до третьої нормальної форми процес проектування реляційної бази даних, як правило, закінчується. Однак іноді корисно продовжити процес нормалізації.

Розглянемо приклад схеми відношення:
СПІВРОБІТНИКИ-ПРОЕКТИ (СПІВРОБІТНИКА_НОМЕР, СПІВРОБІТНИКА_ПРІЗВИЩЕ, ПРОЕКТУ_НОМЕР, СПІВРОБІТНИКА ЗАВДАННЯ).

Можливі ключі (важливо, що на цій стадії нормалізації до уваги приймається існування можливих ключів):
СПІВРОБІТНИКА_НОМЕР, ПРОЕКТУ_НОМЕР;
СПІВРОБІТНИКА_ПРІЗВИЩЕ, ПРОЕКТУ_НОМЕР.

Функціональні залежності:
СПІВРОБІТНИКА_НОМЕР -> СПІВРОБІТНИКА_ПРІЗВИЩЕ;
СПІВРОБІТНИКА_НОМЕР -> ПРОЕКТУ_НОМЕР;
СПІВРОБІТНИКА_ПРІЗВИЩЕ -> СПІВРОБІТНИКА_НОМЕР;
СПІВРОБІТНИКА_ПРІЗВИЩЕ -> ПРОЕКТУ_НОМЕР;

СПІВРОБІТНИКА_НОМЕР, ПРОЕКТУ_НОМЕР -> СПІВРОБІТНИКА_ЗАВДАННЯ;

СПІВРОБІТНИКА_ПРІЗВИЩЕ, ПРОЕКТУ_НОМЕР -> СПІВРОБІТНИКА_ЗАВДАННЯ.

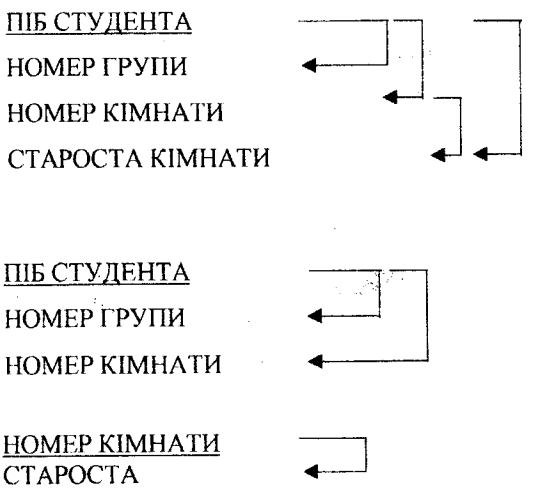


Рисунок 4.9 – Приклад усунення транзитивної залежності

У цьому прикладі припускаємо, що особа співробітника повністю визначається як його номером, так і прізвищем.

Незалежно від того, який із можливих ключів обраний в якості первинного ключа, ця схема знаходиться в 3НФ. Однак той факт, що є функціональні залежності атрибутів відношення від атрибута, що є частиною первинного ключа, приводить до аномалій. Наприклад, для того, щоб змінити ПРІЗВИЩЕ співробітника з даним номером погодженим способом, буде потрібно модифікувати всі кортежі, які включають його номер. Введемо визначення.

Детермінантом називається будь-який атрибут, від котрого функціонально повно залежить деякий інший атрибут.

Нормальна форма Бойса-Кодда. Відношення R знаходиться в *нормальній формі Бойса-Кодда (БКНФ) у тому і тільки в тому випадку,* якщо кожний детермінант є можливим ключем.

Зауважимо, що якщо у відношенні є тільки один можливий ключ (який є первинним ключем), то це визначення стає еквівалентним визначенню третьої нормальної форми.

Очевидно, що ця вимога не виконана для відношення СПІВРОБІТНИКИ-ПРОЕКТИ. Можна виконати його декомпозицію до відношень СПІВРОБІТНИКИ і СПІВРОБІТНИКИ-ПРОЕКТИ:

СПІВРОБІТНИКИ (СПІВРОБІТНИКА_НОМЕР, СПІВРОБІТНИКА_ ПРІЗВИЩЕ) .

Можливі ключі:

СПІВРОБІТНИКА_НОМЕР;

СПІВРОБІТНИКА_ПРІЗВИЩЕ.

Функціональні залежності:

СПІВРОБІТНИКА_НОМЕР -> СПІВРОБІТНИКА_ПРІЗВИЩЕ;

СПІВРОБІТНИКА_ПРІЗВИЩЕ -> СПІВРОБІТНИКА_НОМЕР.

СПІВРОБІТНИКИ-ПРОЕКТИ (СПІВРОБІТНИКА_НОМЕР, ПРОЕКТУ_ НОМЕР, СПІВРОБІТНИКА_ЗАВДАННЯ).

Можливий ключ:

СПІВРОБІТНИКА_НОМЕР, ПРОЕКТУ_НОМЕР.

Функціональні залежності:

СПІВРОБІТНИКА_НОМЕР, ПРОЕКТУ_НОМЕР -> СПІВРОБІТНИКА_ ЗАВДАННЯ.

Можлива альтернативна декомпозиція, якщо вибрати за основу СПІВРОБІТНИКА_ПРІЗВИЩЕ. В обох випадках отримані відношення СПІВРОБІТНИКИ і СПІВРОБІТНИКИ-ПРОЕКТИ знаходяться в БКНФ, і їм не властиві відзначені аномалії.

Розглянемо відношення *R* (МІСТО, АДРЕСА, ІНДЕКС). Атрибут ІНДЕКС визначає індекс відділення зв'язку, яке обслуговує адресатів деякої вулиці міста, АДРЕСА - назву вулиці і номеру будинку. При цьому будемо припускати, що кортеж *(C, S, Z)* належить деякому відношенню зі схемою відношення R, якщо тільки в місті *C* є будинок за адресою S і Z є відповідним поштовим індексом. У цьому випадку мають місце такі функціональні залежності:

МІСТО, АДРЕСА → ІНДЕКС;

ІНДЕКС → МІСТО.

Іншими словами, повна адреса (назва міста і адреса в місті) визначає поштовий індекс, а поштовий індекс, у свою чергу, визначає назву міста, але не визначає адресу, тому що одне відділення зв'язку обслуговує багато будинків на різних вулицях. Таким чином, в якості основного ключа можна вибрати одне з двох множин атрибутів:

МІСТО, АДРЕСА і АДРЕСА, ІНДЕКС .

Схема відношення *R* (МІСТО, АДРЕСА, ІНДЕКС) не знаходиться в нормальній формі Бойса-Кодда, так як має місце залежність ІНДЕКС→МІСТО. Декомпозицією відношення його можна привести до нормальної форми Бойса-Кодда.

Розглянемо приклад схеми відношення:

ПРОЕКТИ (ПРОЕКТУ_НОМЕР, ПРОЕКТУ_СПІВРОБ, ПРОЕКТУ_ЗАВДАННЯ) .

Відношення ПРОЕКТИ містить номера проектів, кожний проект - список співробітників, які можуть виконувати проект, і список завдань, які передбачаються проектом. Співробітники можуть брати участь у декількох проектах, і різні проекти можуть включати однакові завдання.

Кожний кортеж відношення зв'язує деякий проект із співробітником, які беруть участь у цьому проекті, і з завданням, яке співробітник виконує в рамках даного проекту (припускаємо, що будь-який співробітник, який

бере участь у проекті, виконує всі завдання, передбачені цим проектом). Через сформульовані вище умови єдиним можливим ключем відношення є складовий атрибут ПРОЕКТ_НОМЕР, ПРОЕКТ СПІВРОБ, ПРОЕКТ_ ЗАВДАННЯ, і немає ніяких інших детермінантів. Отже, відношення ПРОЕКТИ знаходиться в БКНФ. Але при цьому воно має аномалії: якщо, наприклад, деякий співробітник приєднується до даного проекту, необхідно вставити у відношення ПРОЕКТИ стільки кортежів, скільки завдань у ньому передбачено.

У відношенні R (A, B, C) існує багатозначна залежність (multi-valued dependence - MVD) (R. A $\rightarrow\!\!\!\rightarrow$ R. B) в тому і тільки в тому випадку, якщо множина значень B, що відповідає парі значень A і C, залежить тільки від A і не залежить від C.

У відношенні ПРОЕКТИ існують дві багатозначні залежності:

ПРОЕКТ_НОМЕР $\rightarrow\!\!\!\rightarrow$ ПРОЕКТ_СПІВРОБ;

ПРОЕКТ_НОМЕР $\rightarrow\!\!\!\rightarrow$ ПРОЕКТУ_ЗАВДАННЯ.

Неважко показати, що в загальному випадку у відношенні R (A, B, C) існує багатозначна залежність A $\rightarrow\!\!\!\rightarrow$B у тому і тільки в тому випадку, коли існує багатозначна залежність A $\rightarrow\!\!\!\rightarrow$ C.

Відношення R знаходиться в четвертій нормальній формі (4НФ) у тому і тільки в тому випадку, якщо у випадку існування багатозначної залежності A $\rightarrow\!\!\!\rightarrow$B всі інші атрибути R функціонально залежать від A.

У нашому прикладі можна виконати декомпозицію відношення ПРОЕКТИ на два відношення ПРОЕКТИ-СПІВРОБІТ і ПРОЕКТИ-ЗАВДАННЯ:

ПРОЕКТИ-СПІВРОБІТ (ПРОЕКТ_НОМЕР, ПРОЕКТ_СПІВРОБІТ);

ПРОЕКТИ-ЗАВДАННЯ (ПРОЕКТ_НОМЕР, ПРОЕКТ_ЗАВДАННЯ).

Обидва ці відношення знаходяться в 4НФ .

Розглянемо ще один приклад. Нехай задано відношення R(СТУДЕНТ, ТОВАРИСТВО, СУСПІЛЬНА_РОБОТА, РІК).

Атрибут ТОВАРИСТВО визначає назву товариств, членом яких є студент; атрибути СУСПІЛЬНА_РОБОТА і РІК - найменування суспільних доручень, виконуваних студентом, і рік їх призначення. Передбачається, що те саме суспільне навантаження не може бути призначене двічі протягом одного року тому ж самому студенту, але заміна одного навантаження на інше протягом року допускається. У табл. 4.16 приведений фрагмент відношення.

Таблиця 4.16 – **Приклад відношення з нетривіальними залежностями**

Студент	Член товариства	Суспільна робота	Рік
Іванов	ВТВР	Староста групи	1999
Іванов	ДТСААФ	Староста групи	1999
Петров	ВТВР	Член інформаційного Прожектору	1999
Петров	ДТСААФ	Член інформаційного "Прожектору"	1999
Петров	ВТВР	Профорг групи	2000
Петров	ДТСААФ	Профорг групи	2000

Виділимо нетривіальні багатозначні залежності:

СТУДЕНТ $\rightarrow\rightarrow$ ТОВАРИСТВО;

СТУДЕНТ $\rightarrow\rightarrow$ (СУСПІЛЬНА_РОБОТА, РІК).

Вважаємо, що атрибут ТОВАРИСТВО не залежить від того, яку суспільну роботу веде студент. Атрибути СУСПІЛЬНА_РОБОТА і РІК взаємозалежні.

Наявність подібних нетривіальних багатозначних залежностей у схемі одного відношення і незалежність їхніх правих частин в остаточному підсумку приводять до комбінації значень правих частин, що ілюструється в табл. 4.16. Відомості про те, що студент Іванов є старостою групи у 1999 р. повторюються двічі в силу того, що він є членом двох товариств (ВТВР,

ДТСААФ). Для студента Петрова в зв'язку зі зміною суспільної роботи (1999 - 2000 р.) доводиться вводити додаткові кортежі, у яких буде повторюватися інформація про членство студента в товариствах ВТВР і ДТСААФ. Незважаючи на надлишковість відношення *R*, що виникає при цьому воно подано в третій нормальній формі, тому що в цьому відношенні відсутні функціональні залежності. З метою усунення надлишковості у відношенні, поданій в третій нормальній формі, необхідно виконати розкладання по багатозначній залежності даного відношення (див.табл.4.17, табл.4.18).

В усіх розглянутих до цього часу нормалізаціях здійснювалась декомпозиція одного відношення на два. Іноді це зробити не вдається, але можлива декомпозиція на більше число відношень, кожне з який має кращі властивості.

Таблиця 4.17 – **Приклад відношення в четвертій нормальній формі**

Студент	Товариство
Іванов	ВТЗВ
Іванов	ДТСААФ
Петров	ВТРВ
Петров	ДТСААФ

Таблиця 4.18 – **Приклад відношення в четвертій нормальній формі**

Студент	Суспільна робота	Рік
Іванов	Староста групи	1999
Петров	Член інформаційного прожектору	1999
Петров	Профорг групи	2000

Розглянемо відношення СПІВРОБІТНИКИ-ВІДДІЛИ-ПРОЕКТИ (СПІВРОБ_НОМЕР, ВІДДІЛУ_НОМЕР, ПРОЕКТУ_НОМЕР).

Припустимо, що той самий співробітник може працювати в декількох відділах і в кожному відділі брати участь у декількох проектах. Первинним ключем цього відношення є повна сукупність його атрибутів. Відсутні функціональні і багатозначні залежності. Тому відношення знаходиться в 4НФ. Однак у ньому можуть існувати аномалії, які можна усунути шляхом декомпозиції на три відношення.

Відношення R (X, Y, ... , Z) задовольняє залежності з'єднання * (X, Y, ... , Z) у тому і тільки в тому випадку, коли R відновлюється без втрат шляхом з'єднання своїх проекцій на X, Y, ... , Z.

Відношення R знаходиться в п'ятій нормальній формі у тому і тільки в тому випадку, коли будь-яка залежність з'єднання в R випливає з існування деякого можливого ключа в R.

Введемо наступні імена складових атрибутів:

З = {СПІВРОБ_НОМЕР, ВІДДІЛУ_НОМЕР};

СП = { СПІВРОБ _НОМЕР, ПРОЕКТУ_НОМЕР};

ВП = {ВІДДІЛУ_НОМЕР, ПРОЕКТУ_НОМЕР}.

Припустимо, що у відношенні СПІВРОБІТНИКИ-ВІДДІЛИ-ПРОЕКТИ існує залежність з'єднання: * (З, СП, ОП)

На прикладах можна легко показати, що при вставках і видаленнях кортежів можуть виникнути проблема. Їх можна усунути шляхом декомпозиції вихідного відношення на три нових відношення:

СПІВРОБІТНИКИ-ВІДДІЛИ (СПІВРОБ_НОМЕР, ВІДДІЛУ_НОМЕР),

СПІВРОБІТНИКИ-ПРОЕКТИ (СПІВРОБ_ НОМЕР, ПРОЕКТУ_НОМЕР),

ВІДДІЛИ-ПРОЕКТИ (ВІДДІЛУ_НОМЕР, ПРОЕКТУ_НОМЕР).

П'ята нормальна форма - це остання нормальна форма, яку можна одержати шляхом декомпозиції.

На закінчення приведемо послідовність етапів нормалізації:

1. Перехід від структурної моделі даних до плоских двовимірних відношень (таблицям).
2. Усунення всіх неповних залежностей атрибутів, які не є основними, від усіх ймовірних ключів.
3. Усунення всіх транзитивних залежностей атрибутів, які не є основними, від усіх ймовірних ключів.
4. Усунення всіх нетривіальних багатозначних залежностей атрибутів, які не є основними, від усіх ймовірних ключів.

Після того як визначені елементи даних і залежності між ними, ці етапи в принципі можуть бути виконані автоматично за приведеним алгоритмом.

Контрольні запитання

1. Для чого виконується нормалізація відношень?
2. Які залежності між атрибутами називають транзитивними та функціонально повними?
3. Приведіть приклади відношень в різних нормальних формах.
4. Приведіть послідовність етапів нормалізації.

5 ЛАБОРАТОРНИЙ ПРАКТИКУМ

Метою проведення лабораторних занять є набуття початкових навичків роботи з реляційними базами даних.

Пропонуємий перелік лабораторних робіт:

Лабораторна робота № 1

Тема: ЗАГАЛЬНА ХАРАКТЕРИСТИКА СУБД.

Мета: придбання первинних навичків роботи із СКБД FOXPRO.

Завдання до лабораторної роботи №1:

1. Вивчити команди створення та запиту бази даних.
2. Підготувати реляційну базу даних, наприклад, список групи, рік народження, національність і т.д.
3. Створити базу даних.
4. Заповнити базу даних. Записів повинно бути не менше 15.
5. Написати звіт про виконання лабораторної роботи.

Лабораторна робота №2

Тема: РЕДАГУВАННЯ ТА ПОШУК ЗАПИСІВ БАЗИ ДАНИХ.

Мета: придбання навичків роботи з базою даних: редагування, пошук, доповнення та видалення записів у базі даних.

Завдання до лабораторної роботи №2:

1. Вивчити команди редагування, перегляду та пошуку записів у базі даних.
2. Підготувати реляційну базу даних.
3. Ввести підготовлену базу даних.
4. Виконати команди редагування.
4. Виконати команди вибіркового перегляду та пошуку записів.
5. Видати результати на друк.
6. Написати звіт про виконання лабораторної роботи.

Лабораторна робота №3

Тема: РОБОТА З БАЗАМИ ДАНИХ

Мета: вивчення команд зміни структури бази даних, сортування та індексування записів у базі даних.

Завдання до лабораторної роботи №3.

1. Вивчити команди, що дозволяють змінювати структуру бази даних, сортувати та індексувати записи у базі даних.
2. Виконати команди зміни структури БД з метою поширити БД.
3. Провести виділення індексних файлів і впорядкувати записи.
4. Вивчити роботу з двома БД, виконати основні команди.
5. Результати видати на друк.
6. Підготувати звіт по лабораторній роботі.

Лабораторна робота №4

Тема: СТВОРЕННЯ ФОРМАТНОГО ФАЙЛА ЗВІТУ.

Мета: набути практичних навичків формування файлів звіту в СКБД FOX-PRO.

Завдання до лабораторної роботи №4:

1. Вивчити команди CREATE REPORT, MODIFY REPORT.
2. Активізувати базу даних.
3. Сформувати звіт за допомогою команди REPORT, змінюючи параметри PLAIN, HEADING, NOEJECT, TO PRINT, TO FILE, SUMMARY.
4. Вивчити команди створення звітних форматів.
5. Результати вивести на друк.
6. Підготувати звіт по лабораторній роботі.

Лабораторна робота №5

Тема: РОБОТА З КОМАНДНИМИ ФАЙЛАМИ.

Мета: ознайомлення з принципами організації командних файлів в СКБД FOXPRO.

Завдання до лабораторної роботи №5:

1. Вивчити команди створення, редагування командного файла, а також виклику й завершення його роботи.
2. Створити командний файл та відредагувати його.
3. Описати змінні різних типів в СКБД FOXPRO.
4. Вивчити і практично реалізувати команди виведення змінних та даних, а також команди виведення текстової інформації.
5. Виконати команди екранного редагування.
6. Вивчити роботу з інформацією в командному файлі і засоби налагоджування.
7. Оформити звіт за результатами виконання лабораторної роботи.

Лабораторна робота №6

Тема: КОМАНДИ СТРУКТУРНОГО ПРОГРАМУВАННЯ.

Мета: засвоєння методів управління обчислювальним процесом в командному файлі з принципами організації командних файлів в СКБД FOXPRO.

Завдання до лабораторної роботи №6:

1. Вивчити команди прийняття рішень, організації циклу та вибору в СКБД FOXPRO.
2. Створити командний файл управління обчислювальним процесом і перевірити його роботу.
3. Організувати розташування коментарів в програмі.
4. Вивчити роботу з інформацією в командному файлі і засоби налагоджування.
5. Оформити звіт за результатами виконання лабораторної роботи.

Лабораторна робота №7

Тема: ДОСЛІДЖЕННЯ ПАРАМЕТРІВ СИСТЕМИ ТА НАЛАГОДЖУВАННЯ КОМАНДНИХ ФАЙЛІВ.

Мета: придбання практичних навичків установки параметрів системи в залежності від постановки задачі в СКБД FOXPRO.

Завдання до лабораторної роботи №7:

1. Вивчити команди установки параметрів системи SET <параметр>.
2. Вивчити команди, що використовуються для налагоджування командних файлів.
3. Встановити параметри системи.
4. Продемонструвати вміння налагоджувати командні файли.
5. Оформити звіт за результатами виконання лабораторної роботи.

Лабораторна робота №8

Тема: РОБОТА З БУДІВЕЛЬНИКОМ МЕНЮ в СКБД FOXPRO.

Мета: надбання практичних навичок проектування власних систем меню в СКБД FOXPRO.

Завдання до лабораторної роботи №8:

1. Набути навички створення нової системи меню.
2. Визначити підказки для пунктів меню.
3. Визначитись з клавішами вибору пунктів меню.
4. Призначити результати пунктам меню.
5. Визначити підказки та клавіші вибору для підменю й призначити результати пунктам підменю.
6. Ознайомитися з командами, що забезпечують зберігання файла меню.
7. Ознайомитися з COMENT OPTIONS.
8. Оформити звіт за результатами виконання лабораторної роботи.

КУРСОВЕ ПРОЕКТУВАННЯ

Метою курсового проектування є глибоке вивчення основних розділів курсу, детальне ознайомлення з основними етапами формування бази даних й поглиблене оволодіння методикою їх проектування.

Зміст курсової роботи повністю визначається завданням, яке видається кожному студенту окремо. Курсове проектування включає такі послідовні етапи:

- аналіз предметної області та постановка задачі;
- розробка універсального відношення;
- розробка ER-моделі предметної області;
- проектування нормалізованих відношень;
- формування початкових відношень за методом "сутність-зв'язок";
- нормалізація відношень методом декомпозиції;
- оцінка спроектованих НФБК - відношень;
- аналіз реалізованих запитів;
- розробка вихідних форм;
- розробка блок-схеми програми, опис складу та призначення програмних файлів;
- вибір та опис тимчасових змінних;
- розробка алгоритмів програмних модулів;
- розробка та відлагодження програми.

В завданні на курсову роботу необхідно вказати номер варіанту, сформулювати завдання, вибрати мову програмування.

Розробка завдання повина задовольняти такі мінімальні вимоги:

- ступінь універсального відношення, не менше 12 ;
- потужність універсального відношення, не менше 12 ;
- кількість "сутей" ER-діаграми, не менше 4;
- кількість попередніх відношень, не менше 4;
- форма нормалізації первиних відношень, не менше 3;

- кількість вихідних форм, не менше 5;
- кількість зреалізовуваних запитів, не менше 5;
- вхідна мова програмування, Fox Base + (або ж будь-яка інша по узгодженню з викладачем).

База даних повинна забезпечувати :

- роботу користувача у режимі кольорового меню;
- введеня, видалення, обновлення інформації;
- видачу відповідей на запити по вибору користувача на екран дисплея, або ж пристрій друку;
- режим підказки для роботи куристувача;
- збереження файлів;
- контроль пріоритетів доступу до окремих блоків даних шляхом накладання пароля.

Запропонований перелік тем курсових робіт

1. Матеріальне забезпечення підприємств.
2. Успішність.
3. Бібліотека.
4. Залізничні каси.
5. Фонотека.
6. Прокат відеофільмів.
7. Деканат.
8. Аеропорт.
9. Перевезення вантажів.
10. Автопідприємство.
11. Регістратура.
12. ЗАГС.
13. Податкова інспекція.
14. Контроль діяльності підприємства.
15. Автовокзал.

16.Читальний зал.

17.Санаторій.

18.Диспансеризація.

19.Кредити.

20.Зарплата.

21.Бізнесмен.

22.Радіотелеательє.

23.Фермер.

24.Олімпійські ігри.

25.Шаховий турнір.

26.Готель.

27.Прокуратура.

28.Рибалка.

29.Постачальник.

30.Військова база.

31.Секретні служби.

32.Облік поточних затрат.

33.Демографія.

34.Облік руху товарів.

35.Продукти харчування.

36.Рецептурний довідник.

37.Контроль оплати по кридитах.

38.Біржові угоди.

39.Посередницька діяльність.

40.Обчислювальна техніка.

41.Автомобілі.

42.Дорожньо-транспортні угоди.

43.Космічні війни.

44.Політичні діячі.

45.Діагностика.

46.Записна книжка.

47.Облік кадрів.

48.Географія Вінницької області.

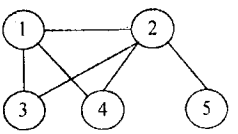
49.Історія Подільского краю.

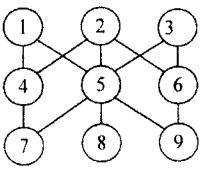
50.Програмні продукти.

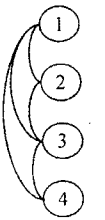
51.Бронювання місць.

ЗАДАЧІ ДЛЯ САМОСТІЙНОЇ РОБОТИ

1. Розробити ієрархічну модель даних факультету.
2. Розробити чотирирівневу ієрархічну модель.
3. Подати мережні структури за допомогою надлишкових ієрархічних структур.







4. Визначити можливі ключі відношень

ЛІКАР (НОМЕР_ЛІКАРЯ, ПРІЗВИЩЕ, ІМ'Я, ПО_БАТЬКОВІ, АДРЕСА, СТАЖ_РОБОТИ, НОМЕР_КАБІНЕТУ, СПЕЦІАЛЬНІСТЬ),

ПАЦІЄНТ (РЕССТРАЦ_НОМЕР, НОМЕР_КОЙКИ, ПРІЗВИЩЕ, ІМ'Я, ПО_БАТЬКОВІ, АДРЕСА, ДАТА_НАРОДЖЕННЯ, СТАТЬ).

5. Розробити відношення, які мають простий, повністю складений та напівскладений ключ.
6. Який тип ключа має відношення

Студент	Викладач	Дисципліна
Іванов	Харитонов	фізика
Петров	Харитонов	фізика
Іванов	Міщенко	інформатика
Петров	Борисов	інформатика

7. Привести приклад зовнішнього ключа.
8. Привести відношення СТУДЕНТ-КУРСОВИЙ_ПРОЕКТ (НОМЕР_ЗАЛІКОВОЇ_КНИЖКИ, КОД_ПРЕДМЕТА, ДАТА_ЗДАЧІ, ПРІЗВИ

ЩЕ_СТУДЕНТА, НОМЕР_ГРУПИ, ВИКЛАДАЧ, ПРОЦЕНТ_ВИКОНАННЯ) до другої нормальної форми.

9. Привести приклади відношень, які не відносять до 3НФ. Провести нормалізацію.
10. Визначити, які основні операції входять до складу операцій ділення та перетину.
11. Які відношення треба використати, щоб в результаті декартового добутку отримати відношення ЕКЗАМЕНАЦІЙНА_ВІДОМІСТЬ.
12. Привести приклад транзитивної та повної функціональної залежності.
13. Виконати нормалізацію відношення СПІВРОБІТНИКИ-ВІДДІЛИ-ПРОЕКТИ (СПІВРОБІТНИКА_НОМЕР, СПІВРОБІТНИКА_ЗАРПЛАТА, ВІДДІЛУ_НОМЕР, ПРОЕКТУ_НОМЕР, СПІВРОБІТНИКА_ЗАВДАННЯ).
14. Виконати нормалізацію відношення ГУРТОЖИТОК (ПІБ_СТУДЕНТА, НОМЕР_ГРУПИ, НОМЕР_КІМНАТИ, СТАРОСТА_КІМНАТИ, РОЗМІРИ_КІМНАТИ, ДАТА_ПОСЕЛЕННЯ).
15. Виконати нормалізацію відношення СПІВРОБІТНИКИ-ПРОЕКТИ (СПІВРОБІТНИКА_НОМЕР, СПІВРОБІТНИКА_ПРІЗВИЩЕ, ПРОЕКТУ_НОМЕР, СПІВРОБІТНИКА_ЗАВДАННЯ).
16. Привести приклади багатозначної залежності.
17. Виконати нормалізацію відношення ПРОЕКТИ (ПРОЕКТУ_НОМЕР, ПРОЕКТУ_СПІВРОБІТНИК, ПРОЕКТУ_ЗАВДАННЯ).
18. Привести приклади відношень, які не належать до 2НФ. Виконати над ними нормалізацію.
19. Визначити тип зв'язку та графічно подати відношення ЧИТАЧ_БІБЛІОТЕКИ-КНИГА.
20. Визначити тип зв'язку та графічно подати відношення ФАКУЛЬТЕТ-СТУДЕНТ.

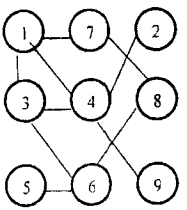
21. Визначити тип зв'язку та графічно подати відношення СТУДЕНТСЬКИЙ_КОЛЕКТИВ-УЧБОВА_ГРУПА.

22. Визначити тип зв'язку та графічно подати відношення ВИКЛАДАЧ-СТУДЕНТ.

23. Визначитись з типом моделі та подати структуру:

А) у вигляді цільного дерева;

Б) у вигляді суми дерев.

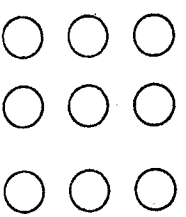


24. Спроектувати ієрархічну базу даних про викладачів, студентів та дисципліни у взаємозв'язку "СТУДЕНТ-ДИСЦИПЛІНА".

25. Визначитись з типом моделі та подати структуру:

А) у вигляді цільного дерева;

Б) у вигляді суми дерев.



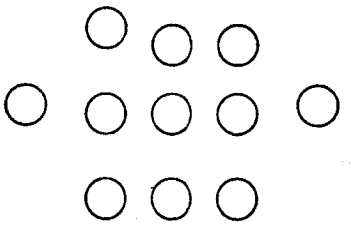
26. Спроектувати ієрархічну базу даних про викладачів, студентів та дисципліни у взаємозв'язку ВИКЛАДАЧ-ДИСЦИПЛІНА.

27. Визначити тип зв'язку та графічно подати відношення між об'єктами АВТОБУС-ПАСАЖИР.

28. Визначитись з типом моделі та подати структуру:

А) у вигляді цільного дерева;

Б) у вигляді суми дерев.



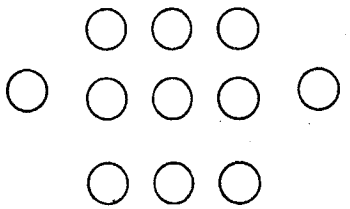
29. Спроектувати ієрархічну базу даних про викладачів, студентів та дисципліни у взаємозв'язку СТУДЕНТ-ДИСЦИПЛІНА.
30. Використовуючи набір для об'єднання записів про студентів однієї групи, визначитись з типом набору, типом запису-володаря, типом записів-членів й типом взаємозв'язку між володарем та членами набору.
31. Визначити тип зв'язку та графічно подати відношення між об'єктами АБОНЕНТ-ТЕЛЕФОН.
32. Спроектувати ієрархічну базу даних про абонентів, їх адреси та телефони у взаємозв'язку АБОНЕНТ-ТЕЛЕФОН.
33. Спроектувати ієрархічну базу даних про абонентів, їх адреси та телефони у взаємозв'язку АДРЕСА-ТЕЛЕФОН.
34. Визначити тип зв'язку та графічно подати відношення між об'єктами Олімпійських ігор КРАЇНА-УЧАСНИК.
35. Використовуючи набір для об'єднання записів про продукцію певної фірми-виробника, визначитись з типом набору, типом запису-володаря, типом записів-членів й типом взаємозв'язку між володарем та членами набору.
36. Використовуючи набір для об'єднання записів про спеціалістів-програмістів банку "АВАЛЬ", визначитись з типом набору, типом

запису-володаря, типом записів-членів й типом взаємозв'язку між володарем та членами набору.

37.Визначитись з типом моделі та подати структуру:

А) у вигляді цільного дерева;

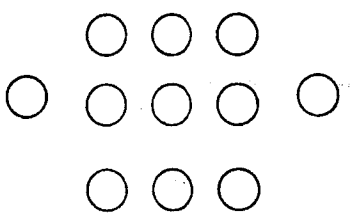
Б) у вигляді суми дерев.



38.Визначитись з типом моделі та подати структуру:

А) у вигляді цільного дерева;

Б) у вигляді суми дерев.



39.Скласти фрагмент програми на мові FOXPRO для БД "ЧИТАЧ", яка містить поля: KOD, NAME, ADRESS, PHONE, WORK; БД "КНИГА", що містить поля: NVM, TITLE, FVTOR; та БД КАРТКА: KOD, NVM, DATE. Вивести в алфавітному порядку всіх читачів, в яких була книга під номером N.

40.Скласти фрагмент програми на мові FOXPRO для БД "ВІДДІЛ_КАДРІВ", що містить поля KOD, NAME, ADRESS, D, PON

Проіндексувати створену БД по полях KOD, NAME. Знайти, використовуючи індексний файл, співробітника з певним прізвищем (NAME).

41. Скласти фрагмент программи на мові FOXPRO для: БД "Відділ кадрів", що містить поля: KOD, NAME, ADRESS, D_BON. Змінити файлову структуру БД, внести нові поля LAST_NAME та W_PHONE; проіндексувати БД за заданими полями: NAME та LAST_NAME. Знайти, використовуючи індексний файл, співробітника з певним прізвищем (NAME).

42. Скласти фрагмент програми на мові FOXPRO для БД "ЧИТАЧ", яка містить поля: KOD, NAME, ADRESS, PHONE, WORK. Скласти запит, який по введеному коду виводить такі значення: NAME ADRESS PHONE.

43. Скласти фрагмент програми на мові FOXPRO для БД "ЧИТАЧ", що містить поля: KOD, NAME, ADRESS, PHONE, WORK; БД "КНИГА", що містить поля: NVM, TITLE, AVTOR; та БД "КАРТКА" з полями: KOD, NVM, DATE. Вивести в алфавітному порядку всіх читачів, які взяли книги у визначений день.

44. Створити БД "ЧИТАЧ", що містить поля: KOD, NAME, ADRESS, PHONE, WORK, БД "КНИГА", що містить поля: NVM, TITLE, AVTOR та БД "КАРТКА", що містить поля: KOD, NVM, DATE. Скласти фрагмент програми на мові FOXPRO, яка виводить в алфавітному порядку всіх читачів, що користувалися книгою під номером N у певний період часу.

45. Визначити наявність функціональної залежності у відношенні СТУДЕНТ (НОМЕР_ГРУПИ, НОМЕР_ЗАЛІКОВОЇ_КНИЖКИ, ПІБ) та проаналізувати його на наявність аномалій.

46. Визначити наявність функціональної залежності у відношенні ПРЕДМЕТ-СТУДЕНТ (ПІБ__СТУДЕНТА, НАЗВА_ДИСЦИПЛІНИ,

ОЦІНКА) та проаналізувати його на наявність аномалій.

47. Визначити декартовий добуток доменів $D1=\{2,3,7,6\}$ та $D1=\{a,w,d,r\}$. Результат подати у векторному та матричному вигляді. Виконати операцію різниці відношень $R(a_r, b_r, c_r)$ та $S(d, a_r, h)$.

48. Визначити в якій нормальній формі знаходиться відношення ГУРТОЖИТОК (ПІБ_СТУДЕНТА, НОМЕР_ГРУПИ, НОМЕР_КІМ- НАТИ, СТАРОСТА_ КІМНАТИ) та проаналізувати його на наявність аномалій.

49. Виявити в якій нормальній формі знаходиться відношення R (МІСТО, АДРЕСА, ІНДЕКС) та проаналізувати його на наявність аномалій.

50. Визначити в якій нормальній формі знаходиться відношення R(ПІБ_СТУДЕНТА, ГРОМАДА, ГРОМАДСЬКА_РОБОТА, РІК) та проаналізувати його на наявність аномалій.

51. Виконати операції реляційної алгебри:

- операцію проекції ПРЕДМЕТ-СТУДЕНТ, що повинна включати лише успішність студента з кожної із вивчених дисциплін, на відношення СТУДЕНТ-УСПІШНІСТЬ (ПІБ_СТУДЕНТА, НАЗВА_ ДИСЦИПЛІНИ, ДАТА_ЗДАЧІ_ІСПИТУ, ОЦІНКА, ЗАЛІК, ПІБ_ВИКЛАДАЧА);
- операцію об'єднання відношень $R(a_r, b_r, c_r)$ та $S(d, a_r, h)$;
- операцію різниці відношень $R(a_r, b_r, c_r)$ та $S(d, a_r, h)$.

52. Визначити декартовий добуток доменів $D1=\{2,1,7,6\}$ та $D1=\{a,w,k,a,r\}$. Результат подати у векторному та матричному виглядах; виконати операцію різниці відношень $R(a_r, b_r, c_r)$ та $S(d, a_r, h)$.

53. Виконати операції реляційної алгебри:

- операцію проекції СТУДЕНТ-УСПІШНІСТЬ на відношення ПРЕДМЕТ-СТУДЕНТ(ПІБ__СТУДЕНТА, НАЗВА_ДИСЦИПЛІНИ, ОЦІНКА);

- операцію об'єднання відношень $R(a_r, b_r)$ та $S(d, a_r, h)$;
- операцію різниці відношень $R(a_r, b_r, c_r)$ та $S(d, a_r, h)$.

54. Визначити в якій нормальній формі знаходиться відношення ПРЕДМЕТ-СТУДЕНТ (ПІБ__СТУДЕНТА, НАЗВА_ДИСЦИПЛІНИ, ОЦІНКА) та проаналізувати його на наявність аномалій.

55. Створити БД "ЧИТАЧ", що містить поля: KOD, NAME, ADRESS, PHONE, WORK), БД "КНИГА", що містить поля: NVM, TITLE, AVTOR та БД "КАРТКА", що містить поля: KOD, NVM, DATE. Скласти фрагмент програми на мові FOXPRO, яка виводить в алфавітному порядку всіх читачів, що користувалися книгами певного автора у певний період часу.

56. Визначити в якій нормальній формі знаходиться відношення *R*(НАЗВА_ФАКУЛЬТЕТУ, ПІБ_ДЕКАНА, ТЕЛЕФОН) та проаналізувати його на наявність аномалій.

57. Визначити в якій нормальній формі знаходиться відношення *R*(ШИФР_ГРУПИ, КІЛЬКІСТЬ_СТУДЕНТІВ, ФАКУЛЬТЕТ, СТАРОСТА, КУРАТОР_ГРУПИ) та проаналізувати його на наявність аномалій.

58. Скласти фрагмент програми на мові FOXPRO для БД "ВІДДІЛ_КАДРІВ", що містить поля KOD, NAME, ADRESS, D_BON. Проіндексувати створену БД по полях KOD, D_BON. Знайти співробітників з певною датою народження (D_BON).

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Атре Ш. Структурный подход к организации баз данных. – М.: Финансы и статистика, 1983. – 320 с.
2. Бойко В.В., Савинков В.М. Проектирование баз данных информационных систем. – М.: Финансы и статистика, 1989. – 351 с.
3. Дейт К. Руководство по реляционной СУБД DB2. – М.: Финансы и статистика, 1988. – 320 с.
4. Джексон Г. Проектирование реляционных баз данных для использования с микроЭВМ. -М.: Мир, 1991. – 252 с.
5. Костин А.Е., Шальгин В.Ф. Организация и обработка структур данных в вычислительных системах Учебное пособие для вузов. -М.: Высш. Шк.,1987.-248 с.
6. Кириллов В.В. Структуризованный язык запросов (SQL). – СПб.: ИТМО, 1994. – 80 с.
7. Мартин Дж. Планирование развития автоматизированных систем. – М.: Финансы и статистика, 1984. – 196 с.
8. Месюра В.І., Романюк О.Н., Денисюк А.В. Методичні вказівки до виконання лабораторних робіт з дисципліни "Організація та управління базами даних для студентів інженерної спеціальності 7.080403" Програмне забезпечення обчислювальної техніки та автоматизованих систем". - Вінниця, :ВДТУ, 1995.
9. Мейер М. Теория реляционных баз данных. – М.: Мир, 1987. – 608 с.
10. Романюк О.Н., Місюра В.І., Денисюк А.В. Методичні вказівки до курсового проектування по дисципліні "Організація та управління базами даних" для студентів інженерії спеціальності 7.080403 "Програмне забезпечення обчислювальної техніки та автоматизованих систем ".- Вінниця, :ВДТУ, 1994.
11. Системы управления базами данных и знаний: Справочное издание / А.Н. Наумов, А.М. Вендров, В.К. Иванов и др.; Под ред. А.Н.Наумова. – М.: Финансы и статистика, 1991. – 352 с.

12. Тиори Т., Фрай Дж. Проектирование структур баз данных. В 2 кн., – М.: Мир, 1985. Кн. 1. – 287 с.: Кн. 2. – 320 с.

13. Ульман Дж. Базы данных на Паскале. – М.: Машиностроение, 1990. – 386 с.

14. Хаббард Дж. Автоматизированное проектирование баз данных. – М.: Мир, 1984. – 294 с.

15. Четвериков В.Н., Ревунков Г.И., Самохвалов Э.Н. Базы и бланки данных. – М.: Высшая школа, 1993.

16. Цикритзис Д., Лоховски Ф. Модели данных. – М.: Финансы и статистика, 1985. – 344 с.

Навчальне видання

Романюк О.Н., Савчук Т.О.

ОРГАНІЗАЦІЯ БАЗ ДАНИХ І ЗНАНЬ

Частина 1

Навчальний посібник

Оригінал-макет підготовлено авторами

Редактор С.А.Малішевська

Підписано до друку 28.02.2001р
Формат 29,7х42 1/4 Гарнітура Times New Roman
Друк різографічний Ум.друк.арк. 7,05
Тираж 75 прим.
Зам.№ 2001-034

Віддруковано в комп'ютерному інформаційно-видавничому центрі Вінницького державного технічного університету
21021, м.Вінниця, Хмельницьке шосе, 95, ВДТУ, ГНК, 9-й поверх
Тел. **(0432) 44-01-59**