МІНІСТЕРСТО ОСВІТИ І НАУКИ УКРАЇНИ
ЦЕНТРАЛЬНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

ОРГАНІЗАЦІЯ БАЗ ДАНИХ

НАВЧАЛЬНИЙ ПОСІБНИК

м. Кропивницький
Видавець. Лисенко В. Ф., 2018

ББК 32.97

Т38

УДК 004

Рекомендовано вченою радою ЦНТУ як навчальний посібник для студентів вищих навчальних закладів
Протокол №8 від 23.04.2018 р.

Рецензенти:

О. А. Смірнов	д.т.н., проф., завідувач кафедри кібербезпеки та програмного забезпечення Центральноукраїнського національного технічного університету;
С. Д. Паращук	к.ф-м.н., доцент, в. о. завідувача кафедри інформатики Центральноукраїнського державного педагогічного університету.

Сидоренко В.В., Константинова Л.В., Смірнов С.А.
Т38 Організація баз даних: Навчальний посібник. – Кропивницький: ЦНТУ, 2018. – 274 с.

Навчальний посібник представляє теоретичний та практичний матеріал про керування та структуру баз даних. Розглянуто введення в інформаційні системи, питання моделювання даних предметної області, теоретичні мови запитів, освітлюються основні прийоми роботи з базами даних з застосуванням мови запитів SQL.

Призначений для студентів, що навчаються за спеціальностями: «Комп'ютерна інженерія» та «Кібербезпека». Для полегшення засвоєння наведеного матеріалу надаються ілюстрації та приклади.

Зміст

Вступ

З швидкими технологічними та соціальними змінами у світі об’єми інформації, які підлягають зберіганню значно зростають. Виробники програмного забезпечення змушені розробляти нові гнучкі підходи для керування великими об’ємами даних. Тому бази даних стають невід’ємною частиною життя сучасної людини.

Цей посібник призначений для студентів, що навчаються за спеціальностями «Комп’ютерна інженерія» та «Комп’ютерні науки» для користувачів та розробників додатків, що ставлять перед собою задачу вивчення систем керування реляційними базами даних та просто людей, які бажають поглибити знання в області баз даних. Тут даються загальні та ключові поняття інформаційних систем. Розглядається моделювання даних, теоретичні мови запитів. Представлена мова реляційного числення за зразком QBE (Query by example). Особливу увагу приділяється структурованій мові запитів SQL. Розглядаються етапи проектування баз даних. Проектування і використання баз даних – розділ, у якому розглядаються проблеми проектування. Це структуризація даних, надмірне дублювання даних, аномалії, нормалізація даних, забезпечення цілісності бази даних. Окремий розділ виділено для огляду семантичного моделювання даних та ER-моделювання даних а також розглянуто приклад побудови ER-моделі. Описано методологію функціонального моделювання. Кожен розділ містить вкінці контрольні запитання та тестові завдання для контролю знань теоретичного та практичного матеріалу, що покращує засвоєння матеріалу.

Розділ 1

Введення в інформаційні системи

- **Загальні поняття**
- **Класифікація інформаційних систем**
- **Користувачі інформаційних систем**
- **Моделі представлення даних**

Інформація – відомості, які передаються людьми в усній, письмовій або іншій формі.

База даних (БД) – сукупність спеціальним чином організованих даних, які зберігаються в пам'яті обчислювальної системи і відображають стан об'єктів та їх взаємозв'язки в даній предметній області.

Система керування базами даних (СКБД) – комплекс мовних і програмних засобів призначених для створення, ведення і сумісного використання БД.

Обчислювальна система – сукупність взаємопов'язаних і погоджено діючих ЕОМ, що забезпечують автоматизацію процесів прийому, обробки і видачі інформації споживачам, а також здійснює контроль за оперативною та зовнішньою пам'яттю, яка потрібна для розв'язку задач.

Інформаційна система (ІС) являє собою систему програмних, мовних, організаційних і технічних засобів призначених для її централізованого накопичення та колективного використання даних.

Автоматизована інформаційна система (АІС) – це інформаційна система, яка працює під управлінням обчислювальної техніки.

Банк даних (БнД) – різновид ІС, в якій реалізовані функції централізованого зберігання та накопичення оброблюваної інформації, що об'єднується в одну або декілька БД.

Словник даних (СД) – це підсистема банку даних призначена для централізованого зберігання інформації про структури даних, взаємозв'язки файлів БД, типи даних, формати їх представлення і коди захисту та розмежування доступу. Функції словника даних виконуються СКБД і викликаються з основного меню системи.

Інформаційна система служить для збирання та накопичення інформації, її ефективного використання для різних цілей. **Основна функція ІС:** моделювання стану об'єктів, частини реального світу, що підлягає автоматизації (ця частина називається предметна область

(ПО)) та відображення зв'язків між об'єктами під час розв'язування функціональних задач.

Під час проектування ІС вирішуються наступні питання:

1. Які відомості та для яких цілей будуть зберігатися в системі;

2. Які дані будуть розташовані в пам’яті комп’ютера, як вони будуть оброблюватись та підтримуватись під час експлуатації.

Класифікація інформаційних систем

Класифікація ІС за ознаками виконуваних функцій:

1. СКБД – організація, збереження та оновлення БД;

2. ІС обробки та накопичення даних, а також засоби реалізації запитів і формування звітів;

3. Програмно-технічні комплекси для автоматизації проектування БД і додатків користувача (їх ще називають CASE – засоби).

Класифікація ІС за типом використання:

1. Локальні ІС (для одного користувача);

2. Мережеві ІС – ІС колективного використання, в яких здійснюється
одночасний доступ до даних кількох користувачів.

Мережеві ІС поділяються на:

- системи «файл-сервер», в яких БД зберігаються на окремому комп’ютері, а обробкою даних та їх поданням займаються прикладні системи;

- системи «клієнт-сервер» - збереження та обробка даних в цих системах виконується на сервері даних та додатків, а прикладні системи виконують тільки функції інтерфейсу між користувачем і середовищем серверу.

Класифікація ІС за типом завдань:

1. Інформаційно-пошукові системи – орієнтовані на пошук даних за вказаним критерієм пошуку.

2. Фактографічні ІС зберігають відомості про об’єкти предметної області.

Основним завданням таких систем є підтримка актуальності БД та видача звітів про поточний стан об’єктів.

3. Системи цільової обробки та аналізу даних. Дані в таку систему зазвичай надходять з датчиків або інших ІС. Основним завданням таких ІС є аналіз цих даних, їх обробка та видача звітів.

4. Документальні ІС, об’єктом зберігання яких є документи, які накопичуються і обробляються системою.

5. Документально-фактографічні системи поєднують в собі характеристики документальної та фактографічної ІС.

Всі користувачі ІС поділяються на дві групи:

- внутрішні (ті, що займаються розробкою ІС);
- кінцеві (ті, для яких створено ІС).

Внутрішні користувачі поділяються на:

- адміністраторів БД;
- адміністраторів функціональних підсистем;
- системних програмістів;
- прикладних програмістів.

Адміністратор БД (АБД) – особа або група осіб, що відповідає за створення вимог до БД, її проектування, розробку та ефективне використання. В мережевих системах АБД взаємодіє з адміністратором мережі, в обов'язки якого входить контроль за апаратно-програмними засобами мережі.

Адміністратори функціональних підсистем, разом з адміністраторами БД, розробляють фільтри для користувачів, крім цього адміністратори функціональних підсистем визначають алгоритми обробки даних, які необхідні для проектування ІС.

Системні програмісти виконують генерацію СКБД, стежать за її функціонуванням у середовищі операційної системи, а також розробляють програми, що розширюють можливості СКБД.

Основною функцією прикладних програмістів є написання модулів (додатків), що потребує знання алгоритмічних мов та мовних засобів СКБД.

Додаток – це програма або комплекс програм, яка забезпечує автоматизацію обробки інформації для деякої прикладної задачі. Додатки бувають зовнішні і внутрішні. Для створення зовнішніх додатків використовуються різні мови програмування, які мають засоби доступу до БД. Внутрішні додатки пишуться на вбудованій мові програмування (В MS Access це Visual Basic).

Модель представлення даних – це логічна структура даних, за якою побудована БД. Існують наступні основні моделі:

1. Ієрархічна модель;
2. Мережева модель;
3. Реляційна модель;
4. Об'єктно-орієнтована модель.

В залежності від моделі представлених даних, СКБД відповідно підрозділяються на: ієрархічні, мережеві, реляційні та об’єктно-орієнтовані.

Контрольні запитання:

1. Дайте визначення, що собою являє: інформаційна система, автоматизована інформаційна система, банк даних, БД, СКБД, предметна область, словник даних.
2. Назвіть основні функції адміністратора БД.
3. Яке призначення СКБД?
4. Назвіть основні моделі даних.
5. Дайте визначення додатку.
6. Вкажіть призначення словника даних.
7. Перерахуйте функції інформаційної системи.
8. Які існують інформаційні системи за типом використання?
9. Які існують інформаційні системи за типом завдань?

Тестові завдання:

1. Інформаційна система - це:

а) Програмний засіб, який вираховує дані і видає необхідну інформацію користувачу;

б) Система, яка забезпечує централізоване накопичення та колективне використання даних;

в) Організовані і оброблені дані;

г) Автоматизована система, призначена виключно для збереження великих об’ємів інформації;

д) Немає правильної відповіді.

2. Що представляє собою база даних?

а)Розрізнені дані і факти;

б)Спеціальним чином організовані дані, які зберігаються в пам’яті та відображають стан об’єктів і їх взаємозв’язки для даної предметної області;

в)Теоретична, не процедурна декларативна мова запитів;

г)Різновидність інформаційної системи, в якій реалізовані функції централізованого зберігання інформації;

д) Немає правильної відповіді.

3. Система керування базами даних – це:

а) Логічна структура бази даних;
б) Програма, яка обслуговує доступ до бази даних клієнтських машин в мережах користувача;
в) Системна таблиця, що об’єднує дані для загального користування;
г) Це комплекс мовних і програмних засобів, призначений для створення, ведення і сумісного використання БД;
д) Немає правильної відповіді.

4. Що розуміють під предметною областю?
а) Набір програм для обробки даних;
б) Область оперативної пам’яті;
в) Системна таблиця, що об’єднує дані для загального користування;
г) Частина реального світу, що підлягає автоматизації;
д) Немає правильної відповіді.

5. Словник даник – це:
а) Підсистема банку для централізованого збереження інформації про структури даних, взаємозв’язки з іншими даними, джерела, застосування і т. д.;
б) Кількість кортежів у таблиці;
в) Логічно зв’язана сукупність даних визначеної довжини;
г) Тупікова ситуація;
д) Немає правильної відповіді.

Розділ 2

Архітектура інформаційних систем

- **Рівні абстракції ІС**
- **Схема взаємодії рівнів подання даних**
- **Життєвий цикл інформаційних систем**
- **Моделі життєвого циклу ІС**

Особливістю ІС є підтримка різноманітних представлень користувачів про ІС. Для кінцевих користувачів ІС – це сховище деяких відомостей. Для внутрішніх користувачів – це елементи даних, записи, сторінки, файли і т.д. Представлення внутрішніх користувачів не є однаковим.

Прикладний програміст оперує з елементами даних, записами, ключами. Системний програміст та адміністратор БД займаються фізичною організацією даних, тобто апробують систему до визначення конкретної задачі.

В сучасних ІС існує декілька рівнів подання даних. Ці рівні можуть відрізнятись найменуванням, але за змістом і своїми функціями вони схожі між собою.

Різновид рівнів подання даних асоціюється з поняттям архітектури ІС.

Під час аналізу та проектування ІС розглядаються наступні рівні абстракції:

1. Локальне подання даних –уявлення кінцевих користувачів про предметну область.

2. Концептуальне подання даних – представляє собою інформаційні потреби системи та відображає особливості предметної області. Концептуальне представлення про ПО існує поза зв’язком із засобами реалізації ІС. Це рівень адміністратора БД та прикладних програмістів.

3. Формалізоване подання даних працює за підтримки СКБД, на якій буде виконуватися дана система і представляє собою логічну організацію даних з погляду адміністратора БД, але на відміну від концептуального рівня здійснюється контроль і прив’язка конкретної СКБД.

4. Внутрішнє подання даних займається фізичним зберіганням даних, це рівень системних програмістів та адміністраторів БД. Цей рівень найбільше впливає на ефективність роботи ІС.

Крім того ці рівніобов’язково діють у взаємозв’язку (рисунок 2.1).

Зовнішній рівень – погляд на предметну область кінцевих користувачів та прикладних програмістів. На цьому рівні формується конкретний опис даних та їх взаємозв'язків.

Інфологічний рівень – відповідає погляду адміністратора на предметну область. На цьому рівні спостерігається вся множина інформаційних об'єктів і зв'язки між ними. Сутність інфологічного моделювання полягає у виділенні інформаційних об'єктів, які підлягають зберіганню в БД, а також у визначенні атрибутів об'єктів і зв'язків між ними.

Концептуальний (датологічний) рівень відповідає уявленню про логічну організацію даних і формується з урахуванням специфіки конкретної СКБД. Цей рівень дуже схожий з інфологічним, але має суттєву відмінність, яка полягає у прив'язці до засобів реалізації в конкретній СКБД. Опис даних на цьому рівні здійснюється мовою опису даних конкретної СКБД.

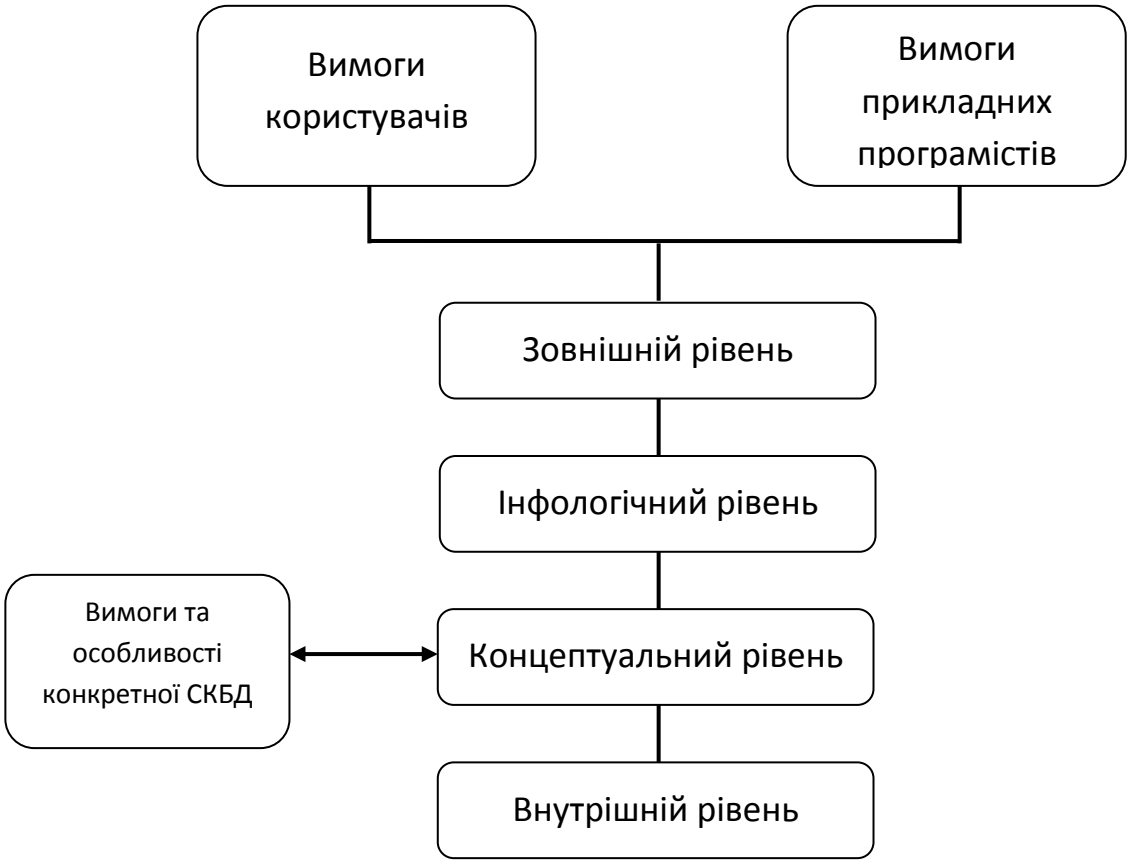


Рисунок 2.1 – Схема взаємозв'язку рівнів подання даних в БД

Внутрішній рівень відповідає представленню і збереженню даних в пам'яті ПЕОМ. Параметри внутрішнього рівня представлення баз даних впливають на ефективність роботи інформаційної системи.

На кожному рівні абстракції визначається своя модель предметної області. Опис цих моделей називається схемами. Три рівні абстракції даних (концептуальний, зовнішній та внутрішній) були запропоновані організацією CODASYL в 1971 році. Американський національний інститут стандартів (ANSI) опублікував цю ідею в 1975 році, як основну в теорії СКБД.

Життєвий цикл інформаційних систем

Як і будь-який інший продукт, ІС проходить певний життєвий цикл, котрий починається з рішення почати створення системи і закінчується припиненням експлуатації внаслідок морального старіння (фізичний знос для ІС є неактуальним). Життєвий цикл ІС містить декілька обов'язкових етапів:

- проектування (визначення та аналіз вимог);
- реалізація (програмний продукт);
- експлуатація та супровід.

Проектування.Проектування виконують за допомогою вивчення ПО та вимог, які ставляться до ІС. На цій «паперовій» стадії життя системи обирають:

- структуру даних і стратегію їх зберігання у пам'яті ЕОМ;
- технологію обслуговування ІС та взаємодію з нею кінцевих користувачів;
- технічні та стандартні програмні засоби, а також розробку оригінальних програмних засобів обслуговування ІС.

Реалізація.На цій стадії розробляють і налагоджують програмне забезпечення ІС, створюють первинну БД, розробляють необхідні програмні додатки. На стадії реалізації перевіряють і коригують технологію обслуговування ІС.

Експлуатація. Починається з наповнення системи реальною інформацією. Експлуатація охоплює весь комплекс дій для підтримки функціонування ІС, тобто: ведення словника-довідника даних, забезпечення захисту даних, організація колективного використання даних, аналіз і керування ефективністю системи тощо.

Крім цього, стадія експлуатації містить у собі розробку нових додатків, а також удосконалення та подальший розвиток ІС.

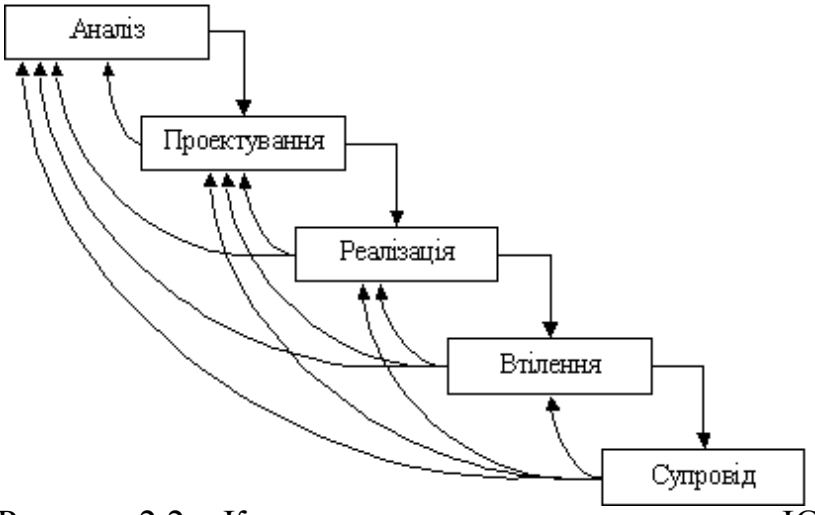


Рисунок 2.2 – Каскадна модель життєвого циклу ІС

Існуючі моделі життєвого циклу відрізняються структурою та конкретним змістом етапів створення та впровадження. Від вибраної моделі життєвого циклу залежить, наскільки довго можна буде підтримувати працездатність ІС. Усі існуючі моделі являють собою спектр, на протилежних кінцях якого знаходяться так звані каскадна та спіральна моделі.

Каскадна модель характеризується суворою впорядкованістю стадій, з яких складаються етапи створення та впровадження. Така впорядкованість потребує досить ретельного виконання робіт, передбачених на кожній стадії, для того, щоб не виникало необхідності переглядати раніше прийняті рішення. На рисунку 2.2 зображено каскадну модель життєвого циклу ІС.

Спіральна модель передбачає багаторазове проходження стадій розробки доти, доки отриманий продукт не буде повністю задовільняти замовника. Ця модель відображає ітераційний характер процесу проектування. На кожній ітерації створюється діючий прототип, який оцінюється, і на підставі цієї оцінки приймають рішення щодо подальшого удосконалення. На рисунку 2.3 зображено спіральну модель життєвого циклу ІС.

Рисунок 2.3 – Спіральна модель життєвого циклу ІС

Перевага каскадної моделі – її детермінованість і чітка регламентація робіт, що спрощує управління проектом. Її недоліком є те, що від затвердження технічного завдання до впровадження готового продукту проходить дуже багато часу. Існує ризик, що до цього часу реальні потреби користувача можуть змінитися і система повністю перестане задовільняти його вимоги. Крім цього, сам користувач у більшості випадків не може правильно сформулювати вимоги, поки сам не почне працювати з системою.

Спіральна модель – навпаки, вільна від цього недоліку, оскільки на кожному витку спіралі є можливість змінити проект з метою досягнення відповідності новим вимогам користувача. До недоліків такої моделі належить практична неможливість планування та контролю виконання проекту, оскільки завчасно невідомо, скільки витків спіралі буде потрібно для отримання остаточної версії системи. Тому не можна точно оцінити затрати на розробку.

Отже, в першому випадку замовник може проконтролювати строки виконання та якість отриманого продукту, визначити затрати та результат. Однак при цьому існує ризик, що створена система хоч і повністю відповідає розробленим спочатку вимогам, не відповідає реальним вимогам користувача. У другому випадку в результаті користувач все ж таки отримує таку систему, яка йому потрібна, але не відомо, скільки на це буде потрібно часу та ресурсів.

Контрольні запитання:

1. Які основні рівні подання даних в ІС?
2. Сформулюйте визначення зовнішнього рівня.
3. Сформулюйте визначення інфологічного рівня.

4. Сформулюйте визначення концептуального рівня.

5. За що відповідає внутрішній рівень представлення даних?

6. Які основні обов'язкові етапи містить життєвий цикл ІС?

7. Охарактеризуйте етап проектування та основні задачі, які вирішуються на цьому етапі.

8. Охарактеризуйте етап реалізації ІС та питання, які вирішуються на цьому етапі.

9. Назвіть основні моделі життєвого циклу ІС.

10. В чому полягає сутність каскадної моделі?

11. В чому полягає сутність спіральної моделі?

Тестові завдання:

1. Бази даних мають такі рівні представлення даних:

а) Концептуальний, внутрішній та зовнішній;

б) Зовнішній, програмний, технічний;

в) Лінгвістичний, технічний, внутрішній;

г) Програмний, зовнішній та периферійний;

д) Немає правильної відповіді.

2. Внутрішній або фізичний рівень представлення даних використовується:

а) Розробником операційної системи;

б) Системним програмістом та адміністратором БД;

в) СКБД для розміщення даних на зовнішніх носіях інформації;

г) Користувачем;

д) Немає правильної відповіді.

3. Логічний або зовнішній рівень використовується:

а) Розробником;

б) Замовником;

в) СКБД;

г) Прикладним програмістом та користувачами;

д) Немає правильної відповіді.

4. Концептуальний рівень це:

а) Користувацьке представлення даних;

б) Логічна схема бази даних;

в) Фізичний вигляд бази даних;

г) Визначає допустимі значення елементів даних;
д) Немає правильної відповіді.

5. *Концептуальне проектування бази даних – це:*
а) Проектування фізичної структури баз;
б) Визначає засоби збереження та використання даних і індексів;
в) Визначення елементів даних, відношень між ними і обмежень;
г) Визначення логічної структури бази даних з урахуванням специфіки конкретної СКБД;
д) Немає правильної відповіді.

Розділ 3

Моделювання даних. Модель «об'єкт-атрибут-зв'язок»

- **Основні види моделей даних**
- **Ієрархічна модель**
- **Мережева модель**
- **Реляційна модель**
- **Поняття ключа та його призначення**

Питання моделювання даних предметної області є ключовими в теорії автоматизованих систем.

Основні види моделей:

а) модель предметної галузі (області);

б) модель даних;

в) модель бази даних.

Існує два основних підходи до створення моделі предметної галузі.

В межах першого підходу модель предметної області будують на основі аналізу і інтеграції інформаційних потреб користувачів банку даних.

Другий підхід базується на аналізі самої предметної області з урахуванням інформаційних потреб користувачів. На основі аналізу цих потреб розробник банку даних створює модель предметної області, і в результаті формалізації отримують два види моделей:

1. Інфологічна модель (концептуальна) зорієнтована на користувача;

2. Датологічна модель зорієнтована на реалізацію в середовищі інформаційної (комп'ютерної) системи.

У банку даних інфологічної моделі предметної області, в середовищі СКБД, виникає одна з можливих датологічних моделей.

Модель даних – це сукупність правил породження структур даних у часі, допустимих операцій над ними та обмежень цілісності, що визначає допустимі зв'язки, значення даних і послідовність їхньої зміни.

У банку даних використовується одна з трьох моделей даних: ієрархічна, мережева або реляційна. Останнім часом набуває поширення використання об'єктно – орієнтованих моделей.

Ієрархічна модель даних

Ієрархічна модель даних базується на принципі субпідпорядкованості між елементами даних і являє собою деревовидну структуру (рисунок 3.1), яка складається з вузлів (сегментів) і дуг (гілок).

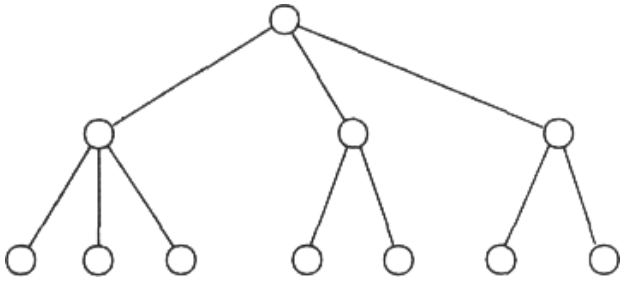


Рисунок 3.1 – Представлення зв'язків в ієрархічній моделі даних

Кожен вузол дерева це набір логічно взаємозв'язаних елементів даних, які описують конкретні об'єкти предметної області. Дерево в ієрархічній моделі впорядковане, тобто існують правила розміщення його вузлів і гілок. Наприклад, структуру організації Обласного управління освітою можна представити за допомогою ієрархічної моделі, схема якої зображена на рисунку 3.2. З прикладу видно, що отримати доступ, наприклад, до елементу «ЗОШ 2» можна лише через елемент «Район 1». Коренем в даному прикладі є «Обласне управління освіти». **Корінь** – основний або верхній вузол.

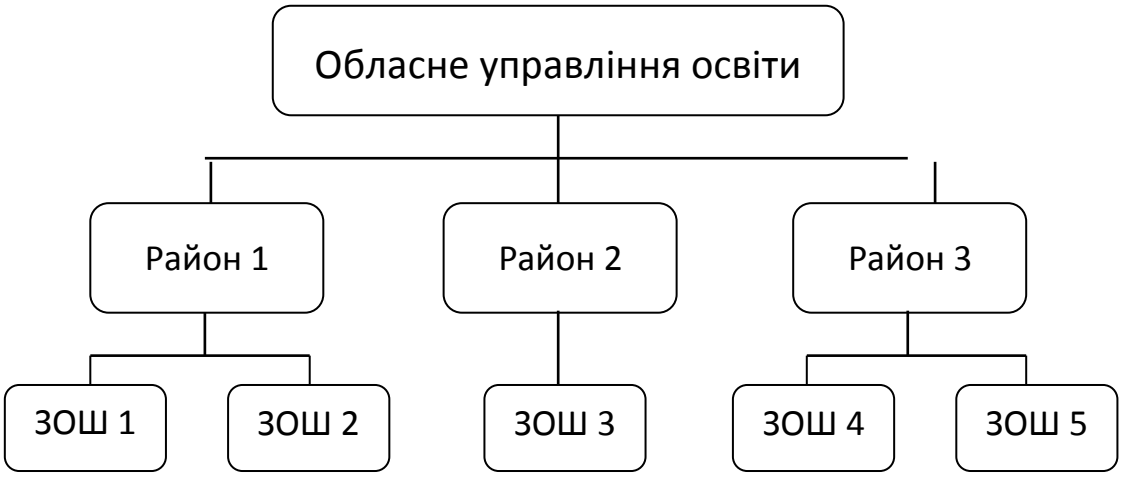


Рисунок 3.2 – Приклад ієрархічної моделі даних

Основні правила побудови ієрархічної моделі:

1. На найвищому рівні міститься вузол, який називається коренем.

2. Взаємозв'язки в ієрархічній моделі даних базуються за принципом «корінний - породжений». Наприклад, вузол 2-го рівня залежить від вузла 1-го рівня і є породженим.

3. Кожен первинний вузол може мати кілька породжених.

4. В ієрархічній моделі даних реалізовано два типи взаємозв’язків: «один до одного» і «один до багатьох».

5. Доступ до кожного вузла відбувається через його породжений вузол.

6. Кожний примірник породженого вузла пов’язаний з вузлом первинним.

7. Примірник породженого вузла не може існувати, якщо відсутній первинний вузол.

8. При знищенні примірника первинного вузла знищуються також пов’язані з ним примірники породжених вузлів.

Мережева модель даних

Мережева модель є розширенням ієрархічного дерева даних. В ній нащадок може мати будь-яку кількість предків і навпаки: предок може мати будь-яку кількість нащадків. У мережевій моделі підтримуються всі три типи взаємозв’язків: «один до одного», «один до багатьох» та «багато до багатьох».

Мережеву модель можна зобразити у вигляді довільного графа, у якого є вершини і ребра (рисунок 3.3). Мережева база даних складається з набору записів і набору відповідних зв’язків. Ніяких обмежень на формування зв’язків не накладається. Якщо в ієрархічній моделі запис-нащадок може мати лише одного предка, то в мережевій моделі даних запис-нащадок може мати довільне число предків.

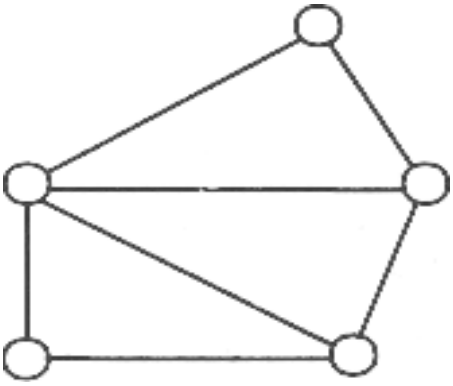


Рисунок 3.3 – Представлення зв’язків в мережевій моделі даних

Переваги мережевої та ієрархічної моделі: ефективність реалізації за показниками пам’яті та оперативності.

Недоліки мережевої та ієрархічної моделі: висока складність і жорсткість схеми бази даних, а також складність для розміщення та розуміння при обробці інформації. Крім того в мережевій моделі слабкий контроль цілісності даних.

Ці дві моделі (мережева та ієрархічна) використовувались раніше на великих та малих ЕОМ. Більш популярною в даний час є реляційна модель даних.

Реляційна модель даних

Реляційна модель даних (РМД) деякої предметної області являється набором відношень, які змінюються в часі. При створенні інформаційної системи сукупність відношень дозволяє зберігати дані про об'єкти предметної області і моделювати зв'язки між ними.

Концепція реляційної моделі даних була запропонована Едгаром Коддом в 1970 році для розв'язування наступної задачі: забезпечити незалежність представлення опису даних від прикладних програм.

В основі РМД лежить поняття «відношення» (relations) подане у вигляді таблиці з дотримуванням деяких обмежувальних умов.

Елементи РМД можна представити у вигляді наступної таблиці:

Таблиця 3.1 – Елементи РМД

Елементи РМД	**Форма представлення**
Відношення	Таблиця
Схема відношення	Рядок заголовків стовпців таблиці
Кортеж	Рядок таблиці
Сутність	Опис властивостей об'єкту
Атрибут	Заголовок стовпця таблиці
Домен	Множина допустимих значень атрибутів
Значення атрибута	Значення поля в записі
Первинний ключ	Один або кілька атрибутів
Тип даних	Тип значень елементів таблиці

Відношення – це найважливіше поняття БД, воно представляє собою двовимірну таблицю, яка містить деякі дані.

Відношення реляційної БД діляться на два класи:

- об'єктні відношення;
- зв'язні відношення.

Об'єктне відношення зберігає дані про об'єкти (екземпляри сутності). В об'єктному відношенні один або кілька атрибутів однозначно ідентифікують об'єкт. Ці атрибути називаються ключем відношення (первинним ключем). В об'єктному відношенні атрибути не повинні дублюватись, це основне обмеження в РМД, що забезпечує цілісність даних.

Зв'язне відношення зберігає ключі двох або більше об'єктних відношень і по цих ключах встановлюються зв'язки між об'єктами відношень.

Сутність – це об'єкт будь-якої природи, дані про який зберігаються в БД. Дані про сутність зберігаються у відношенні.

Атрибут – це властивості, які характеризують сутність . В структурі таблиці кожен елемент іменується і йому відповідає заголовок деякого стовпця таблиці.

Кортеж - представляє собою рядок таблиці (екземпляр таблиці). Порядок кортежів у відношенні невизначений. Порядок атрибутів у відношенні визначений і якщо переставити атрибути, то буде отримано нове відношення.

Домен – множина всіх можливих значень певного атрибута відношення.

Схема відношення (заголовок відношення) – це список імен атрибутів.

Множина кортежів відношення називається тілом відношення.

На рисунку 3.4 зображено таблицю, яка відображає тип об'єкта реального світу, а кожен рядок – конкретний екземпляр об'єкта.

Стовпець таблиці – це сукупність значень конкретного атрибута об'єкта. Ці значення вибираються з множини всіх можливих значень атрибутів об'єкта, яка називається доменом.

Важливим поняттям РМД є поняття ключа. Кожне відношення обов'язково має комбінацію атрибутів, яка може служити ключем. Можливі випадки, коли відношення має кілька комбінацій атрибутів, кожна з яких однозначно визначає всі кортежі відношення і може бути можливим ключем відношення. Будь-який з можливих ключів може бути вибраний як первинний. Якщо вибраний первинний ключ складається з мінімально необхідного набору атрибутів, то він є ненадмірним.

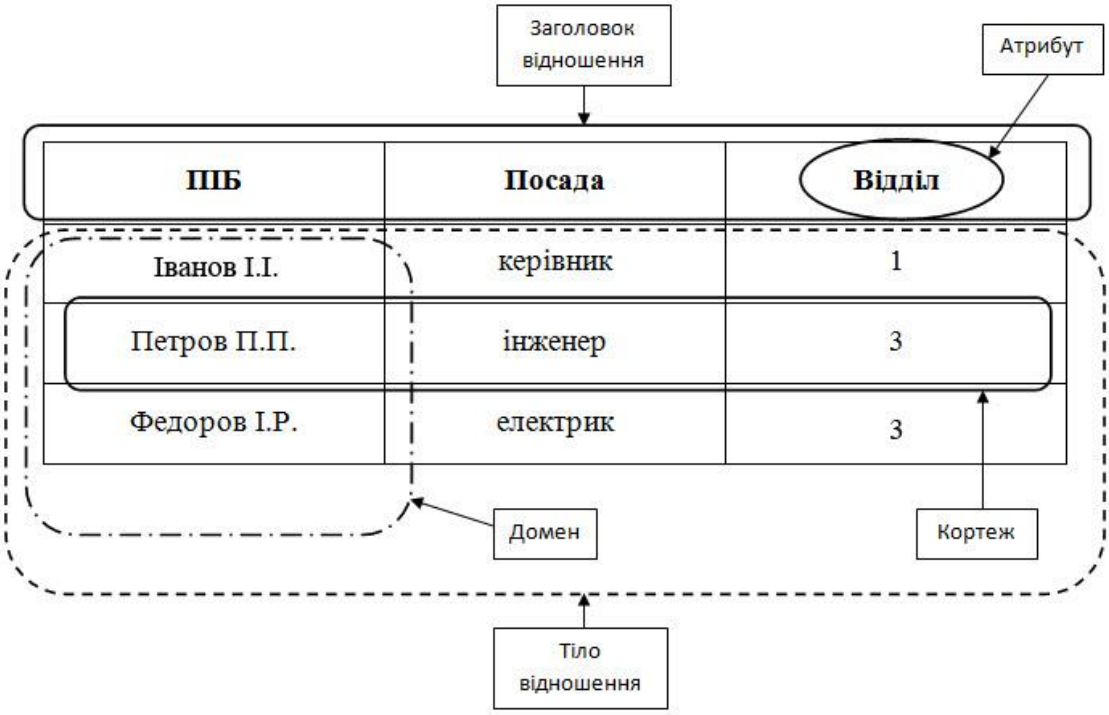


Рисунок 3.4 – Таблиця «Співробітники»

Ключі використовуються для досягнення наступних цілей:

1. Виключення дублювання значень в ключових атрибутах (інші атрибути не враховуються);

2. Впорядкування кортежів, яке можливе за умови збільшення або зменшення значень всіх ключових атрибутів;

3. Прискорення роботи з кортежами відношення;

4. Операції з’єднання таблиць.

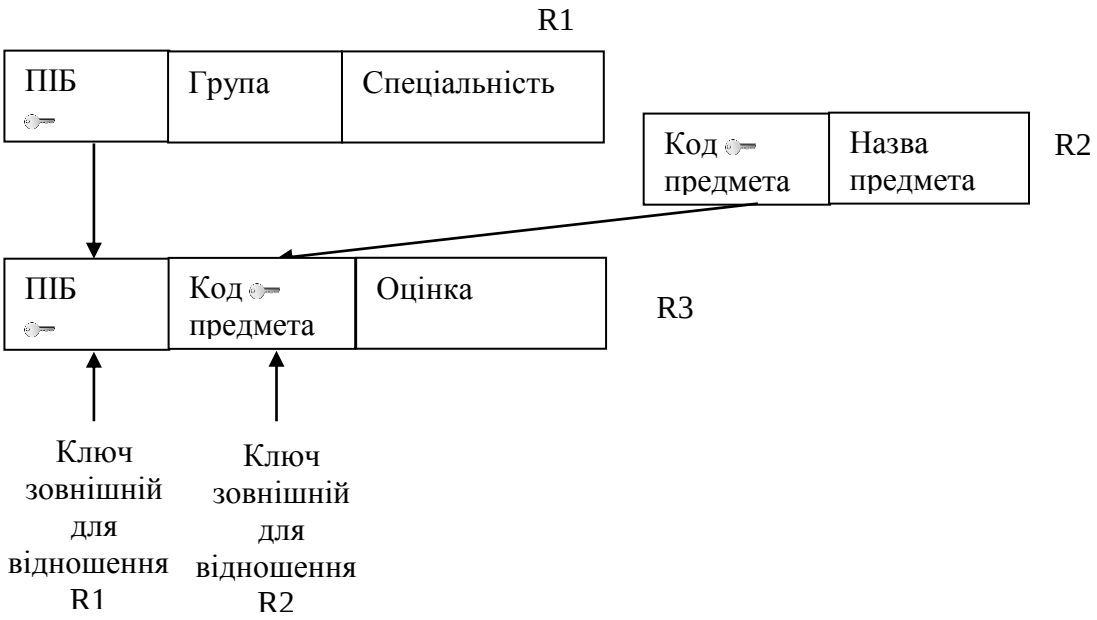


Рисунок 3.5 – Зовнішні ключі

Нехай відношення R1 має не ключовий атрибут А, значення якого є значеннями ключового атрибута іншого відношення, тоді говорять, що атрибут А відношення R1 є зовнішнім ключем (рисунок 3.5). В нашому прикладі є два відношення: R1 з первинним ключем «ПІБ» і R2 з первинним ключем «Код предмета», які зв'язані з відношенням R3, в якому атрибути «ПІБ» і «Код предмета» є зовнішніми ключами по відношенню до відношень R1 і R2 відповідно.

Реляційна модель даних накладає на зовнішні ключі обмеження для забезпечення цілісності даних, яке називається **посилальною цілісністю**. Це означає, що кожному значенню зовнішнього ключа повинні відповідати рядки у зв'язувальних відношеннях. Але не кожній таблиці можна поставити у відповідність відношення.

Таблиця може вважатись відношенням за виконання наступних умов:

1. Не може бути однакових рядків (записів) в таблиці, інакше не може бути однакових первинних ключів.

2. Всі рядки повинні мати однакову типову структуру.

3. Імена стовпців таблиці повинні бути різними, а значення стовпців повинні мати один тип.

4. Значення стовпців повинні бути атомарними (неподільними), не може бути кілька значень в одному стовпці.

5. Повинна зберігатись цілісність для зовнішніх ключів, тобто цілісність посилань.

6. Порядок розміщення рядків в таблиці неістотний, він впливає лише на швидкість доступу до потрібного рядка.

Найчастіше таблиця з відношеннями розміщується в окремому файлі, тоді окрема таблиця являється базою даних. СКБД MS Access являється однією з СКБД, в якій в одному файлі БД зберігаються всі таблиці, а також інші об'єкти бази даних (запити, звіти, форми, макроси, модулі).

Сучасні БД представляються сукупністю взаємопов'язаних таблиць. Тіло типового файлу БД для однієї таблиці містить заголовок таблиці та послідовно організований набір записів.

В заголовку файлу міститься наступна інформація:

- перелік полів з характеристиками (назва, тип, довжина поля);
- час створення або оновлення файлу;
- кількість записів у файлі;
- інша допоміжна інформація.

Контрольні запитання:

1. Назвіть основні види моделей даних.
2. Які основні підходи існують до створення моделі ПО?
3. Що називають моделлю даних?
4. Які моделі даних відносять до класичних?
5. Охарактеризуйте ієрархічну модель даних.
6. Охарактеризуйте мережеву модель даних.
7. Назвіть основні переваги та недоліки ієрархічної та мережевої моделей даних.
8. Назвіть основні правила побудови ієрархічної моделі даних.
9. Дайте визначення реляційної моделі даних і назвіть основні її елементи.
10. Для розв'язання якої задачі Е.Коддом була запропонована реляційна модель даних?
11. Що лежить в основі реляційної моделі даних?
12. Назвіть основні елементи реляційної моделі даних.
13. Що представляє собою «відношення» в реляційній моделі даних?
14. На які два класи діляться відношення?
15. Охарактеризуйте об'єктне та зв'язне відношення.
16. Які основні умови дозволяють називати двовимірну таблицю відношенням?
17. Для яких цілей використовують ключі відношення?
18. Дайте визначення зовнішнього ключа відношення.
19. Сформулюйте поняття посилальної цілісності відношень.

Тестові завдання:

1. Що представляє собою первинний ключ?

а) Кількість кортежів у таблиці;

б) Набір даних любої природи з різними можливостями відображення їх на екрані;

в) Атрибут відношення, однозначно ідентифікуючий кожний з його кортежів;

г) Область пам’яті для обміну даними між додатками;

д) Немає правильної відповіді.

2. Що представляє собою зовнішній ключ?

а) Множина даних, що представляється двовимірною таблицею, яка складається з рядків і стовпців;
б) Кількість однакових кортежів у таблиці;
в) Набір атрибутів однієї таблиці, який є ключем іншої таблиці, використовується для зв’язку даних;
г) Область пам’яті для обміну даними між додатками;
д) Немає правильної відповіді.

3. Ієрархічна модель даних – це:
а) Модель, що підтримує дані у вигляді тривимірної таблиці;
б) Модель, яка відображає різні взаємозв’язки елементів даних у вигляді довільного графа;
в) Модель, де дані відображаються у вигляді «дерева» (впорядкованого графа);
г) Модель, що підтримує дані у вигляді двовимірної таблиці;
д) Немає правильної відповіді.

4. Мережна модель даних – це:
а) Модель, де дані відображаються у вигляді впорядкованого графа;
б) Модель, що відображає різноманітні зв’язки елементів даних у вигляді довільного графа;
в) Модель даних, де дані представлені у вигляді взаємозв’язаних двовимірних таблиць;
г) Модель, де дані не зв’язані між собою;
д) Немає правильної відповіді.

5. Реляційна модель даних –це:
а) Модель, в якій дані представлені у вигляді взаємозв’язаних двовимірних таблиць;
б) Модель підтримки даних у вигляді впорядкованого графа;
в) Модель даних, де дані представлені у вигляді тривимірної таблиці;
г) Модель даних у вигляді дерева;
д) Немає правильної відповіді.

6. Відношення – це:
а) Множина даних, що представляється двовимірною таблицею, яка складається з рядків і стовпців даних;

б) Властивість суттєвості;
в) Послідовність макрокоманд;
г) Логічна структура даних;
д) Немає правильної відповіді.

7. Що називають кортежем?
а) Мову запитів;
б) Деяку комірку таблиці;
в) Стовбець таблиці;
г) Кількість атрибутів;
д) Немає правильної відповіді.

8. Що називають атрибутом?
а) Мову запитів;
б) Рядок таблиці;
в) Назва стовбця таблиці;
г) Кількість рядків;
д) Немає правильної відповіді.

Розділ4

Теоретичні мови запитів

- **Операції над множинами**
- **Операції над відношеннями**
- **Реляційна алгебра Кодда**
- **Приклади виконання операцій реляційної алгебри**

Операції, які виконуються над відношеннями можна розділити на дві групи. Першу групу складають операції над множинами, до яких відносяться наступні операції: об'єднання, перетин, різниця, ділення і декартовий добуток. Другу групу складають спеціальні операції над відношеннями, до яких відносяться такі операції: проекція, з'єднання та вибору.

В різних СКБД реалізована деяка частина операцій над відношеннями, що визначає можливості даної СКБД, а також складність реалізації запитів до БД. В реляційних СКБД для виконання операцій над відношеннями використовують дві групи мов, які мають математичну основу та були запропоновані Едгаром Коддом: реляційна алгебра та реляційне числення. Ці мови представляють мінімальні можливості реальних мов маніпулювання даними, відповідно до реляційної моделі, і еквівалентні між собою за своїми можливостями.

В реляційній алгебрі операнди і результатати всіх дій є відношеннями.

Мови реляційної алгебри є процедурними, оскільки відношення, яке отримується в результаті запиту, обчислюється при виконанніпослідовних реляційних операторів, що застосовуються до відношень. Оператори складаються з операндів, в ролі яких виступають відношення реляційних операцій.

Мови реляційного числення, на відміну від мов реляційної алгебри, є не процедурними і дозволяють виражати запити за допомогою предиката першого порядку (вислови у вигляді функцій), якому повинні задовольняти кортежі або домени відношень. Запит до БД, з використанням подібної мови, містить лише інформацію про бажаний результат. Існує набір правил для запису запитів. Як приклад мови реляційного числення можна привести мову SQL.

Реляційна алгебра

Прикладом мови заснованої на реляційній алгебрі є мова ISBL (Information Systems Base Language) – це базова мова інформаційних систем.

Мови запитів побудовані на основі реляційної алгебри не отримали широкого розповсюдження в сучасних СКБД, але знайомство з ними корисне для розуміння суті реляційних операцій.

Ступінь відношення – це кількість атрибутів, які в нього входять (або кількість стовпців у таблиці).

Потужність відношення – кількість записів у таблиці відношень (або кількість рядків без заголовку в таблиці).

Варіант реляційної алгебри запропонований Коддом включає наступні основні операції: об'єднання, різниця (віднімання), перетин, декартовий добуток, вибірка (селекція і обмеження), проекція, розподіл і з'єднання.

По зауваженню Дейта, ці операції мають кілька недоліків. **По-перше**, вісім перерахованих операцій з одного боку надмірні за своїми функціями, оскільки мінімально необхідний набір складає тільки п'ять операцій: об'єднання, віднімання, добуток, проекція і вибірка. Три інші операції (перетин, з'єднання, розподіл) можна визначити через п'ять мінімально необхідних. **По-друге -** цих восьми операцій не досить для побудови реальної СКБД за принципами реляційної алгебри. Потрібні розширення, які включають наступні операції: перейменування атрибутів, утворення нових обчислювальних атрибутів, обчислення підсумкових функцій, побудова складних виразів алгебри, присвоєння, порівняння та інші.

Операції реляційної алгебри Кодда

Операції реляційної алгебри Кодда можна розділити на дві групи:

- базові теоретико-множинні операції;
- спеціально-реляційні операції.

Перша група операцій це класичні операції теорії множин (об'єднання, різниця, перетин, добуток), друга – операції, які є розвитком теоретико-множинних операцій: проекція, селекція, розподіл і з'єднання.

Операції реляційної алгебри можуть виконуватись над одним відношенням (унарні операції), наприклад проекція, або над двома відношеннями (бінарні операції), в цьому випадку відношення, що беруть участь в бінарній операції повинні бути сумісні за структурою.

Сумісність структури відношень означає сумісність атрибутів і типів відповідних доменів.Структура результуючого відношення за певними правилами успадковує властивості структур початкових відношень. У більшості бінарних операцій заголовки початкових відношень ідентичні, що не виключає проблеми з заголовками результуючого відношення.

Об'єднанням двох сумісних відношень ($R_1 \cup R_2$ або R_1 UNION R_2) є відношення R, яке містить всі елементи початкових відношень (виключаючи повторення).

Різницею двох сумісних відношень ($R_1 - R_2$ або R_1MINUSR_2) однакової розмірності є відношення, тіло якого складається з множини кортежів, що належать відношенню R_1, але не належать відношенню R_2.

Перетин двох сумісних відношень ($R_1 \cap R_2$ або R_1INTERSECTR_2) однакової розмірності породжує відношення R з тілом, що включає кортежі, які одночасно належать обом відношенням.

Добутком відношення R_1 в ступені k_1 і відношення R_2 в ступені k_2 ($R_1 \times R_2$ або R_1TIMESR_2), ступінь відношення це кількість атрибутів у відношення, які не мають однакових імен атрибутів, є відношення R в ступені $k_1 + k_2$, заголовок якого представляє собою об'єднання заголовків відношення R_1і R_2, а тіло має такі кортежі, що перші k_1 елементів кортежів належать множині R_1, а останні k_2 елементів – множині R_2.

Вибірка відношення R по формулі F (RWHEREF) є відношення з таким же заголовком і тілом, що складається з таких кортежів відношення R, які задовільняють істинності логічного виразу заданого формулою F. Для запису формули F використовують операнди (імена атрибутів або номери стовпців), константи, логічні операції порівняння і дужки.

Проекція відношення A на атрибути x,y,…,z (A[x,y,…,z]), де множина {x,y,…,z } є підмножиною повного списку атрибутів заголовка відношення A є відношення із заголовком x,y,…,z і тілом, що містить кортежі відношення A, за винятком кортежів, які повторюються. Повторення однакових кортежів атрибутів у списку x,y,…,z забороняється. **Операція проекції допускає наступні додаткові варіанти запису:**

1. Відсутність списку атрибутів має на увазі вказівку всіх атрибутів (операція тотожної проекції).

2. Вираз вигляду R[] означає порожню проекцію, результатом якої є порожня множина.

3. Операція проекції може застосовуватись до довільного відношення, в тому числі і до результату вибірки.

Результатом операції розподілу відношення R_1 з атрибутами a і b (R_1(a,b)) на відношення R_2 з атрибутом b (R_2(b)), де ai b – прості або складні атрибути, причому атрибут b – загальний атрибут обох відношень визначений на одному й тому ж домені, є відношення із заголовком *a* і тілом, що складається з кортежівr, s, таких, що в відношенні R1 є кортежі (r, s), причому множина значень s включає множину значень атрибута bвідношенняR2.

Операція з'єднання відношень R_1 і R_2 ((R_1TIMESR_2) WHEREF), за умовою, що задається формулою F, є відношення R, яке можна отримати шляхом декартового добутку R_1і R_2 з подальшим застосуванням до результату операції вибірки за формулою F, правила запису формули такі ж як для операції вибірки.

Приклади виконання операцій реляційної алгебри

Операція об'єднання: R1 union R2 (R1 + R2).

Операція об'єднання проводиться над двома сумісними відношеннями. Результуюче відношення включає всі записи першого відношення і ті записи другого відношення, яких немає в першому. Наприклад, розглянемо такі відношення (рисунок 4.1):

Відношення 1:

Прізвище	Вік
Ананатійчук Р.І.	30
Бас І.М.	25
Білань І.М	24
Вільховська С.О.	32

Відношення 2:

Прізвище	Вік
Ананатійчук Р.І.	30
Вільховська С.О.	32
Гамар О.М.	25

Результуюче відношення:

Прізвище	Вік
Ананатійчук Р.І.	30
Бас І.М.	25
Білань І.М	24
Вільховська С.О.	32
Гамар О.М.	25

Рисунок 4.1

Операція перетину: R1 intersect_with R2.

Перетин виконується над двома відношеннями. Результуюче відношення містить тільки ті записи, які є одночасно в першому і другому відношеннях. Наприклад, розглянемо такі відношення і виконаємо перетин (рисунок 4.2):

Відношення 1:

Прізвище	Вік
Ананатійчук Р.І.	30
Бас І.М.	25
Білань І.М	24
Вільховська С.О.	32

Відношення 2:

Прізвище	Вік
Ананатійчук Р.І.	30
Вільховська С.О.	32
Гамар О.М.	25

Результуюче відношення:

Прізвище	Вік
Ананатійчук Р.І.	30
Вільховська С.О.	32

Рисунок 4.2

Операція різниці: R1 without R2 (R1 – R2).

Операція різниці проводиться над двома відношеннями. Результуюче відношення містить ті записи першого відношення, яких немає в другому відношенні. Наприклад, розглянемо операцію різниці на таких відношеннях (рисунок 4.3):

Відношення 1:

Прізвище	Вік
Ананатійчук Р.І.	30
Бас І.М.	25
Білань І.М	24
Вільховська С.О.	32

Відношення 2:

Прізвище	Вік
Ананатійчук Р.І.	30
Вільховська С.О.	32
Гамар О.М.	25

Результуюче відношення:

Прізвище	Вік
Бас І.М.	25
Білань І.М	24

Рисунок 4.3

Операція декартового добутку: R1 produced_withR2 (R1 * R2).

Декартовий добуток виконується над двома відношеннями, ступінь результуючого відношення дорівнює сумі ступенів первинних відношень, а потужність рівна добутку їх потужностей. Результуюче відношення містить всі можливі комбінації в записі первинних відношень. Наприклад (рисунок 4.4):

Відношення 1:

Прізвище
Гасюк У.І.
Добровольська І.В.
Дробенко Ю.Г

Відношення 2:

Предмет	**Дата екзамену**
СКБД ПК	09.01.15
Історія	14.01.14

Результуюче відношення:

Прізвище	**Предмет**	**Дата екзамену**
Гасюк У.І.	СКБД ПК	09.01.15
Гасюк У.І.	Історія	14.01.14
Добровольська О.В.	СКБД ПК	09.01.15
Добровольська О.В.	Історія	14.01.14
Дробенко Ю.Г.	СКБД ПК	09.01.15
Дробенко Ю.Г.	Історія	14.01.14

Рисунок 4.4

Операція ділення: R1 divideby R2.

Операція ділення – відношення дільника повинно містити підмножину атрибутів відношення діленого. Результуюче відношення включає тільки ті записи декартового добутку результуючого відношення з дільником, які містяться в діленому. Крім того, результуюче відношення містить тільки ті відношення діленого, яких немає в дільнику. Наприклад, розглянемо операцію ділення на таких відношеннях (рисунок 4.5):

Відношення 1 (ділене):

Прізвище	Предмет	Дата екзамену
Гасюк У.І.	СКБД ПК	09.01.15
Гасюк У.І.	Історія	14.01.14
Добровольська О.В.	СКБД ПК	09.01.15
Добровольська О.В.	Історія	14.01.14
Дробенко Ю.Г.	СКБД ПК	09.01.15
Дробенко Ю.Г.	Історія	14.01.14

Відношення 2 (дільник):

Предмет	Дата екзамену
СКБД ПК	09.01.15
Історія	14.01.14

Результуюче відношення:

Прізвище
Гасюк У.І.
Добровольська О.В.
Дробенко Ю.Г.

Рисунок 4.5

Операція проекції.

Операція проекції виконується над одним відношенням. Результуюче відношення включає частину атрибутів вихідного, на які виконується проекція. Наприклад, для відношення 1 знайдемо перелік посад для кожного відділу (рисунок 4.6).

Відношення 1:

Прізвище	Номер відділу	Посада
Ткаченко О.В.	1	Інженер
Хороз Н.Б.	1	Інженер
Рапій І.М.	2	Інженер
Сколоздра С.Т.	2	Технік

Результуюче відношення:

Номер відділу	Посада
1	Інженер
2	Інженер
2	Технік

Рисунок 4.6

Операція з'єднання:Cf (R1, R2).

Операція з’єднання виконується над двома відношеннями. В кожному відношенні повинні знаходитись як мінімум один, можливо і більше, атрибутів, що співпадають. Результуюче відношення включає всі атрибути першого і другого відношень. Наприклад, для відношення 1 і 2 будемо мати (рисунок 4.7):

Відношення 1:

Спеціальність	Код студента
Менеджмент	2
Економіка	3
Історія	8

Відношення 2 :

Код студента	Прізвище	Курс
1	Кусий О.А.	2
2	Єлісєєнко О.С.	1
3	Кухар Н.Є.	1
8	Стоцько О.О.	3

Результуюче відношення:

Спеціальність	Код студента	Прізвище	Курс
Менеджмент	2	Єлісєєнко О.С.	1
Економіка	3	Кухар Н.Є.	1
Історія	8	Стоцько О.О.	3

Рисунок 4.7

Операція вибору: R where f.

Операція вибору проводиться над одним відношенням. Результуюче відношення містить тільки ті записи, які відповідають певній умові з даного атрибуту. Наприклад, проведемо вибірку для відношення по ознаці «Ріст більший 165 сантиметрів» (рисунок 4.8):

Відношення 1:

Прізвище	**Ріст**
Сало Є.В.	185
Ткачук Ю.В.	165
Вільховська С.О.	173
Гамар О.І.	165

Результуюче відношення:

Прізвище	**Ріст**
Сало Є.В.	185
Вільховська С.О.	173

Рисунок 4.8

Розглянуті операції дозволяють виділяти із відношень їх підмножину, знову ж об’єднувати ці підмножини в ємніші відношення, поновлювати вміст відношень і представляти їх в потрібному вигляді.

Контрольні запитання:

1. Дайте загальну характеристику теоретичних мов запитів.
2. Назвіть операції реляційної алгебри запропонованої Коддом.
3. На які дві групи діляться операції реляційної алгебри Кодда?
4. Що означає сумісність структур відношень?
5. Що є результатом всіх операцій реляційної алгебри?
6. Що є результатом об’єднання двох відношень?
7. Над якими відношеннями можна проводити операції об’єднання, віднімання, перетину та добутку?
8. Сформулюйте визначення проекції деякого відношення на його атрибути.
9. Мова реляційної алгебри являється процедурною чи описовою?
10. З використанням якої мови запит до БД містить лише інформацію про бажаний результат?

Тестові завдання:

1. Що представляє собою поняття реляційна алгебра?

а) Об’єкт бази даних;

б) Об’єктно-орієнтована мова;
в) Послідовність макрокоманд;
г) Теоретична процедурна мова запитів;
д) Немає правильної відповіді.

2. Що представляє собою поняття реляційне числення?
а) Теоретична декларативна мова запитів;
б) Теоретична процедурна мова запитів;
в) Програма обробки даних;
г) Послідовність макрокоманд;
д) Немає правильної відповіді.

3. Операції реляційної алгебри Кодда можна розділити на дві групи:
а) Таблиці і макроси;
б) Макроси і спеціальні операції;
в)базові теоретичні звіти і множинні операції;
г)базові теоретико-множинні операції і спеціально-реліційні операції;
д) Немає правильної відповіді.

4. Які операції відносять до базових теоретико-множинних операцій?
а) Програмування на мові PHP;
б) Програмування на мові С;
в) Проекція, селекція, розподіл і з’єднання;
г) Об’єднання, різниця, перетин, добуток;
д) Немає правильної відповіді.

5.Які операції відносять до спеціально-реляційних операцій?
а) Приховати стовпці таблиці;
б) Показати приховані стовпці таблиці;
в) Проекція, селекція, розподіл і з’єднання;
г) Об’єднання, різниця, перетин, добуток;
д) Немає правильної відповіді.

6.Врезультаті сортування таблиць дані:
а) Вилучаються;
б) Доповнюються;

в) Змінюється їх структура;
г) Дані впорядковуються за певним критерієм;
д) Немає правильної відповіді.

Розділ 5

Додаткові операції реляційної алгебри запропоновані Дейтом

- **Додаткові операції реляційної алгебри**
- **Операції реляційного числення**
- **Правильно побудовані формули (Well Formed Formula)**
- **Реляційне обчислення доменів**

Дейтом були запропоновані наступні додаткові операції реляційної алгебри: перейменування даних, розширення, підведення підсумків, присвоєння, вставки, оновлення даних, видалення даних, реляційного порівняння. Розглянемо кожну з цих операцій детально.

Операція перейменування даних

RENAME <початкове відношення> <старе ім'я атрибута> AS <нове ім'я атрибута>

Ця операція дає змогу перейменувати атрибути. Початкове відношення задається ім'ям відношення або виразом реляційної алгебри, який заключається в круглі дужки.

Операція множинного перейменування атрибутів

RENAME <відношення> <старе ім'я атрибута 1> AS <нове ім'я атрибута 1>, <старе ім'я атрибута 2> AS <нове ім'я атрибута 2>, …

Операція розширення

EXTEND<початкове відношення>ADD<вираз>AS<новий атрибут>

Операція розширення породжує нове відношення схоже на початкове, яке відрізняється наявністю доданого атрибута, значення якого знаходять шляхом деяких скалярних обчислень. Початкове відношення може бути задано ім'ям відношення або за допомогою виразу реляційної алгебри, що заключається в круглі дужки. При цьому ім'я нового атрибута не повинно входити в заголовок початкового відношення і не може використовуватись у «виразі». Крім звичайних арифметичних операцій порівняння у виразі використовуються підсумкові функції, такі як: COUNT (кількість) SUM (сума)AVG (середнє арифметичне)MAX (максимальне значення)MIN (мінімальне значення).

Користуючись операцією розширення інколи виконують перейменування атрибутів. Для цього у виразі потрібно вказати ім'я атрибута, а в конструкції AS визначити нове ім'я цього атрибута, а

потім виконати проекцію отриманого відношення на множину атрибутів, включаючи старий атрибут.

Операція множинного розширення

EXTEND <відношення> ADD <вираз 1> AS <новий атрибут 1>, <вираз 2>AS<новий атрибут 2>, …

Ця операція є подібною до операції розширення, вона дозволяє в одній синтаксичній конструкції обчислювати кілька нових атрибутів.

Операція підведення підсумків

SUMMARIZE<початкове відношення>ADD<вираз>AS<новий атрибут>

Дана операція виконує вертикальні або групові обчислення. Початкове відношення задається ім'ям відношення або виразом реляційної алгебри в круглих дужках. Результатом операції SUMMARIZE є відношення R з заголовком, що складається з атрибутів списку розширеного новим атрибутом. Для отримання тіла відношення R спочатку проводиться проектування початкового відношення на атрибути, а після цього кожний кортеж відношення розширюється новим N+1 атрибутом. Оскільки проектування приводить до скорочення кількості кортежів, бо видаляються однакові кортежі, то вважають, що відбувається своєрідне групування кортежів початкового відношення.

Значення N+1 атрибуту кожного кортежу відношення R формується шляхом обчислення виразу над відповідною цьому кортежу групою кортежів початкового відношення. Функція COUNT визначає кількість кортежів в кожній із груп початкового відношення.

Операція множинного підведення підсумків

Операція множинного підведення підсумків подібна відповідним операціям перейменування і розширення. Вона виконує одночасно кілька вертикальних обчислень і записує результати в окремі нові атрибути.

Операція присвоєння

<вираз-ціль> := <вираз-джерело>

Операція присвоєння являється основною операцією, що змінює тіло існуючого відношення, крім неї змінюють тіло існуючого відношення операції вставки, оновлення та видалення. В операції присвоєння зліва вказане ім'я відношення, а праворуч – деякий вираз реляційної алгебри, при цьому ці вирази повинні бути сумісні за структурами і задавати сумісні по структурі відношення.

Виконання операції присвоєння зводиться до заміни попереднього значення на нове. За допомогою цієї операції можна здійснювати додавання та видалення кортежів.

Операція вставки

INSERT<вираз-джерело>INTO<вираз-ціль>

Обидва вирази повинні бути сумісні по структурі. Виконання операції зводиться до обчислення виразу джерела і вставки отриманих кортежів в задане відношення вираз-ціль.

Операція оновлення

UPDATE <вираз-ціль> <список елементів>

Де список елементів є послідовністю розділених комами операцій присвоєння, що мають наступний вигляд:

<атрибут> := <скалярний вираз>

Результатом виконання цієї операції є відношення отримане після присвоєння відповідних значень атрибутам відношення, що задаються цільовим виразом.

Операція видалення

DELETE <вираз-ціль>

Тут вираз-ціль є реляційним виразом, що описує кортежі, які видаляються.

Операція реляційного порівняння

<вираз 1> 0 <вираз 2>

Тут обидва вирази задають сумісні по структурі відношення, а знак 0 – одну з наступних операцій: =, ≠, <, >.

Операції реляційного числення

До операцій реляційного числення відносяться операції, в яких запити до існуючої реляційної СКБД записуються за допомогою властивостей потрібного відношення, без конкретизації процедури його отримання.

Принципова відмінність між реляційною алгеброю і реляційним численням полягає в тому, що в першому випадку процес отримання результату описується явним чином, шляхом вказівки набору операцій, які треба виконати для отримання результату. В другому ж випадку ми вказуємо, що ми хочемо отримати і звідки це взяти, без конкретизації процедури отримання.

Ззовні підходи сильно відрізняються: один з них приписуючий (реляційна алгебра), а інший описовий (реляційне числення), але на більш низькому рівні підходи еквівалентні. Це зумовлено тим, що будь-

які вирази реляційної алгебри можуть бути перетворені в семантичний еквівалент виразів реляційного числення і навпаки. Для цих перетворень використовують алгоритм редукції Кодда або алгоритми інших авторів.

Припустимо, що ми працюємо з БД, яка визначається схемами:

СПІВРОБІТНИКИ(співробітник_номер (ключ), співробітник_ім'я, співробітник_зарплата, співробітник_номер_відділу);

ВІДДІЛ (відділ_номер (ключ), відділ_кількість, відділ_начальник).

Реляційне обчислення є частиною формального механізму обчислення предикатів першого порядку. **Предикати** – це вислови у вигляді функцій.

Базовими поняттями обчислення є поняття змінної з деякою областю допустимих значень і поняття правильно побудованої формули, що спирається на змінні, предикати і квантори. Залежно від того, що є областю визначення розрізняють обчислення кортежів і обчислення доменів.

В обчисленні кортежів областями визначення змінних є відношення БД, тобто допустимими значеннями кожної змінної є кортежі деякого відношення.

В обчисленні доменів областями визначення змінних є домени, на яких визначені атрибути БД, тобто допустимими значеннями кожної змінної є значення деякого домену.

В обчисленні кортежів для визначення кортежної змінної використовується оператор RANGE. Наприклад, для того, щоб визначити змінну СПІВРОБІТНИК, областю визначення якої є відношення СПІВРОБІТНИКИ, потрібно використати наступну конструкцію:

RANGE СПІВРОБІТНИК IS СПІВРОБІТНИКИ

З цього визначення випливає, що у будь-який момент часу змінна СПІВРОБІТНИК представляє деякий кортеж відношення СПІВРОБІТНИКИ. Під час використання кортежних змінних у формулах можна посилатися на значення атрибута змінної (аналогічно структурам у мові СІ). Наприклад, для того щоб зіслатись на значення атрибута СПІВРОБІТНИК_ІМ'Я, змінної СПІВРОБІТНИК, використовується конструкція:

СПІВРОБІТНИК **.** СПІВРОБІТНИК_ІМ'Я

Правильно побудовані формули (WFF – WellFormedFormula)

WFF використовуються для виразу умов, що накладаються на кортежні змінні. Основою WFF є прості порівняння (comporison), які являють собою операції порівняння скалярних значень (значень атрибутів змінних, або літерально заданих констант). Наприклад, конструкція:

СПІВРОБІТНИК.СПІВРОБІТНИК_НОМЕР = 140

є простим порівнянням, тобто вона і є WFF. Більш складні варіанти WFF будуються за допомогою логічних зв'язків, з використанням логічних операцій: NOT, AND, OR, IF…THEN.

Якщо Form – правильно побудована формула, а Comp – просте порівняння, то NOT Form; Comp AND Form; Comp OR Form і IF Comp THEN Form теж являються WFF.Крім цього допускається побудова WFF за допомогою кванторів. Якщо Form – правильно побудована формула, в якій бере участь змінна VAR, то конструкції:

EXISTSVAR (Form) – квантор існування

FORRALVAR (Form) – квантор загальності

являтимуться теж правильно побудованими формулами.

Змінні, які входять в WFF можуть бути вільними або зв'язаними. Всі змінні, що входять до WFF, для побудови якої не використовувались квантори є вільними. Це означає, що якщо для якого-небудь набору значень вільних кортежних змінних після обчислення WFF отримано значення true, то ці значення кортежних змінних можуть входити у відношення-результат. Якщо ж ім'я змінної використовувалось відразу після квантора, під час побудови WFF вигляду EXISTSVAR (Form) або FORRALVAR (Form), то в цій WFF і в усіх WFF побудованих за її участю, VAR являється зв'язною змінною. Це означає, що таку змінну не видно за межами мінімальної WFF, яка зв'язала цю змінну.

Під час обчислення значення такої правильно побудованої формули, використовується не одне значення зв'язаної змінної, а вся її область визначення.

Нехай СПІВРОБ1 і СПІВРОБ2 – дві кортежні змінні, які визначені на відношенні СПІВРОБІТНИКИ.

Приклад WFF 1:

EXISTS СПІВРОБ2 (СПІВРОБ1.СПІВРОБ_ЗАРПЛАТА > СПІВРОБ2.СПІВРОБ_ЗАРПЛАТА)

Для поточного кортежу змінної СПІВРОБ1цей вираз набуває значення true лише тоді, коли у всьому відношенні СПІВРОБІТНИКИ знаходиться такий кортеж (зв'язаний зі змінною СПІВРОБ2), що значення його атрибута СПІВРОБ_ЗАРПЛАТА задовільнить внутрішню умову порівняння.

Приклад WFF 2:

FORRAL СПІВРОБ2(СПІВРОБ1.СПІВРОБ_ЗАРПЛАТА > СПІВРОБ2.СПІВРОБ_ЗАРПЛАТА)

Для поточного кортежу змінна СПІВРОБ1набуває значення true тільки в тому випадку, якщо для усіх кортежів відношення СПІВРОБІТНИКИ (зв'язаних зі змінною СПІВРОБ2) значення атрибута СПІВРОБ_ЗАРПЛАТА задовольняє умову порівняння.

Насправді ж говорять не про зв'язані і вільні змінні, а про вільні і зв'язані входження змінних.

Прикладом мови, яка заснована на основі обчислення кортежів є мова SQL.

Цільові списки і вирази реляційного обчислення

WFF забезпечують засоби формування умови вибірки з відношення БД. Для того щоб можна було використати обчислення для реальної роботи з БД потрібен ще один компонент, що визначає набір та імена стовпців відношення-результату.Цей компонент називається цільовим списком (target-list).

Цільовий список будується з цільових елементів, кожний з яких може мати наступний вигляд: *var.attr, де var – ім'я вільної змінної відповідної WFF, а attr – ім'я атрибута відношення, на якому визначена змінна var.

Виразом реляційного обчислення кортежів називають конструкцію вигляду:

target_list WHERE WFF

Значенням виразу є відношення, тіло якого визначається WFF, а набір атрибутів і їхні імена – цільовим списком target_list.

Реляційне обчислення доменів

В обчисленні доменів областю визначення змінних є не відношення, а домени. Стосовно БД СПІВРОБІТНИКИ – ВІДДІЛИ можна говорити, наприклад, про доменні змінні СПІВРОБІТИК_ІМ'Я (значення – допустимі імена) або СПІВРОБІТИК_НОМЕР (значення – допустимі номера співробітників).

Основна відмінність обчислення доменів від обчислення кортежів є наявність додаткового набору предикатів, що дозволяють виражати умови членства.

Якщо R відношення з атрибутами $<a_1, a_2, \ldots, a_n>$, то умова членства має вигляд: $R(a_{i1} : v_{i1}, a_{i2} : v_{i2}, \ldots, a_{im}: v_{im})$ (m<=n), де v_{ij} – це або літеральна константа, що задається, або ім'я доменної змінної.

Умова членства набуває значення TRUE лише тоді, коли у відношенні R існує кортеж, який містить задані значення визначених атрибутів. Якщо v_{ij}– константа, то на атрибут a_{ij} накладається жорстка умова, яка не залежить від поточних значень доменних змінних.

Якщо ж v_{ij} – ім'я доменної змінної, то умова членства може набувати різних значень при різних значеннях цієї змінної.

Реляційне обчислення доменів є основою більшості мов запитів, що базуються на використанні форм. На цьому обчисленні базується мова запитів за зразком QUERY BY EXAMPLE.

Контрольні запитання:

1. Назвіть операції реляційної алгебри, що були запропоновані Дейтом.
2. Вкажіть підсумкові операції, які використовуються у виразі операції розширення, які дії вони виконують?
3. Перелічіть операції, що можуть використовуватися у виразі реляційного порівняння.
4. Які операції відносять до операцій реляційного числення?
5. Що таке предикат?
6. Що є областю визначення змінних в обчисленні кортежів і в обчисленні доменів?
7. Що таке правильно побудована формула (WFF) і для чого вона використовується?
8. Дайте визначення вільним і зв'язаним змінним правильно побудованої формули.
9. Що собою представляє цільовий список?
10. Назвіть основну відмінність обчислення доменів від обчислення кортежів.

Тестові завдання:

1. Яка функція визначає кількість записів, що повертаються запитом:

а) AVG;
б) MAX;
в) STDEV;
г) FIRST;
д) Немає правильної відповіді.

2. Які булеві оператори застосовуються в умовах вибору запиту?
а) =, >, <;
б) Sum, Avg, Min;
в) Max, Min, Avg;
г) AND, OR, NOT;
д) Немає правильної відповіді.

3.Яка функція вставляє новий рядок у представлення?
а)UPDATE;
б) UNION;
в) INSERT;
г) EXISTS;
д) Немає правильної відповіді.

4. Яка функція оновлює будь-яке значення у існуючому рядку?
а) FETCH;
б) UPDATE;
в) GROUPBY;
г)INSERT;
д) Немає правильної відповіді.

5. Яка функція видаляє існуючий рядок з представлення?
а)INSERT;
б) DELETE;
в) UPDATE;
г) UNION;
д) Немає правильної відповіді.

Розділ 6

Мова реляційного числення за зразком QBE (Query By Example)

- **Числення засноване на доменах**
- **Опис первинного варіанту мови QBE**
- **Характеристика мов QBE сучасних СКБД**

Числення кортежів схоже з численням доменів, але на відміну від числення кортежів, в численні доменів основою будь-якого запиту виступають змінні доменів.

Змінна домена – це скалярна змінна, значення якої охоплюють елементи деякого домену. Велика частина відмінностей даних числень полягає в тому, що числення доменів підтримує додаткову форму умови, яка називається умовою належності. Умова належності записується наступним чином:

$R(A_1 : V_1, A_2 : V_2, \ldots)$

де A_i – атрибут відношення R, а V_i – змінна домену або літерал. Умова, що перевіряється , справедлива якщо існує кортеж відношення R, що має атрибути A рівні, заданим у виразі, значенням V.

Дані, що зберігаються в БД можна обробляти за допомогою засобів СКБД. Для підвищення ефективності обробки даних використовують запити, які дозволяють проводити множинну обробку даних, тобто одночасно вводити, редагувати, видаляти множини з записів, а також вибирати необхідні дані з таблиць.

Запит – це спеціальним чином описана вимога, яка визначає склад операцій, що виконуються над БД (вибірка, видалення та модифікація). Для підготовки запитів різних СКБД використовуються дві основні мови: QBE (запити за зразком) та SQL (структурована мова запитів). Головна відмінність між цими мовами полягає у способі формування запитів. Мова QBE допускає ручне або візуальне формування запиту, а SQL – програмування запиту.

Первинний варіант мови QBE

Первинний варіант мови QBE був запропонований М. М. Злуфом в 1975-1977 роках. В основі цієї мови лежить реляційне числення основане на доменах. Крім Злуфа розробкою QBE займалися Лакроїкс та Піротте, які розробили на її основі мову ILL. Іншими мовами, які були засновані на обчисленні доменів були FQL та DEDUCE. За

твердженнями Дейта, QBE включає елементи числення кортежів і числення доменів, але є більш близькою до числення кортежів. Вона не є реляційноповною, оскільки не підтримує операцію заперечення квантора існування (NOTEXISTS).

Розглянемо основні можливості операцій, що використовуються в запитах.

Операція вибірки даних може бути трьох видів:

- проста вибірка – вибірка при якій вибираються ті чи інші значення з таблиці за введеним в запиті критеріями;

- проста вибірказ впорядкуванням.Для впорядкування значень, що виводяться, в порядку зростання або спадання використовуються конструкції «AT.» і «DO.» відповідно. Якщо вимагається виконати впорядковування по декількох стовпцях, застосовують конструкції вигляду: «AT(1).», «AT(2).».

- вибірка з кваліфікаторами (умовами).Вибір записів з початкової таблиці в загальному випадку може бути заснований на: точному збігу, частковому збігу та порівнянні.Умова порівняннязаписується за допомогою операцій порівняння: дорівнює (=), більше (>), менше (<), більше або дорівнює (>=), менше або дорівнює (<=), не дорівнює (¬ = або просто ¬), не більше (¬>), не менше (¬<).

У випадках, коли умови відбору записів для вибірки представляють великі вирази, які незручно або важко задати в шаблоні, можна використовувати **блок умов.** Віннагадує порожній шаблон з одним полем і ім'ям CONDITIONS. Блок умов призначений для запису логічних виразів. Записані в одному шаблоні логічні вирази, в загальному випадку, можуть включати операції логічного множення (операція AND) і логічного додавання (операція OR).

При записі логічних виразів на мові QBE можуть застосовуватися **вбудовані функції**, такі як: CNT. (лічильник або кількість), SUM. (сума), AVG. (середнє значення), MIN. (мінімум), MAX. (максимум), UN. (унікальне значення) і ALL. (всі значення, у тому числі і ті, що повторюються). Перші п'ять з них є статистичними, а останні дві визначають характер вибірки: включати або не включати у вибірку значення, що повторюються.

Функцію UN. можна приєднувати до функцій CNT., SUM. і AVG.. Так, запис CNT.UN. означає кількість значень, що відрізняються одне від одного. В протилежність цьому, запис CNT.ALL. означатиме кількість всіх значень.

На відміну від розглянутих операцій вибірки, що тільки фільтрують і відображають дані, операції вставки, видалення і модифікації приводять до змінипочаткової таблиці. Вид операції (вставка - I., видалення - D., модифікація - U.) записується в шаблоні під ім'ям таблиці, а константи і умовні вирази вказуються за тими ж правилами, що і в операціях вибірки.

Характеристика мов QBE сучасних СКБД

Мова QBE отримала широке розповсюдження в сучасних СКБД. Ця мова дозволяє задавати складні запити до бази даних шляхом заповнення запитальної форми, яку їй пропонує СКБД. Такий спосіб не вимагає вказівки алгоритму виконуваних операцій, достатньо описати лише зразок очікуваного результату. У кожної СКБД існує своя QBE. **За допомогою запитів на мові QBE можна виконувати такі основні операції:**

- вибірка даних;
- обчислення над даними;
- вставка нових записів;
- видалення записів;
- модифікація даних.

Результатом виконання запиту є нова таблиця, для перших двох операцій або оновлена початкова таблиця, для решти операцій. Обчислення над даними задаються за допомогою арифметичних виразів і породжують в таблицях нові поля, які називаються обчислювальними.

Основна відмінність мови QBE в сучасних СКБД від мови запропонованої Злуфом зводиться до незначних змін в інтерпретації окремих операцій та введення нових операцій, а також до змін в формі представлення мови.

Наприклад, в системі Paradox for Windows замість операції друку «P.» застосований метод відмітки вибраних в формі запиту (шаблоні) полів. Для цього на початку кожного з полів форми запиту розташовуються прапорці для вибору поля. Відзначаючи деякі поля, користувач може вказати послідовність сортування у відповідній таблиці.

Форма запиту має вид таблиці, ім'я і назви полів якої співпадають з ім'ям і назвами полів відповідної початкової таблиці. Щоб взнати імена доступних таблиць БД, в мові QBE передбачений запит на вибірку. Назви полів початкової таблиці можуть вводитися в шаблон вручну або

автоматично. В другому випадку використовується запит на вибірку заголовків стовпців.

Наочними є форми запитів в Microsoft Access. Діалогове вікно (рисунок 6.1) при підготовці форм запитів складається з двох частин: у верхній частині розташовується модель взаємозв'язку початкових таблиць, а в нижній — решта інформації про запит по кожному з полів (необхідність виводу значень, вид сортування, умова відбору і т. д.).

В сучасних СКБД, наприклад, в Access і Visual FoxPro, багато дій по підготовці запитів за допомогою мови QBE виконуються візуально за допомогою миші. Зокрема, візуальне скріплення таблиць для підготовки запиту виконується не елементами додатків, а просто «перетягуванням» поля однієї таблиці до поля іншої.

Аналіз сучасних СКБД дозволяє припустити наступні напрями розвитку мови QBE:

- підвищення наочності і зручності;

- поява нових засобів, що розширюватимуть можливості СКБД, наприклад, формування неточних або нечітких запитів, маніпулювання великими об’ємами даних, тощо;

- використання нових типів даних (графічних, аудіо-, відео- та інших);

- застосування у найближчому майбутньому обмеженої природної мови формування запитів;

- у віддаленій перспективі, використання мовного введення запитів.

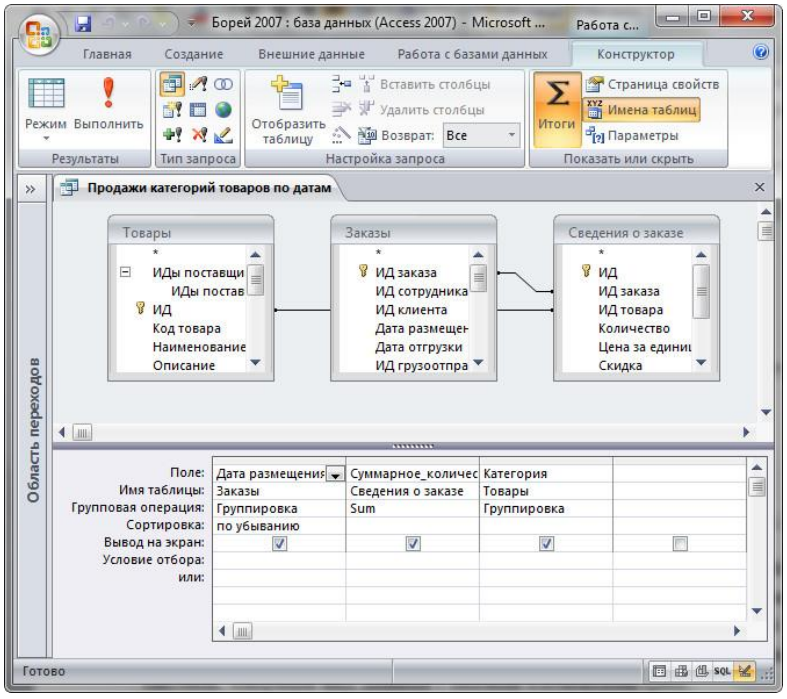


Рисунок 6.1 – Вигляд форми формування запиту в Microsoft Access

Принципові можливості для переходу до природної мови спілкування з засобами мовного введення є вже сьогодні. Це можна зробити, наприклад, у вигляді надбудови над існуючими СКБД при використанні словників відповідності термінів обмеженої природної мови і назв таблиць БД, полів таблиць, операцій над даними і іншими елементами QBE.

Контрольні запитання:

1. Що розуміють під змінною домену?
2. Що таке запит?
3. Які види операції вибірки даних є в QBE?
4. Перечисліть основні вбудовані функції мовиQBE.
5. Для чого використовується функція «UN.»?
6. Охарактеризуйте мову QBE.
7. Які операції можна виконувати за допомогою запитів на мові QBE?
8. Що представляє собою проста вибірка в запитах?
9. Що представляє собою проста вибірка з впорядкуванням в запитах?
10. Що представляє собою проста вибірка з кваліфікаторами в запитах?

Тестові завдання:

1. QBE-запит - це:

а) Запит за зразком;

б) Послідовність інструкцій, в яку можуть входити вирази і статистичні функції SQL;

в) Дескриптор;

г) Форма, яка використовується для виведення на екран великого об’єму інформації;

д) Немає правильної відповіді.

2. Яку умову необхідно вставити у рядку умови відбору для виявлення незавершених чи пустих записів за допомогою QBE запиту?

а) IS NOT NULL;

б) IS NULL;

в) = 0;
г) != NULL;
д) Немає правильної відповіді.

3. Який буде результат, якщо у QBE запиті використовувати умову: >=#01.01.97# AND <#01.01.98#?
а) Виведуться дати, що більші або дорівнюють 01.01.97 і менші за 01.01.98;
б) Виведуться тільки дати 01.01.97 і 01.01.98;
в) Виведуться дати менші за 01.01.97;
г) Виведуться дати менші за 01.01.97 і більші за 01.01.98;
д) Немає правильної відповіді.

4. Яка умова відповідає умові для відображення лише тих записів, де є дані у необхідному полі для QBE запиту?
а) IS NULL;
б) IS NOT NULL;
в) = 0;
г) *= NULL;
д) Немає правильної відповіді.

5. Умова відбору (для деякого поля): Like "К" виконає:*
а) Видалить всі записи поля, що починаються з великої літери «К»;
б) Присвоїть всім записам поля значення в лапках, тобто «К*»;
в) Замінить в записах поля всі великі літери К на малі;
г) Виведе на екран всі значення поля, що починаються з великої літери «К»;
д) Немає правильної відповіді.

Розділ 7

Структурована мова запитів SQL

- **Загальна характеристика мови SQL та огляд її можливостей**
- **Основні оператори мови SQL**
- **Приклади використання операторів SQL**

Структурована мова запитів SQL заснована на реляційному численні із змінними кортежами. Мова має декілька стандартів, найпоширенішими з яких є SQL-89 і SQL-92.

Мова SQL призначена для виконання операцій над таблицями (створення, видалення, зміна структури) і над даними таблиць (вибірка, зміна, додавання і видалення), а також деяких супутніх операцій. SQL є непроцедурною мовою і не містить операторів управління, організації підпрограм, введення-виведення і т.п. У зв'язку з цим SQL автономно не використовується, зазвичай вона вбудована в середовище мови програмування тієї чи іншої СКБД (наприклад, FoxPro – СКБД Visual FoxPro, ObjectPAL –СКБД Paradox, Visual Basic for Applications – СКБД Access).

В сучасних СКБД з інтерактивним інтерфейсом можна створювати запити, використовуючи інші засоби, наприклад QBE. Проте вживання SQL часто дозволяє підвищити ефективність обробки даних в базі. Наприклад, при підготовці запиту в середовищі Access можна перейти з вікна Конструктора запитів (формулювання запиту за зразком на мові QBE) у вікно з еквівалентним оператором SQL. Підготовку нового запиту шляхом редагування в більшості випадків простіше виконати через зміну оператора SQL. В різних СКБД склад операторів SQL може дещо відрізнятися.

Мова SQL не володіє функціями повноцінної мови розробки, а орієнтована на доступ до даних, тому її включають до складу засобів розробки програм. В цьому випадку її називають вбудованою SQL. Розрізняють два основні методи використовування вбудованої SQL: статичний і динамічний.

При статичному використанні мови (статична SQL) в тексті програми є виклики функцій мови SQL, які включаються у виконуваний модуль після компіляції. Зміни у функціях, що викликаються, можуть бути на рівні окремих параметрів викликів за допомогою змінних мови програмування.

При динамічному використанні мови (динамічна SQL) передбачається динамічна побудова викликів SQL-функцій і інтерпретація цих викликів, наприклад, звернення до даних віддаленої бази, в ході виконання програми. Динамічний метод звичайно застосовується у випадках, коли в додатку наперед невідомий вид SQL-виклику і він будується в діалозі з користувачем.Основним призначенням мови SQL (як і інших мов для роботи з базами даних) є підготовка і виконання запитів. В результаті вибірки даних з однієї абодекількох таблиць може бути отримано безліч записів, що звуться **представленням**. Представлення по суті є таблицею, сформованою в результаті виконання запиту. За одними і тими ж таблицями можна побудувати декілька представлень. Саме представлення описується шляхом вказівки ідентифікатора представлення і запиту, який повинен бути виконаний для його отримання.

Для зручності роботи з представленнями в мову SQL введено поняття курсора. **Курсор** є своєрідним покажчиком, що використовується для переміщення по наборах записів при їх обробці. Опис і використання курсора в мові SQL виконується таким чином. В описовій частині програми виконують скріплення змінної типу курсор (CURSOR) з оператором SQL (зазвичай з оператором SELECT). У виконуваній частині програми проводиться відкриття курсора (OPEN<ім'я курсора>), переміщення курсора по записах (FETCH <ім'я курсора>), супроводжується відповідною обробкою, і, нарешті, закриття курсора (CLOSE<ім'я курсора>).

Основні оператори мови SQL

Опишемо мінімальну підмножину мови SQL, спираючись на її реалізацію в стандартному інтерфейсі ODBC (Open Database Connectivity – сумісність відкритих баз даних) фірми Microsoft.

Оператори мови SQL можна умовно розділити на підмови: **мова визначення даних (**Data Definition Language **– DDL)** і **мова маніпулювання даними (**Data Manipulation Language **– DML) (**таблиця 7.1), **мова керування даними** (Data Control Language - **DCL**), **мова керування транзакціями TCL** (Transaction Control Language - **TCL**)**.**

DCL складається з команд: GRANT - використовується для призначення привілеїв користувачам, REVOKE – скасування привилеїв. TCL складається з команд: BEGIN Transaction – відкриває транзакцію, COMMIT Transaction - здійснює транзакцію, ROLLBACK Transaction -

можна використовувати для скасування всіх змін даних, що були отримані з початку транзакції або до точки збереження.

Таблиця 7.1 – Оператори мови SQL

Вид	Назва	Призначення
	CREATE TABLE	Створення таблиці
	DROP TABLE	Видалення таблиці
	ALTER TABLE	Зміна структури таблиці
	CREATE INDEX	Створення індексу
	DROP INDEX	Видалення індексу
	CREATE VIEW	Створення представлення
	DROP VIEW	Видалення представлення
	GRAND*	Призначення привілеїв
	REVOKE*	Видалення привілеїв
	SELECT	Вибірка записів
	UPDATE	Зміна записів
	INSERT	Вставка нових записів
	DELETE	Видалення записів

Розглянемо формат і основні можливості найважливіших операторів, за винятком специфічних операторів, відзначених в таблиці символом «*». Неістотні операнди і елементи синтаксису (наприклад, прийняте в багатьох системах програмування правило ставити «;» в кінці оператора) опускатимемо.

1. Оператор створення таблиці має формат вигляду:

CREATE TABLE <ім’я таблиці>

(<ім’я стовпця> <тип даних> [NOT NULL] (,<ім’я стовпця><тип даних>[NOT NULL]), …)

Обов'язковими операндами оператора є ім'я створюваної таблиці і ім'я хоча б одного стовпця (поля) з вказівкою типу даних, збережених в цьому стовпці.

При створенні таблиці для окремих полів можуть вказуватися деякі додаткові правила контролю для значень, що вводяться. Конструкція NOT NULL (не порожнє) служить саме втаких цілях і для стовпця таблиці означає, що в цьому стовпці повинне бути визначено значення.

В загальному випадку в різних СКБД можливе використання різних типів даних. В інтерфейсі ODBC підтримуються свої стандартні типи даних, наприклад, символьні (SQL_CHAR, SQL_VARCHAR, SQL_LONGVARCHAR) та ін. При роботі з БД в деякій СКБД за допомогою інтерфейсу ODBC виконується автоматичне перетворення стандартних типів даних, які підтримуються інтерфейсом, в типи даних джерел і навпаки. При необхідності обмін даними між програмою і джерелом даних може вестися без перетворення – у внутрішньому форматі даних джерела.

2. Оператор зміни структури таблиці має формат вигляду:

ALTER TABLE <ім’я таблиці>
((ADD, MODIFY, DROP) <ім’я стовпця> [<тип даних>] [NOT NULL]
[,(ADD, MODIFY, DROP) <ім’я стовпця> [<тип даних>] [NOT NULL]]…)

Зміна структури таблиці може полягати в додаванні (ADD), зміні (MODIFY) або видаленні (DROP) одного або декількох стовпців таблиці. Правила запису оператора ALTER TABLE такі ж, як і оператора CREATE TABLE. При видаленні стовпця вказувати <тип даних> непотрібно.

3. Оператор видалення таблиці має формат вигляду:

DROP TABLE<ім’я таблиці>

Оператор дозволяє видалити наявну таблицю. Наприклад, для видалення таблиці з ім'ям Items достатньо записати оператор вигляду:
DROP TABLE Items.

4. Оператор створення індексу має формат вигляду:

CREATE [UNIQUE] INDEX<ім’я індексу>ON<ім’я таблиці> (<ім’я стовпця> [ASC|DESC] [, <ім’я стовпця> [ASC|DESC] …)

Оператор дозволяє створити індекс для одного або декількох стовпців заданої таблиці з метою прискорення виконання запитальних і пошукових операцій з таблицею. Для однієї таблиці можна створити декілька індексів. Задавши необов'язкову опцію UNIQUE, можна забезпечити унікальність значень у всіх вказаних в операторі стовпцях. По суті, створення індексу з вказівкою ознаки UNIQUE означає визначення ключа в створеній раніше таблиці. При створенні індексу можна задати порядок автоматичного сортування значень в стовпцях — в порядку зростання ASC (за замовчуванням), або в порядку спадання DESC. Для різних стовпців можна задавати різний порядок сортування.

5. Оператор видалення індексу має формат вигляду:

DROP INDEX <ім’я індексу>

Цей оператор дозволяє видаляти створений раніше індекс з відповідним ім'ям. Так, наприклад, для знищення індексу main_indx до таблиці emp достатньо записати оператор DROP INDEX main_indx.

6. Оператор створення представлення має формат вигляду:

CREATE VIEW <ім’я представлення>
[(<ім’я стовпця>[, <ім’я стовпця>]…)] AS <оператор SELECT>

Даний оператор дозволяє створити представлення. Якщо імена стовпців в уявленні не вказуються, то використовуватимуться імена стовпців із запиту, що описаний відповідним оператором SELECT.

7. Оператор видалення представлення має формат вигляду:

DROP VIEW <ім’я представлення>

Оператор дозволяє видалити створене раніше представлення. При видаленні представлення, таблиці, що беруть участь в запиті, видаленню не підлягають. Видалення представлення проводиться оператором вигляду:

DROP VIEW Repr.

8. Оператор вибірки записів має формат вигляду:

SELECT [ALL | DISTINCT] <список даних> FROM <список таблиць>
[WHERE<умова вибірки>] [GROUP<ім’я стовпця> [, <ім’я стовпця>] …] [HAVING<умова пошуку>] [ORDER<специфікація> [, <специфікація>] …]

Це найважливіший оператор зі всіх операторів SQL. Функціональні можливості його величезні. Розглянемо основні з них.

Операнд SELECT дозволяє проводити вибірку і обчислення над даними з однієї або декількох таблиць. Результатом виконання оператора є таблиця, яка може мати (ALL), або не мати (DISTINCT) рядків, що повторюються. За замовчуванням у результуючу таблицю включаються всі рядки, у тому числі і ті, що повторюються. У відборі даних беруть участь записи однієї або декількох таблиць, перерахованих в списку операнда FROM. Список даних може містити імена стовпців, що беруть участь в запиті, а також вирази над стовпцями. В найпростішому випадку у виразах можна записувати імена стовпців, знаки арифметичних операцій (+, - , *, /) константи і круглі дужки. Якщо в списку даних записаний вираз, то разом з вибіркою даних виконуються обчислення, результати якого потрапляють в новий (створюваний) стовпець результуючої таблиці.

При використанні в списках даних імен стовпців декількох таблиць, для вказівки належності стовпця деякій таблиці, застосовують конструкцію вигляду: <ім'я таблиці>. <ім'я стовпця>.

Операнд WHERE задає умови, яким повинні задовільняти записи в результуючій таблиці. Вираз <умова вибірки> є логічним. Його елементами можуть бути імена стовпців, операції порівняння, арифметичні операції, логічні зв'язки (І, АБО, НІ), дужки, спеціальні функції LIKE, NULL, IN і т.д.

Операнд GROUP дозволяє виділяти в результуючій таблиці безлічі записів групи. Групою є записи із співпадаючими значеннями в стовпцях, що перераховані за ключовими словами GROUP. Виділення груп потрібне для використання в логічних виразах операндів WHERE і HAVING, а також для виконання операцій (обчислень) над групами.

В логічних і арифметичних виразах можна використовувати наступні групові операції (функції): AVG (середнє значення в групі), MAX (максимальне значення в групі),MIN (мінімальне значення в групі), SUM (сума значень в групі), COUNT (число значень в групі).

Операнд HAVING діє спільно з операндом GROUP і використовується для додаткової селекції записів під час визначення груп. Правила запису <умови пошуку> аналогічні правилам формування <умови вибірки> операнда WHERE.

Операнд ORDER задає порядок сортування результуючої множини. Звичайно кожна <специфікація> аналогічна відповідній конструкції оператора CREATE INDEX і є парою вигляду: <ім’я стовпця> [ASC | DESC].

Зауваження. Оператор SELECT може мати і інші складніші синтаксичні конструкції. Однією з таких конструкцій, наприклад, є так звані підзапити. Вони дозволяють формулювати вкладені запити, коли результати одного оператора SELECT використовуються в логічному виразі умови вибірки операнда WHERE іншого оператора SELECT.

Іншим прикладом складнішої форми оператора SELECT є оператор, в якому відібрані записи надалі передбачається модифікувати (конструкція FOR UPDATE). СКБД після виконання такого оператора зазвичай блокує (захищає) відібрані записи від модифікації їх іншими користувачами.

Ще один випадок специфічного використання оператора SELECT - виконання об'єднань результуючих таблиць, при виконанні декількох операторів SELECT (операнд UNION).

9. Оператор зміни записів має формат вигляду:

UPDATE<ім'я таблиці> SET <ім’я стовпця> = {<вираз>, NULL} [, SET<ім’я стовпця> = {<вираз>, NULL} …] [WHERE<умова>]

Виконання оператора UPDATE полягає в зміні значень у визначених операндом SET стовпцях таблиці, для тих записів, які задовільняють умові, що задана операндом WHERE. Нові значення полів в записах можуть бути порожніми (NULL), або обчислюватися відповідно до арифметичного виразу. Правила запису арифметичних і логічних виразів аналогічні відповідним правилам оператора SELECT.

10. Оператор вставки нових записів має формат двох видів:

1) INSERT INTO <ім’я таблиці> [(<список стовпців>)] VALUES (<список значень>)

2) INSERT INTO <ім’я таблиці> [(<список стовпців>)] <речення SELECT>

В першому форматі оператор INSERT призначений для введення нових записів із заданими значеннями в стовпцях. Порядок переліку імен стовпців повинен відповідати порядку значень, перерахованих в списку операнда VALUES. Якщо <список стовпців> опущений, то в <списку значень> повинні бути перераховані значення в порядку стовпців структури таблиці.

В другому форматі оператор INSERT призначений для введення в задану таблицю нових рядків, відібраних з іншої таблиці за допомогою пропозиції SELECT.

11. Оператор видалення записів має формат вигляду:

DELETE FROM <ім’я таблиці> [WHERE <умова>]

Результатом виконання оператора DELETE є видалення з вказаної таблиці рядків, які задовільняють умові, що визначена операндом WHERE. Якщо необов'язковий операнд WHERE опущений, тобто умова відбору записів, що видаляються, відсутня, видаленню підлягають всі записи таблиці.

Приклади використання операторів SQL

Приклад створення таблиці.

Нехай вимагається створити таблицю goods опису товарів, що має поля: type– вид товару, comp_id – ідентифікатор компанії-виробника, name– назва товару і price– ціна товару. Оператор визначення таблиці може мати наступний вигляд:

CREATE TABLE goods (type SQL_CHAR(8) NOT NULL comp_id SQL_CHAR(10) NOT NULL, name SQL_VARCHAR(20), price SQL_DECIMAL(8,2)).

Приклад додавання поля таблиці.

Нехай в створеній раніше таблиці goods необхідно додати поле number, що відводиться для збереження величини запасу товару. Для цього слід записати оператор вигляду:

ALTER TABLE goods (ADD number SQL_INTEGER).

Приклад створення індексу.

Хай для таблиці emp, що має поля: name (ім'я), sal (зарплата), mrg (керівник) idept (відділ), потрібно створити індексmain_indx для сортування імен в алфавітному порядку і спадання розмірів зарплати. Оператор створення індексу може мати вигляд:

CREATE INDEX main_indx ON emp (name, sal DESC).

Приклад створення уявлення.

Хай є таблиця companies опису виробників товарів з полями:comp_id (ідентифікатор компанії), comp_name(назва організації), comp_adress(адреса) iphone (телефон), а також таблиця goods вироблюваних товарів з полями: type (вид товару), comp_id (ідентифікатор компанії), name (назва товару) і price (ціна товару). Таблиці зв'язані між собою по полю comp_id. Вимагається створити представлення з короткою інформацією про товари і їх виробників: вид товару, назва виробника і ціна товару. Оператор визначення уявлення може мати наступний вигляд:

CREATE VIEW
ReprAS
SELECT
goods.type, companies.comp_name, goods.price FROM goods, companies WHERE goods.comp_id = companies.comp_id.

Приклад вибору записів.

Для таблиці emp, що має поля: name (ім'я), sal (зарплата), mgr (керівник) і DEPT (відділ), вимагається вивести імена співробітників і розмір їх зарплати, збільшений на 100 одиниць. Оператор вибору можна записати таким чином: SELECT name, sal+100 FROM emp.

Приклад вибору з умовою.

Вивести назви таких відділів таблиці emp, в яких в даний момент відсутні керівники. Оператор SELECT для цього запиту можна записати так:

SELECT dept FROM emp WHERE mgris NULL.

Приклад вибору з групуванням.

Хай вимагається знайти мінімальну і максимальну зарплати для кожного з відділів (в таблиці emp). Оператор SELECT для цього запиту має вигляд:

SELECT dept, MIN(sal), MAX(sal) FROM emp GROUP dept.

Приклад зміни записів.

Хай необхідно збільшити на 500 одиниць зарплату тим службовцям, які отримують не більше 6000 (по таблиці emp). Запит, сформований за допомогою оператора SELECT,може виглядати так:

UPDATE emp
SET sal = 6500
WHERE sal <= 6000.

Приклад введення записів.

Ввести в таблицю emp запис про нового співробітника. Для цього можна записати такий оператор:

INSERT INTO emp
VALUES («Ivanov», 7500, «Lee», «cosmetics»).

Приклад видалення записів.

У зв'язку з ліквідацією відділу іграшок (toy), вимагається видалити з таблиці всіх співробітників цього відділу. Оператор DELETE для цієї задачі виглядатиме так:

DELETE FROM emp
WHERE dept – «toy».

На закінчення відзначимо, що, за словами Дейта, мова SQL є гібридом реляційної алгебри і реляційного числення. В ній присутні елементи алгебри (оператор об’єднання UNION) і числення (квантор існування EXISTS). Крім того, мова SQL володіє реляційною повнотою.

Контрольні запитання:

1. Що таке представлення?
2. Що таке курсор?
3. Для чого призначена мова SQL?
4. Що розуміють під статичною і динамічною SQL?
5. На які підмови можна розділити оператори мови SQL?
6. Назвіть декілька основних операторів мови SQL та їх призначення.

7. Для чого використовується конструкція NOT NULL?

8. Назвіть основні типи даних, що підтримуються в інтерфейсі ODBC.

9. Які три оператори мови SQL, які дозволяють змінити структуру таблиці?

10. Що дозволяє зробити додаткова опція UNIQUE при створенні нового індексу?

11. Для чого використовується оператор ORDER?

12. Назвіть групові операції, які можна використовувати в логічних і арифметичних виразах.

Тестові завдання:

1. Мова SQL - це:

а) Мова запитів, основана на операціях пошуку даних;

б) Базова мова інформаційних систем;

в) Мова запитів, основана на реляційному численні із змінними кортежами;

г) Процедурна мова реляційної алгебри;

д) Немає правильної відповіді.

2. Яку функцію виконує SQL:

а) Забезпечує зв'язок з VBA;

б) Створює нові звіти;

в) Створює модулі;

г) Здійснює управління даними;

д) Немає правильної відповіді.

3. SQL–запит це:

а) Запит за зразком;

б) Послідовність інструкцій, в яку можуть входити вирази і статистичні функції SQL;

в) Звіт, параметри якого встановлюються у вікні конструктора звітів;

г) Форма, яка використовується для виведення на екран великого об’єму інформації;

д) Немає правильної відповіді.

4. Що представляє собою поняття DDL?

а)Мова визначення даних;

б)Мова маніпулювання даними;
в)Зовнішній ключ;
г)Журналізація;
д)Немає правильної відповіді.

5. Що представляє собою поняття DML?
а)Мова визначення даних;
б) Мова маніпулювання даними;
в) Зовнішній ключ;
г) Журналізація;
д) Немає правильної відповіді.

*6. Що означає інструкція: SELECT * FROM STD; ?*
а) Відбудеться виведення даних всіх полів таблиці STD;
б) Відбудеться виведення поля STD будь-якої таблиці;
в) Відбудеться видалення будь-якого значення полів таблиці STD;
г) Відбудеться поновлення даних будь-якого поля таблиці STD;
д) Немає правильної відповіді.

7. Для створення унікальних індексів у БД за допомогою SQL використовується ключове слово:
а) UNIQUE;
б) ALL;
в) 33 Н.Ф.;
г) SUB;
д) Немає правильної відповіді.

8. Яка команда DDL створює порожню таблицю з визначеним ім'ям та визначеним набором полів?
а) Бойса Кодда;
б)UNIQUE;
в) CREATETABLE;
г) DROPTABLE;
д) Немає правильної відповіді.

9. Що виконує така інструкція SQL: DROP TABLE STUDENTS;?
а)Приведе до зміни в таблиці STUDENTS всіх оцінок на «5»;
б)Захищає базу даних за допомогою пароля STUDENTS;

в)Виконується видалення пустої таблиці STUDENTS;
г)Створення унікальних індексів;
д) Немає правильної відповіді.

10.За допомогою якої команди можна відібрати рядки, в яких значення будь-якого стовпця знаходиться у заданому діапазоні?
а)WHERE;
б) CREAT VIEW;
в) DECLARE CURSOR;
г) BETWEEN та AND;
д) Немає правильної відповіді.

Розділ 8

Історія мови SQL та її можливості

- **Історія мови SQL та огляд її можливостей**
- **Засоби пошуку даних**
- **Θ – з'єднання**
- **Використання агрегатних функцій**

Історія мови SQL та огляд її можливостей

Історія SQL починається з 70-х років XX століття, коли в дослідницькій лабораторії IBM у штаті Каліфорнія було розроблено першу версію цієї мови. **Назва SQL** є абревіатурою від Structured Query Language (структурована мова запитів), й іноді її вимовляють як «sequel» (первісна назва). Спочатку ця мова була реалізована в реляційній СКБД DB2 виробництва IBM. На відміну від мов третього покоління (COBOL, C), які з'явилися в той самий час, мова SQL не є процедурною. Не процедурна мова - це мова, в якій описується, що потрібно одержати, а не як це зробити.

Особливість реляційних СКБД полягає у тому, що вони надають множинно-орієнтовану мову маніпулювання базами даних, тобто результатом дії мовного оператора є таблиця, яка містить множину даних. Більшість сучасних реляційних СКБД використовують саме мову SQL.

Американський інститут національних стандартів (American National Standards Institute ANSI) та Міжнародна організація стандартів (International Standards Organization — ISO) займаються описом і підтримкою стандартів цієї мови. Усі сучасні СКБД підтримують певний стандарт, проте є й відхилення, які в кожному конкретному випадку специфікуються в документації програмного продукту. Окрім того, у багатьох системах розроблено розширення SQL, що дають змогу використовувати мову запитів у середовищі програмування.

SQL надає такі можливості:

• створювати й видаляти таблиці бази даних, а також змінювати заголовки таблиць;

• вставляти, змінювати й видаляти рядки в таблицях;

• виконувати пошук даних у багатьох таблицях та впорядковувати результати цього пошуку:

• описувати процедури підтримки цілісності;

• визначати та змінювати інформацію про захист даних.

Керуючись стандартами ANSI-92 та ANSI-99, розглянемо можливості SQL на численних прикладах. Усі запити конструюватимуться для тієї ж бази даних, що була використана під час розгляду реляційної алгебри та реляційного числення:

ФАКУЛЬТЕТ (#F, Назва, Декан, Корпус, Фонд)

КАФЕДРА (#D, #F, Назва, #ЗАВІДУВАЧ, Корпус, Фонд)

ВИКЛАДАЧ (#T, #D, Прізвище, Посада, Тел)

ГРУПА (#G, #D, Курс, Номер, Кількість, #КУРАТ0Р)

ПРЕДМЕТ (#S, Назва)

АУДИТОРІЯ (#R, Номер, Корпус, Місткість)

ЛЕКЦІЯ (#T, #G, #S, #R, Тип, День, Тиждень)

Засоби пошуку даних

Основна конструкція, призначена у мові SQL для вибирання даних, складається з фраз SELECT і FROM. Фраза FROM вказує, з якої таблиці потрібно вибрати дані, а фраза SELECT - які саме атрибути (стовпці) з цієї таблиці мають бути вибрані. Запит

SELECT Назва

FROM ФАКУЛЬТЕТ

здійснює виведення назв факультетів. Ці дві фрази обов'язково мають бути в будь-якому запиті.

Виведення окремих стовпців

У фразі SELECT можна зазначати список імен стовпців. Передбачається, що результат виведення буде впорядкований за стовпцями відповідно до того, як розташовані імена у фразі:

SELECT Номер, Курс, Кількість

FROM ГРУПА

Виведення всіх стовпців

Якщо необхідно вивести всі стовпці таблиці, то у фразі SELECT використовується **символ *:**

SELECT *

FROM КАФЕДРА

Неповторювані рядки

Хоча в реляційних відношеннях не має бути повторюваних рядків (дублікатів), у SQL за замовчуванням встановлено, що всі дублікати рядків у таблиці-результаті виводяться. Щоб унаслідок виконання запиту одержати унікальні (неповторювані) значення, потрібно використовувати модифікатор DISTINCT (за замовчуванням застосовується модифікатор ALL). Наприклад, щоб отримати список

усіх типів лекцій, які читаються у вузі, і щоб кожен тип виводився лише один раз, потрібно записати:

SELECT DISTINCT Тип
FROM ЛЕКЦІЯ

Без модифікатора DISTINCT ми одержали б список із кількох сотень рядків (його довжина дорівнювала б кількості всіх лекцій у вузі).

Зазначимо, що весь запит можна розмістити в одному рядку.

Перевизначення імен стовпців

Фраза SELECT надає можливість перевизначити імена стовпців кінцевої таблиці. Для цього після імені стовпця вихідної таблиці необхідно зазначити ім'я стовпця кінцевої таблиці (з використанням необов'язкової фрази AS). Наприклад, у наведеному нижче запиті перевизначаються імена обох стовпців:

SELECT Назва AS Назва_факультету, Декан AS Декан_факультету
FROM ФАКУЛЬТЕТ

Умова відбору

Для запису умови відбору використовується фраза WHERE. У ній зазначено, якій умові мають відповідати вихідні дані. Алгоритм обробки запиту з фразою WHERE:

• вибрати рядок із таблиці;

• перевірити його відповідність вказаній умові;

• якщо рядок відповідає умові, то вивести значення стовпців, вказаних у фразі SELECT.

Цей запит виводить список усіх професорів вузу:

SELECT Прізвище
FROM ВИКЛАДАЧ
WHERE Посада = "професор"

Вирази, умови та оператори

У мові SQL існує багато різновидів виразів, у яких використовуються дані різних типів - рядки, числа, логічні значення. Конструкція SELECT...FROM...WHERE також є виразом, не кажучи вже про просте згадування імені стовпця у фразі SELECT.

Умова - це вираз, що повертає логічне значення — TRUE чи FALSE. Умовні вирази обов'язково використовуються у фразі WHERE, а також можуть застосовуватися в інших фразах, наприклад SELECT. Прикладом умовного виразу є конструкція Посада = <професор>.

Оператори - це конструкції, що використовуються у виразах для означення певних дій над даними. Є кілька типів операторів:

арифметичні, логічні, теоретико - множинні. оператори порівняння, оператори над рядками. У конкретних системах списки цих операторів можуть відрізнятися, зокрема бути ширшими, ніж наведений далі список.

- Арифметичні оператори: +,-,*,/.
- Оператор зчеплення рядків ||.
- Теоретико - множинні оператори: UNION, INTERSECT, EXCEPT (або MINUS),
- Логічніоператори: AND, OR, NOT.
- Оператори порівняння: >, <, =, <>, >=,<=.

Оператори порівняння. Їхнє призначення та приклади використання наведені в таблиці 8.1.

Таблиця 8.1 – Оператори порівняння

Оператор	**Призначення**	**Приклад**
=	Перевірка на рівність	**SELECT * FROM** КАФЕДРА **WHERE** Фонд =1500
!=, ^=, <>	Перевірка на нерівність	**SELECT * FROM** КАФЕДРА **WHERE** Фонд !=1500
>, <	Перевірка, чи одне значення більше або менше іншого	**SELECT * FROM** КАФЕДРА **WHERE** Фонд>1500 **SELECT * FROM** КАФЕДРА **WHERE** Фонд< 1500
>=, <=	Перевірка, чи одне значення не більше або не менше іншого	**SELECT * FROM** ФАКУЛЬТЕТ **WHERE** Фонд>= 1500 **SELECT * FROM** ФАКУЛЬТЕТ **WHERE** Фонд<= 1500
IN	Перевірка на належність елементу множині	SELECT * **FROM** ВИКЛАДАЧ **WHERE** Посада IN ("професор","доцент") **SELECT * FROM** ВИКЛАДАЧ

		WHERE Посада **IN** (SELECT Посада FROM ВИКЛАДАЧ **WHERE** tel ="5261815")
NOTIN	Еквівалентний != ALL Перевірка на неналежність елементу множині	SELECT * **FROM** ВИКЛАДАЧ **WHERE** Посада **NOT IN** ("професор","доцент**")** **SELECT * FROM** ВИКЛАДАЧ **WHERE** Посада**NOT IN** (SELECT Посада **FROM** ВИКЛАДАЧ **WHERE** tel - "5261815")
ANY, **SOME**	Використовуються разом із предикатом =, !=, >, <, <= або >=. Перевіряють, чи істинний цей предикат хоча б для одного елементу множини, заданої у правій частині оператора, відносно елементу, заданого в лівій частині	SELECT * **FROM** ФАКУЛЬТЕТ **WHERE** Фонд = ANY (SELECT Фонд FROM ФАКУЛЬТЕТ WHERE Корпус = 7)
ALL	Використовується разом з предикатом =,!=, >, <, <= або >=. Перевіряє, чи істинний цей предикат для всіх елементів множини, заданої у правій частині оператора, відносно елементу, заданого в	SELECT * **FROM** ФАКУЛЬТЕТ **WHERE** Фонд>= **ALL** (1400,3000)

	лівій частині	
EXISTS	Повертає TRUE, якщо підзапит, до якого застосовується оператор, містить хоча б один рядок	**SELECT** Назва **FROM** Факультет WHERE EXISTS (SELECT * **FROM** Кафедра **WHERE** Кафедра.#F = Факультет. #F)
IS [NOT] NULL	Перевірка на рівність значенню NULL	**SELECT** Назва **FROM** Кафедра **WHERE** Фонд **IS NULL**

Розглянемо на прикладах використання цих операторів.

Запит

Визначити групи, де кількість студентів становить від 40 до 50 осіб, а також курси, які їм відповідають.

SELECT Номер, Курс
FROM ГРУПА
WHERE Кількість >= 40 AND Кількість <= 50

Запит

Визначити прізвища й посади професорів та асистентів.

SELECT Прізвище, Посада
FROM ВИКЛАДАЧ
WHERE Посада = "професор" OR Посада = "асистент"

Вибірка з кількох таблиць

Щоб вибрати дані з багатьох таблиць, необхідно перелічити їхні імена у фразі FROM. Якщо у фразі WHERE не зазначити умову з’єднання таблиць, то обчислюється декартів добуток усіх таблиць із фрази FROM.

Наприклад, задано дві таблиці:

Table1

ROW	REMARK
Row1	Table2
Row2	Table2
Row3	Table2

Table2

ROW	REMARK
Row1	Table1
Row2	Table1
Row3	Table1

Результатом виконання запиту

SELECT *

FROM TABLE1, TABLE2

буде таблиця:

Table1.ROW	**Table1.REMARK**	Table2.ROW	**Table2.REMARK**
Row1	Table1	Row1	Table2
Row1	Table1	Row2	Table2
Row1	Table1	Row3	Table2
Row2	Table1	Row1	Table2
Row2	Table1	Row2	Table2
Row2	Table1	Row3	Table2
Row3	Table1	Row1	Table2
Row3	Table1	Row2	Table2
Row3	Table1	Row3	Table2

Як видно, кожен рядок першої таблиці з'єднався з кожним рядком другої. Зазвичай таблиці поєднуються за певної умови. Наприклад, для визначення назв факультетів та відповідних їм кафедр слід записати:

SELECT ФАКУЛЬТЕТ.Назва, КАФЕДРА.Назва

FROM ФАКУЛЬТЕТ, КАФЕДРА

WHERE ФАКУЛЬТЕТ #F=КАФЕДРА #F

Уточнення імен стовпців

У фразах SELECT* і WHERE імена стовпців можна уточнювати іменами таблиць. Якщо в поєднуваних таблицях є стовпці, що мають

однакові імена, то посилаючись на такий стовпець у запиті, його ім'я потрібно уточнювати іменем таблиці.

Псевдоніми таблиць

Таблиці можуть мати довгі назви, тому мова надає можливість зв'язувати з кожною таблицею певний псевдонім і надалі посилатися на таблицю за ним. Зіставлення таблиці з псевдонімом здійснюється у фразі FROM. Наприклад, попередній запит можна записати в такий спосіб:

```
SELECT f.Назва, d. Назва
FROM   ФАКУЛЬТЕТ f, КАФЕДРА d
WHERE f. #F = d. #F
```

Фраза WHERE може використовуватися для зазначення способу з'єднання таблиць або умови відбору рядків до кінцевої таблиці.
Зокрема, запит

```
SELECT   КАФЕДРА. Назва
FROM     КАФЕДРА, ВИКЛАДАЧ
WHERE    КАФЕДРА. #D = ВИКЛАДАЧ. #D AND ВИКЛАДАЧ.
Прізвище = "Іванов"
```

виводить назву кафедри, де працює викладач Іванов. (Зауважимо, що пошук викладачів ведеться в усьому вузі, тому будуть знайдені всі кафедри, де працюють викладачі на прізвище Іванов). Зверніть увагу на те, що умова пошуку задається на одній таблиці, а результат виводиться з іншої.

Стовпці, що обчислюються

У фразі SELECT можна сформувати новий стовпець. У ньому записуватимуться результати обчислення виразу, в якому використовуються значення з інших стовпців таблиць, що з'єднуються. Наведений нижче запит видає дані про всі кафедри факультету інформатики разом з їхніми фондами та інформацією про те, яку частку становить фонд кафедри від загального фонду факультету.

```
SELECT d. Назва,d. Фонд, (d. Фонд / f. Фонд) * 100
FROM  ФАКУЛЬТЕТ f, КАФЕДРА d
WHERE  f.#F = d.#F AND F. Назва = "інформатики"
```

Еквіз'єднання

Якщо таблиці з'єднуються за рівністю значень в одній чи кількох парах стовпців, до того ж із кожної таблиці вибираються всі стовпці, то таке з'єднання відповідає операції еквіз'єднання реляційної алгебри. Наприклад:

SELECT f.*,d.*
FROM ФАКУЛЬТЕТ f, КАФЕДРА d
WHERE f.#F = d.#F

Природне з’єднання

Операція **природного з’єднання** здійснюється з'єднанням двох чи кількох таблиць за рівністю значень в одній чи кількох парах стовпців із наступним видаленням повторюваних стовпців (необхідні стовпці вказуються у фразі SELECT). Наприклад:

SELECT f.#F, f.Назва, f.Декан, f.Корпус, f.Фонд.
d.#D,d.Назва,d.#ЗАВІДУВАЧ, d.Корпус, d.Фонд
FROM ФАКУЛЬТЕТ f, КАФЕДРА d
WHERE f.#F = d.#F

Хоча стовпці Назва, Корпус, Фонд є в обох таблицях, однак вони не розглядаються як повторювані, оскільки мають різне семантичне навантаження. Стовпцем, який повторюється, *є* той єдиний, за яким виконується з'єднання. — стовпець #F, що виконує роль зовнішнього ключа в таблиці КАФЕДРА.

Θ – з’єднання

Це з'єднання за будь-якою умовою. Θ - з'єднання виконується не за первинним і зовнішнім ключами, а за іншими стовпцями. Раніше наведені різновиди з'єднання є окремими випадками Θ - з'єднання.

З'єднання таблиці зі своєю копією

Інколи необхідно з'єднати таблицю із самою собою. Для цього у фразі FROM потрібно двічі зазначити назву таблиці *з* різними псевдонімами, щоб можна було на кожен її екземпляр посилатися окремо. Розглянемо такий приклад. Необхідно перевірити, чи є в таблиці ФАКУЛЬТЕТ пари рядків, де назви факультетів збігаються, а ключі #F різні

SELECT f1. Назва, f1.#F, f2.#F
FROM ФАКУЛЬТЕТ f1, ФАКУЛЬТЕТ f2
WHERE f1. Назва = f2.Назва AND f1.#F != f2.#F

З'єднання можна виконувати без посередньо у фразі FROM, скориставшись одним із різновидів оператора JOIN (наприклад, Fl INNERJOIN F2 ON F1.#F != F2.F#). Слід пам'ятати, що за певних обставин потрібно саме виконувати JOIN- з'єднання, а не обчислювати декартів добуток таблиць. Наприклад, декартів добуток двох таблиць з десятками тисяч рядків у кожній обчислюватиметься протягом кількох годин, а обсяг кінцевої таблиці буде вимірюватися сотнями гігабайтів!

Розглянемо кілька запитів, що вже використовувались як приклади. Запити будемо записувати як мовою SQL, так і за допомогою реляційної алгебри.

Запит

Визначити всі кафедри факультету інформатики. Мова SQL:

SELECT КАФЕДРА. Назва

FROM ФАКУЛЬТЕТ. КАФЕДРА

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F AND ФАКУЛЬТЕТ. Назва = "інформатики"

Реляційна алгебра: ((ФАКУЛЬТЕТ [#F = #F] КАФЕДРА)[ФАКУЛЬТЕТ. Назва - "інформатики"])[КАФЕДРА.Назва]

Запит

Визначити всіх викладачів кафедри АСУ та їхні телефонні номери. Мова SQL:

SELECT Прізвище, Тел

FROM КАФЕДРА, ВИКЛАДАЧ

WHERE КАФЕДРА. #D = ВИКЛАДАЧ. #D AND КАФЕДРА. Назва = "АСУ"

Реляційна алгебра: ((КАФЕДРА[#D =#D]ВИКЛАДАЧ)[КАФЕДРА. Назва = "АСУ"])[ВИКЛАДАЧ. Прізвище. Тел]

Запит

Вивести список усіх викладачів факультету інформатики разом з їхніми телефонними номерами.

Мова SQL:

SELECT Прізвище, Тел

FROM ФАКУЛЬТЕТ, КАФЕДРА, ВИКЛАДАЧ

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F AND КАФЕДРА.#D = ВИКЛАДАЧ.#D **AND** ФАКУЛЬТЕТ. Назва = "інформатики"

Реляційна алгебра:

(((ФАКУЛЬТЕТ[#F = #F]КАФЕДРА)[#D = #D]ВИКЛАДАЧ) [ФАКУЛЬТЕТ. Назва = "інформатики"]) [ВИКЛАДАЧ. Прізвище, Тел]

Запит

Визначити номери груп першого курсу кафедри АСУ.

Мова SQL:

SELECT Номер

FROM ГРУПА, КАФЕДРА

WHERE ГРУПА.#D = КАФЕДРА.#DAND Назва = "АСУ" AND Курс = 1

Реляційна алгебра: ((ГРУПА[#D = #D]КАФЕДРА)(Назва "АСУ" & Курс=1]) [Номер]

Запит

Визначити номери груп першого курсу кафедри АСУ та прізвища їхніх кураторів.

Мова SQL:

SELECT Номер. Прізвище

FROM ГРУПА, КАФЕДРА, ВИКЛАДАЧ

WHERE ГРУПА.#D = КАФЕДРА.#DAND ГРУПА.#КУРАТОР = ВИКЛАДАЧ.#T **AND** Назва = "АСУ" AND Курс =1

Реляційна алгебра: (((ГРУПА[#D=#D]КАФЕДРА)(#КУРАТОР=#T]ВИКЛАДАЧ)[Назва="АСУ"&Курс =1)] [Номер. Прізвище]

Запит

Визначити лекції, на яких кількість студентів у групі перевищує кількість місць в аудиторії. Вивести номери аудиторій та груп, назви дисциплін, що читаються, а також дні й тижні проведення лекцій.

Мова SQL:

SELECT АУДИТОРІЯ. Номер, ГРУПА. Номер, ПРЕДМЕТ. Назва, Тиждень, День

FROM ЛЕКЦІЯ, ГРУПА, ПРЕДМЕТ, АУДИТОРІЯ

WHERE ЛЕКЦІЯ.#G = ГРУПА.#GAND

ЛЕКЦІЯ.#S = ПРЕДМЕТ.#SAND

ЛЕКЦІЯ.#R = АУДИТОРІЯ.#RAND

ГРУПА.Кількість>АУДИТОРІЯ.Місткість

Реляційнаалгебра:

(((ЛЕКЦІЯ[#G = #G]ГРУПА)[#S = #S]ПРЕДМЕТ)[#R = #R]АУДИТОРІЯ)[ГРУПА.Кількість>

АУДИТОРІЯ.Місткість])[АУДИТОРІЯ.Номер, ГРУПА.Номер, ПРЕДМЕТ.Назва, Тиждень, День]

Використання агрегатних функцій

SQL містить функції, що дають змогу обчислювати різні статистичні характеристики. Такі функції називаються агрегатними. Стандарт ANSI визначає певний набір функцій, однак у СКБД він зазвичай значно розширюється. Наприклад, діалект SQL, що використовується в СКБД Oracle, містить понад 120 функцій.

З усієї множини функцій SQL ми розглянемо лише ті, що перелічені в стандарті ANSI. А саме:

- COUNT — повертає кількість рядків у таблиці;
- SUM — повертає суму всіх значень у стовпці;
- AVG — повертає середнє арифметичне всіх значень у стовпці;
- MAX — повертає найбільше значення у стовпці;
- MIN — повертає найменше значення у стовпці.

Окрім COUNT(*), кожна з цих функцій оперує (як аргументом) сукупністю значень стовпця певної таблиці та повертає єдине значення. Для функцій SUM і AVG стовпець-аргумент повинен містити числові значення.

Слід зазначити, що у даному випадку стовпець-аргумент — це стовпець віртуальної таблиці, в якій можуть міститися дані не лише зі стовпця базової таблиці, але й отримані шляхом функціонального перетворення та/або зв'язування символами арифметичних операцій значень з одного або кількох стовпців. Вирази, що визначають стовпець такої таблиці, мають різний рівень складності, але не можуть містити інших SQL-функцій. Наприклад, вираз AVG(Ставка * Надбавка/2) є припустимим, а вираз AVG(SUM(...)) - ні. SQL-функції можуть входити до складу виразів, наприклад SUM (Фонд)/Count (*).

Аргументам усіх функцій, окрім COUNT(*), може передувати модифікатор DISTINCT (різний), що вказує на необхідність видалення дублікатів перед застосуванням функції. Функція COUNT(*) використовується для обчислення кількості всіх рядків таблиці з урахуванням дублікатів.

Агрегатні функції можна застосовувати лише у фразах SELECT та HAVING. Якщо в запиті немає фрази GROUPBY, то область дії агрегатної функції поширюється на все кінцеве реляційне відношення, а в разі її наявності - на створювані нею групи. Розглянемо використання цих функцій на прикладах.

Запит

Визначити кількість кафедр на факультеті інформатики.

SELECT COUNT(*)
FROM ФАКУЛЬТЕТ, КАФЕДРА
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#FAND
ФАКУЛЬТЕТ. Назва = "Інформатики"

Результат цього запиту може виглядати так:

COUNT (*)

Запит

Визначити кількість предметів, шо читаються.

SELECT COUNT(*) Number_Of_Subjects

FROM ПРЕДМЕТ

Тепер результат виглядає так:

NumberOfSubjects

Запит

Визначити місткість усіх аудиторій у корпусі 5.

SELECT SUM(Місткість)Total_Number_Of_Seats_ln_Building_5

FROM АУДИТОРІЯ

WHERE Корпус = 5

Запит

Визначити кількість студентів факультету інформатики.

SELECT SUM(ГРУПА. Кількість) Number_Of_IT_Faculty_Students

FROM ФАКУЛЬТЕТ,КАФЕДРА,ГРУПА

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#FANDКАФЕДРА#D = ГРУПА.#D**AND**

ФАКУЛЬТЕТ.Назва = "інформатики"

Запит

Визначити найбільший фонд із фінансування серед кафедр факультету інформатики.

SELECT MAX(КАФЕДРА.Фонд) MAX_Department_Fund_in_IT_Faculty

FROM ФАКУЛЬТЕТ,КАФЕДРА

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F AND ФАКУЛЬТЕТ.Назва= "інформатики**"**

Можна задавати кілька агрегатних функцій, що використовуються у виразах.

Запит

Визначити місткість усіх аудиторій корпусу *5,* кількість містить у найменшій та найбільшій аудиторіях, середню місткість.

SELECT SUM(Місткість) TotaL_Number_Of_Seats_ln_Buildlng_5,

MIN(Місткість) Seats_In_Tne_Smallest_Room,

MAX(Місткість) Seats_ln_The_Largest_Room,

SUM(Місткість/COUNT(*) Average Number_Of_Seats

FROM АУДИТОРІЯ

WHERE Корпус = 5

Якщо в запиті немає фрази GROUP BY, то використання імен стовпців разом із агрегатними функціями у фразі SELECT є неприпустимим.

Наприклад, у результаті виконання запиту

SELECT Корпус, SUM (Місткість)
TotaI_Number_Of_Seats_In_Buitding
FROM АУДИТОРІЯ

може бути виведене повідомлення про помилку:

Dynamic SQL Error
-SQL error code = -104
-invalid column reference

Агрегатні функції не використовуються у фразі WHERE. В результаті виконання запиту

SELECT Назва
FROM КАФЕДРА
WHERE Фонд = MAX (Фонд)

може бути виведене повідомлення про помилку:

ERROR at line 3:
ORA-00934: group function is not allowed here

Невизначені значення в агрегатних функціях

У стовпці – аргументі агрегатної функції перед застосуванням будь-якої функції, окрім COUNT(*), виключаються всі невизначені значення. Створюючи запит:

SELECT COUNT(Корпус), COUNT(DISTINCT Корпус),
COUNT(*)
FROM ФАКУЛЬТЕТ

ви можете отримати, наприклад, таку відповідь:

COUNT(Корпус) COUNT(DISTINCT Корпус) COUNT(*)
---
15 11 17

Результат виконання запиту слід інтерпретувати так: усього є 17 факультетів; для 15 з них задані значення корпусу (два факультети в полі корпусу містять NULL), і серед цих 15 значень корпусів лише є різними, тобто неповторюваними.

Якщо виявляється, що аргумент – порожняя множина, функція COUNT набуває значення 0, а інші — NULL

Контрольні запитання:

1. Що таке Structured Query Language?
2. Які можливості надає SQL?
3. Дайте визначення терміну умова.
4. Дайте визначення терміну оператор.
5. Які функції називають агрегатними?
6. Наведіть приклади операторів порівняння.
7. Які основні агрегативні функції SQL ви знаєте?
8. На що вказує модифікатор DISTINCT?

Тестові завдання:

1. Де було розроблено першу версію мови SQL?

а) в Україні
б) в штаті Каліфорнія
в) в Москві
г) в штаті Нью-Йорк

2. Теоретико - множинні оператори це :

а) UNION, INTERSECT, EXCEPT (або MINUS)
б) AND, OR, NOT
в) +,-,*,/
г) правильної відповіді немає

3. SQL містить функції, що дають змогу обчислювати різні статистичні характеристики. Такі функції називаються:

а) агрегатними
б) публічними
в) приватними
г) правильної відповіді немає

4. Стандарт ANSI визначає певний набір функцій. Скільки функцій містить діалект SQL, що використовується в Oracle?

а) понад 120
б) понад 210
в) 137 функцій
г) правильної відповіді немає

5. Функція, яка повертає суму всіх значень у стовпці, це функція:

а) COUNT
б) SUM
в) AVG
г) MAX

Розділ 9

Обробка запитів за допомогою SQL

- **Фраза GROUP BY**
- **Фраза HAVING**
- **Фраза ORDER BY**
- **Порядок обчислення запитів**
- **Підзапити**
- **Використання предикатів ANY, ALL, EXISTS та IN**
- **Використання теоретико - множинних операторів**
- **Оператори INSERT, UPDATE**

Фраза GROUP BY. Групування таблиці за рядками

Фраза GROUP BY дає змогу поділити множину рядків, одержаних після застосування фрази WHERE (тобто після фільтрації), на групи за ознакою рівності значень в одному чи кількох стовпцях. У цьому випадку агрегатні функції, що використовуються у фразі SELECT, діють не в усьому кінцевому реляційному відношенні, а лише в межах кожної групи. За наявності фрази GROUP BY кожний вираз у списку фрази SELECT повинен набувати єдиного значення для всієї групи, тобто він може бути:

- константою:
- агрегатною функцією;
- таким самим виразом, що й у фразі GROUP BY;
- виразом, що побудований з перелічених вище виразів.

Розглянемо кілька прикладів.

Запит

Визначити кількість студентів на кожній кафедрі факультету інформатики.

SELECT КАФЕДРА. Назва, SUM(ГРУПА.Кількість) Number_Of_Students_ln_The_Departments
FROM ФАКУЛЬТЕТ, КАФЕДРА, ГРУПА
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F **AND**
КАФЕДРА.#D = ГРУПА.#D AND
ФАКУЛЬТЕТ. Назва = "Інформатики"
GROUP BY КАФЕДРА.Назва

Запит

Визначити кількість викладачів на кожному факультеті.

SELECT ФАКУЛЬТЕТ. Назва, COUNT(*) Number_Of_Teachers_In_The_Faculty,**SUM** (ГРУПА. Кількість) Number_Of_STudents_In_The_Departments

FROMФАКУЛЬТЕТ, КАФЕДРА, ВИКЛАДАЧ

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F AND

КАФЕДРА.#D = ВИКЛАДАЧ.#D

GROUP BY ФАКУЛЬТЕТ.Назва

Запит

Для кожного викладача визначити кількість дисциплін, які він читає.

SELECT ВИКЛАДАЧ. Прізвище, COUNT(DISTINCT #S) Number_Of_Subjects

FROM ВИКЛАДАЧ. ЛЕКЦІЯ

WHERE ВИКЛАДАЧ. #T = ЛЕКЦІЯ. #T

GROUP BY ВИКЛАДАЧ. Прізвище

Запит

Отримати фонди факультетів з таблиці ФАКУЛЬТЕТ, а також обчислити їх як суми фондів кафедр.

SELECT ФАКУЛЬТЕТ. Назва, ФАКУЛЬТЕТ. Фонд, SUM (КАФЕДРА. Фонд) Departments_Funds **FROM** ФАКУЛЬТЕТ, КАФЕДРА

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F

GROUP BY ФАКУЛЬТЕТ. Назва, ФАКУЛЬТЕТ. Фонд

Фраза HAVING. Умова вибирання для груп рядків

Фраза HAVING має таке ж значення для груп, що й WHERE для рядка усієї таблиці: вона дає змогу задавати умови для груп рядків, сформованих фразою GROUP BY. Фраза HAVING може використовуватися лише за наявності фрази GROUP BY; вирази в HAVING повинні набувати єдиного значення для групи. В умовах, що формулюються у фразі HAVING, можна використовувати агрегатні функції, які діють у межах створюваних груп, а також стовпці, за якими виконується групування. Розглянемо приклади.

Запит

Визначити факультети, де власний фонд перевищує сумарний фонд усіх кафедр.

SELECT ФАКУЛЬТЕТ. Назва

FROM ФАКУЛЬТЕТ. КАФЕДРА

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА. #F

GROUP BY ФАКУЛЬТЕТ. Назва, ФАКУЛЬТЕТ. Фонд

HAVING ФАКУЛЬТЕТ. Фонд >SUM(КАФЕДРА. Фонд)

Запит

Визначити факультети, в яких власний фонд перевищує сумарний фонд усіх кафедр на 2000

SELECT ФАКУЛЬТЕТ. Назва

FROM ФАКУПЬТЕТ, КАФЕДРА

WHERE ФАКУЛЬТЕТ. #F = КАФЕДРА. #F

GROUP BY ФАКУЛЬТЕТ. Назва, ФАКУЛЬТЕТ. Фонд

HAVING (ФАКУЛЬТЕТ. Фонд - SUM(КАФЕДРА. Фонд)) > 2000

Фраза ORDER BY. Впорядкування рядків

У фразі ORDER BY перелічуються стовпці кінцевої таблиці, за значеннями яких слід відсортувати її рядки, а також встановити порядок цього сортування: за зростанням — ASC чи спаданням — DESC. Зауважимо, що впорядковувати таблиці можна за значеннями лише тих стовпців, які перелічені у фразі SELECT.

За замовчуванням здійснюється впорядкування за зростанням.

Запит

Вивести прізвища викладачів у порядку спадання.

SELECT Прізвище

FROM ВИКЛАДАЧ **ORDER BY**

Прізвище **DESC**

Запит

Вивести в порядку зростання кількості викладачів на всіх кафедрах навчального закладу.

SELECT КАФЕДРА.Назва, COUNT(*) Number_Of_Faculty_Teachers

FROM КАФЕДРА, ВИКЛАДАЧ

WHERE КАФЕДРА. #D = ВИКЛАДАЧ. #D

GROUPBY КАФЕДРА. Назва

ORDER BY Number_Of_Faculty_Teachers

Порядок обчислення запитів

У запитах фрази вживають у певному порядку: SELECT, FROM, WHERE, GROUP BY, ORDER BY. Обчислення запиту здійснюється дещо в інший спосіб:

• обчислюється декартів добуток рядків з таблиць, зазначених у фразі FROM;

• до отриманої єдиної таблиці застосовуються умови з фрази WHERE – які формулюються так, що їх істинність перевіряється на рядку єдиної таблиці;

• отримані рядки ґрупуються відповідно до умови, записаної у фразі GROUP BY;

• до згрупованих рядків застосовуються умови, задані у фразі HAVING - вибираються лише ті ґрупи, що відповідають цим умовам;

• рядки (групи) впорядковуються відповідно до фрази ORDER BY;

• виводяться зазначені у фразі SELECT стовпці чи вирази.

Підзапити

Підзапит - це запит, результат виконання якого є аргументом іншого запиту. Підзапити називають ще вкладеними запитами.Для того щоб вкласти підзапит у запит, потрібно зазначити його у фразі WHERE чи HAVING у правому аргументі одного з таких предикатів: IN, EXISTS =, <>, <, <=, >, >=. Існують прості (незалежні) і корельовані (залежні) вкладені підзапити. Інколи корельовані підзапити називають також інвертованими, підкреслюючи тим самим зворотний напрямок їхнього обчислення.

Простий (незалежний) підзапит — це підзапит, обчислення якого здійснюється незалежно від обчислення зовнішнього запиту. Такі підзапити обробляються «знизу вверх»: першим обробляється вкладений підзапит, а множина значень, отримана в результаті його виконання, використовується під час обробки зовнішнього запиту.

Корельований (залежний, пов'язаний) підзапит — це підзапит, обчислення якого залежить від процесу обчислення в зовнішньому запиті. Такі підзапити обробляються «зверху вниз»: спочатку вибирається поточний рядок із таблиці зовнішнього запиту й за значеннями його полів виконується обчислення підзапиту (тобто під час перевірки умов обчислення підзапиту використовуються значення полів із зовнішнього запиту).

Далі перевіряється умова WHERE щодо входження поточного рядка зовнішнього запиту до результату.

Повернення одного значення

У разі використання предикатів, що порівнюють два значення (=,<>,<; <= ,>,>=), підзапит має повертати одне значення. У протилежному випадку видається повідомлення про помилку. Розглянемо приклади.

Запит

Визначити кафедри вузу, що розташовані в тому ж корпусі, що й кафедра АСУ.

SELECT Назва
FROM КАФЕДРА
WHERE Корпус = (SELECT Корпус
FROM КАФЕДРА
WHERE Назва = "АСУ")

Внутрішній запит визначає корпус кафедри АСУ; він використовується в предикаті порівняння (=) зовнішнього запиту.

У підзапиті можна застосовувати агрегатну функцію, що гарантує повернення єдиного значення.

Запит

Визначити факультети, фонд яких перевищує сумарний фонд усіх кафедр факультету інформатики.

SELECT Назва
FROM ФАКУЛЬТЕТ
WHERE Фонд > (SELECT SUM(КАФЕДРА.Фонд)
FROM ФАКУЛЬТЕТ. КАФЕДРА
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#FAND ФАКУЛЬТЕТ.
Назва = “Інформатики”)

Як уже зазначалося, підзапити можуть вкладатися у фразу HAVING.

Запит

Визначити кафедри, де студентів навчається більше, ніж на кафедрі інженерії програмного забезпечення (ІПЗ).

SELECT КАФЕДРА. Назва. SUM(ГРУПА.Кількість)
FROM КАФЕДРА. ГРУПА
WHERE КАФЕДРА.#D = ГРУПА.#D
GROUP BY КАФЕДРА. Назва
HAVING SUM(ГРУПА. Кількість) >
(**SELECT** SUM(ГРУПА. Кількість)
FROM КАФЕДРА, ГРУПА
WHERE КАФЕДРА.#D = ГРУПА.#DAND КАФЕДРА.Назва =
”ІПЗ")

Повернення багатьох значень

Якщо використовується предикат, що перевіряє належить (IN) чи ні (NOTIN) окреме значення множині, то підзапит може повертати множину значень. Розглянемо приклад.

Запит

Визначити, хто з викладачів факультету інформатики працює також на інших факультетах.

SELECT ВИКЛАДАЧ. ПРІЗВИЩЕ
FROM ФАКУЛЬТЕТ, КАФЕДРА, ВИКЛАДАЧ
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#FAND
КАФЕДРА.#D = ВИКЛАДАЧ.#DAND
ФАКУЛЬТЕТ. Назва = “ інформатики ” AND
ВИКЛАДАЧ.ПРІЗВИЩЕ IN
(**SELECT** ВИКЛАДАЧ. ПРІЗВИЩЕ
FROM ФАКУЛЬТЕТ, КАФЕДРА, ВИКЛАДАЧ
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F AND
КАФЕДРА.#D = ВИКЛАДАЧ.#D AND
ФАКУЛЬТЕТ. Назва = "інформатики")

Корельованість (зв'язаність) підзапиту із запитом

У деяких випадках спосіб обчислення підзапиту залежить від значення поточного рядка зовнішнього запиту. Так, у підзапиті запиту 4.27 у фразі FROM згадується лише реляційне відношення КАФЕДРА, однак у фразі WHERE ми посилаємося на відношення ФАКУЛЬТЕТ. Це означає, що відбувається звернення до зовнішнього запиту. За наявності подібного звернення обчислення здійснюється в такий спосіб:

- фіксується поточний рядок зовнішнього реляційного відношення;
- для обчислення умови фрази WHERE виконується обчислення підзапиту, при цьомувикористовуються значення з поточного рядка зовнішнього запиту.

Обробка корельованого підзапиту має повторюватися для кожного з рядків таблиці, обчисленої у фразі FROM зовнішнього підзапиту, а не виконуватися лише раз, як у незв'язаному підзапиті. Розглянемо кілька прикладів корельованих підзапитів.

Запит

Визначити факультети, фонд кожного з яких менший, ніж сума фондів всіх його кафедр.

SELECT Назва
FROM ФАКУЛЬТЕТ
WHERE Фонд< (SELECT SUM (Фонд)
FROM КАФЕДРА
WHERE КАФЕДРА.#F = ФАКУЛЬТЕТ.#F)

Запит

Визначити викладачів, які не є кураторами груп.

SELECT Прізвище
FROM ВИКЛАДАЧ
WHERE NOT EXISTS (SELECT *
FROM ГРУПА
WHERE ВИКЛАДАЧ.#Т = ГРУПА.#КУРАТОР)

Використання предикатів ANY, ALL, EXISTS та IN

Предикати ANY та ALL

Ключові слова ANY та ALL розміщують після символів однієї з операцій порівняння =,<>,>,<,<=,>=, щоб перевірити, чи є предикат істинним принаймні для одного (для всіх) значень множини, заданої в дужках після слова ANY (ALL), стосовно елементу, записаного зліва від символів порівняння. Розглянемо приклад.

Запит

Визначити кафедри, фонди яких більші, ніж хоча б у однієї з кафедр факультету інформатики.

SELECT Назва
FROM КАФЕДРА
WHERE Фонд>
ANY (SELECT КАФЕДРА. Фонд
FROM ФАКУЛЬТЕТ. КАФЕДРА
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА. #F AND ФАКУЛЬТЕТ. Назва = “Інформатики")

Якби у формулюванні запиту замість слів «хоча б у однієї з» були вжиті слова «у всіх», то під час реалізації запиту мовою SQL замість слова ANY слід було б записати слово ALL.

Предикат EXISTS

EXISTS є одноаргументним предикатом, що повертає значення TRUE, коли підзапит, до якого він застосовується, містить хоча б один рядок.

Запит

Визначити викладачів, які читають хоча б один курс лекцій.

SELECT Прізвище
FROM ВИКЛАДАЧ
WHERE EXISTS (SELECT*
FROM ЛЕКЦІЯ
WHERE ЛЕКЦІЯ.#Т = ВИКЛАДАЧ.#Т)

Якби нас цікавили викладачі, що не читають жодної лекції, то замість слова EXISTS слід було б записати NOT EXISTS.

Предикат IN

Предикат IN перевіряє, чи належить елемент множині. Лівий операнд предиката має бути виразом, результат обчислення якого є окремим значенням (не множиною). Лівий операнд предиката IN може бути константою чи іменем поля. Правий операнд має специфікувати множину, він може бути SELECT - запитом або константою-множиною, зображується взятим у дужки списком своїх елементів, — ("Іванов", "Петров", "Ігнатов"). Вираз x IN ("Іванов", "Петров", "Ігнатов") еквівалентний виразу x = "Іванов" OR x = "Петров" ORx = "Ігнатов". Якщо до предиката IN застосувати заперечення, він матиме вигляд NOTIN.

Розглянемо приклади.

Запит

Визначити факультети, що розташовані в корпусах 1,3,5,11.

SELECT Назва
FROM ФАКУЛЬТЕТ
WHERE КорпусIN (1,3,5,11)

Запит

Визначити факультети, які розташовані в тих самих корпусах, що й факультети інформатики або економіки.

SELECT Назва
FROM ФАКУЛЬТЕТ
WHERE КорпусIN (SELECT КОРПУС
FROM ФАКУЛЬТЕТ
WHERE НАЗВА = "інформатики" OR НАЗВА = "економіки")

Якщо потрібно визначити факультети, розташовані не в тих корпусах, що факультети інформатики й економіки, у даному запиті замість предиката IN слід застосувати предикат NOTIN.

Використання теоретико - множинних операторів

У мові SQL означено три теоретико - множинні оператори - UNION, INTERSECT, EXCEPT, що дають змогу об'єднувати, перетинати й віднімати множини. Аргументами цих операторів є таблиці, що мають бути сумісними. Сумісність таблиць означає, що вони мають однакову кількість стовпців і типи даних відповідних пар стовпців є сумісними. Розглянемо кілька прикладів.

Запит

Вивести прізвища викладачів, які мають лекції в понеділок і вівторок.

SELECT Прізвище
FROM ВИКЛАДАЧ, ЛЕКЦІЯ
WHERE ВИКЛАДАЧ.#Т = ЛЕКЦІЯ.#Т **AND**
ЛЕКЦІЯ. День = "понеділок"
INTERSECT
SELECT Прізвище
FROM ВИКЛАДАЧ,ЛЕКЦІЯ
WHERE ВИКЛАДАЧ.#Т = ЛЕКЦІЯ.#Т **AND**
ЛЕКЦІЯ. День = "вівторок"

Операцію перетину можна також зобразити за допомогою предиката IN.

SELECT Прізвище
FROM ВИКЛАДАЧ, ЛЕКЦІЯ
WHERE ВИКЛАДАЧ.#Т = ЛЕКЦІЯ.#Т **AND**
ЛЕКЦІЯ. День = “ понеділок " **AND**
Прізвище IN (SELECT Прізвище
FROM ВИКЛАДАЧ, ЛЕКЦІЯ
WHERE ВИКЛАДАЧ. #Т = ЛЕКЦІЯ.#Т **AND**
ЛЕКЦІЯ. День = "вівторок")

Запит

Визначити коди викладачів, що читають лекції з курсу «Бази даних», але не читають лекцій з курсу «Програмування».

SELECT t.#Т
FROM ВИКЛАДАЧ t. ЛЕКЦІЯ 1, ПРЕД МЕТ s
WHERE t.#Т = 1.#Т AND 1.#S = s.#S **AND**
s. Назва = "Базиданих"
EXCEPT
SELECT t.#Т
FROM ВИКЛАДАЧ t. ЛЕКЦІЯ 1. ПРЕДМЕТ s
WHERE t.#Т = 1.#Т AND 1.#S = s.#S AND
s.Назва = "Програмування"

Операція різниці може бути зображена також за допомогою предиката NOT IN.

SELECT t.#Т
FROM ВИКЛАДАЧ t. ЛЕКЦІЯ 1. ПРЕДМЕТ S

WHERE t.#T = 1.#T AND 1.#S = s.#S AND
s. Назва - "Базиданих" AND t.#T NOT IN
(SELECT t.#T
FROM ВИКЛАДАЧ t. ЛЕКЦІЯ 1, ПРЕДМЕТ s
WHERE t.#T = 1.#T AND 1.#S = s.#S AND
s. Назва = "Програмування")

Запит

Визначити назви всіх факультетів та кафедр.

SELECT Назва
FROM ФАКУЛЬТЕТ
UNION
SELECT Назва
FROM КАФЕДРА

Запити, в яких реалізується квантор загальності

Стандартна SQL не містить предикатів, що перевіряють чи є одна множина підмножиною іншої. За допомогою цих предикатів можна було б легко записувати запити, еквівалентні тим запитам реляційного числення, в яких використовується квантор V. У SQL для реалізації таких запитів застосовуються інші засоби.

Запит

Визначити викладачів, що читають лекції в усіх групах.

Цей запит можна сформулювати так: «Визначити викладачів, для яких не існує таких груп, де вони не читають лекції». Перша частина запиту «Визначити викладачів, для яких не існує таких груп, де...» реалізується в такий спосіб:

SELECT t. Прізвище
FROM ВИКЛАДАЧ t
WHERE NOT EXISTS (SELECT *
FROM ГРУПА g
WHERE...)

Тепер слід записати вираз, який містить умову щодо групи. Для цього сформулюємо допоміжний запит: «Визначити групи, де не веде занять викладач з кодом X». Він подібний до вихідного запиту (за винятком того, що у даному запиті зазначається конкретний викладач):

SELECT *
FROM ГРУПА g
WHERE #G **NOT IN (SELECT** #G
FROM ЛЕКЦІЯ 1

WHERE 1.#T = “X”*)*

Об'єднаємо обидва запити, вклавши другий запит у перший у такий спосіб:

•фраза WHERE другого запиту замінює фразу WHERE вкладеного підзапиту першого запиту;

•у фразі WHERE другого запиту замість коду конкретного викладача (X) використовується код викладача з першого запиту (t.#T).

У результаті одержимо:

SELECT t.Прізвище
FROM ВИКЛАДАЧ t
WHERE NOT EXISTS
(**SELECT** *
FROM ГРУПА g
WHERE #G **NOTIN (SELECT** #G
FROM ЛЕКЦІЯ 1
WHERE 1.#T = t.#T))

Використання невизначених значень

Якщо під час введення даних не внести значення до поля таблиці, то СКБД розмістить там NULL- значення, або невизначене значення. Аналогічне значення може бути введене до поля таблиці під час виконання операції заміни даних. Мова SQL надає можливість маніпулювати невизначеними значеннями. При цьому NULL - значення не вважається рівним іншому null- значенню, тобто NULL = NULL не дорівнює TRUE (дорівнює логічному NULL), і NOTNULL = NULL) також дорівнює логічному NULL. Незважаючи на це, два невизначені значення розглядаються як дублікати, коли необхідно видалити дублікати, і оператор SELECT DISTINCT дасть у результаті не більше одного NULL-значення. Для запису умов з невизначеними значеннями в SQL введені предикати IS та IS NOT, які можна використовувати з константою NULL. Наприклад, якщо потрібно підрахувати середню величину фонду кафедри за умови, що фонд задано, запит буде таким:

SELECT AVG (Фонд)
FROM КАФЕДРА
WHERE Фонд IS NOT NULL

Коли необхідно визначити кафедри, для яких не вказано величину фонду, потрібно записати:

SELECT Назва
FROM КАФЕДРА

WHERE Фонд **IS NULL**

Засоби маніпулювання даними

Ми розглядали, в який спосіб дані вибираються з БД. Далі ми обговоримо, як дані можна вводити до БД, редагувати та видаляти. Мова йтиме про оператори SQL, що дають змогу маніпулювати даними: INSERT, UPDATE і DELETE

Додавання рядків до таблиці. Оператор INSERT

Оператор INSERT дає змогу вводити дані до БД. Є два різновиди цього оператора:

- INSERT … VALUES
- INSERT … SELECT

Оператор INSERT…VALUES

Цей оператор дає можливість додати до таблиці один рядок і має такий формат:

INSERT INTO <ім'я таблиці> (<поле1>[,<поле2>]…)

VALUES (< значення 1> [, <значення 2>]...)

Виконання цього оператора потребує дотримання певних правил:

- типи даних, що вставляються, мають узгоджуватися з типами даних відповіднихстовпців;
- розміри даних мають відповідати розмірам стовпців;
- порядок даних у фразі VALUES має відповідати порядку стовпців у фразі INSERT INTO. Наведемо кілька прикладів.

Запит. Додати рядок до таблиці ФАКУЛЬТЕТ.

INSERT INTO ФАКУЛЬТЕТ (#F. Назва. Декан. Корпус. Фонд) VALUES (15, "Інформатики", "Сидоров", 5,25000)

Список імен стовпців не є обов'язковим, якщо вставляються значення всіх стовпців. У цьому випадку порядок запису значень має збігатися з порядком стовпців у таблиці.

Запит. Додати рядок до таблиці КАФЕДРА

INSERT INTO КАФЕДРА

VALUES (03, 15, "АСУ", "Петренко", "5", 5500)

Якщо значення стовпців не відомі, то слід використовувати NULL-значення.

Запит. Додати рядок до таблиці ВИКЛАДАЧ.

INSERT INTO ВИКЛАДАЧ

VALUES (173, 13, " Резніченко ", NULL, "526-18-15")

Багато СКБД дають змогу означувати стовпці із властивістю UNIQUE. В межах таблиці значення такого стовпця мають бути

унікальними. Якщо це обмеження порушується, під час додавання рядків можуть виникнути помилки. Зазначимо, що стовпці із властивістю UNIQUE не можуть містити NULL- значень.

Оператор INSERT...SELECT

Цей оператор дає змогу додати до таблиці множину рядків (результат виконання запиту) і, таким чином, дозволяє копіювати інформацію з однієї чи кількох таблиць до іншої. Часто за допомогою оператора INSERT...SELECT дані копіюються до похідних таблиць, що створюються з метою підвищення продуктивності виконання тих чи інших операцій над базою даних. Оператор має такий формат:

INSERT INTO <ім'я таблиці> (<список полів>)

SELECT <список полів>

FROM <ім'я таблиці>

WHERE<умова пошуку>

Вихідні результати стандартного оператора SELECT є вхідними даними для оператора INSERT. Наведемо приклад.

Запит. Додати до таблиці ТИМЧАСОВА, що має стовпці Назва_факультету, Назва_кафедри, Прізвище_викладача, наявні в базі даних відомості про викладачів, а також про кафедри та факультети, де вони працюють.

INSERT INTO ТИМЧАСОВА (Назва_факультету, Назва_кафедри, Прізвище_викладача)

SELECT ФАКУЛЬТЕТ.Назва, КАФЕДРА.Назва, ВИКЛАДАЧ.Прізвище

FROM ФАКУЛЬТЕТ, КАФЕДРА, ВИКЛАДАЧ

WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА. #F**AND**

КАФЕДРА. #D = ВИКЛАДАЧ. #D)

Для оператора INSERT-SELECT мають виконуватися додаткові правила:

• оператор SELECT не може вибирати рядок із таблиці, до якої здійснюється додавання;

• кількість стовпців у фразі INSERT INTO має збігатися з кількістю стовпців у фразі SELECT;

• типи даних стовпців у фразі INSERT INTO мають збігатися з типами даних стовпців у фразі SELECT.

Оновлення даних. Оператор UPDATE

Оператор UPDATE надає можливість змінювати значення у наявних рядках. Його синтаксис такий:

UPDATE <ім'я таблиці>
SET<поле 1> = <вираз 1>
[, <поле2>=<вираз2>]...
[**WHERE** <умова пошуку>]

Оновлення за умовою

Усі рядки таблиці, які задовільняють задану у фразі WHERE умову, змінюються згідно з фразою SET.

Запит. Встановити фонд факультету інформатики рівним 250 300.

UPDATE ФАКУЛЬТЕТ
SET Фонд = 250300
WHERE Назва = "інформатики"

Безумовне оновлення

Якщо фразу WHERE не задано, то оновлюються всі рядки.

Запит

Встановити фонд усіх факультетів рівним 260 500.

UPDATE ФАКУЛЬТЕТ **SET** Фонд = 260500

Не константне оновлення

Стовпцю може присвоюватися не константа, а вираз, що обчислюється на поточному рядку.

Запит. Збільшити всім факультетам фонд фінансування на 10%.

UPDATE ФАКУЛЬТЕТ
SET Фонд = Фонд + Фонд/10

Запит. Збільшити всім кафедрам факультету інформатики фонд фінансування на 5%.

UPDATE КАФЕДРА
SET Фонд = Фонд + Фонд/20
WHERE #F IN (SELECT #F
FROM ФАКУЛЬТЕТ
WHERE ФАКУЛЬТЕТ Назва = "Інформатики")

Контрольні запитання:

1. За наявності фрази GROUP BY кожний вираз у списку фрази SELECT повинен набувати єдиного значення для всієї групи. Напишіть яким значенням він може бути.
2. Напишіть про порядок обчислення запитів.
3. Що таке підзапит?
4. Що таке простий підзапит?
5. Що таке корельований підзапит?

6. Які додаткові умови мають виконуватися додаткові правила для оператора INSERT-SELECT?

7. Яких правил потрібно дотримуватися при виконанні оператора INSERT…VALUES?

8. Що таке не константне оновлення?

Тестові завдання:

1.Фраза GROUP BY дає змогу:

а) поділити множину рядків на групи

б) видалити множину рядків з групи

в) створити групу

г) правильної відповіді немає

2. Фраза HAVING може використовуватися:

а) лише за наявності фрази GROUP BY

б) завжди

в) ніколи

г) правильної відповіді немає

3. У разі використання предикатів, що порівнюють два значення (=,<>,<; <= ,>,>=), скільки має повертати значень підзапит?

а) безліч

б) одне

в) два

г) правильної відповіді немає

4.Який оператор дає змогу додати до таблиці множину рядків?

а) INSERT – VALUES

б) UPDATE

в) INSERT-SELECT

г) правильної відповіді немає

5. Який предикат перевіряє,чи належить елемент множині?

а) ANY

б) ALL

в) EXISTS

г) IN

Розділ 10

Додаткові можливості мови SQL

- **Оператор DELETE**
- **Операції над схемою бази даних**
- **Оператори: CREATE DATABASE, CREATE TABLE, ALTER TABLE, DROP TABLE, DROP DATABASE**
- **Віртуальні таблиці та індекси**
- **Додаткові можливості мови**

Видалення рядків таблиці. Оператор DELETE

Оператор DELETE дає змогу видаляти рядки таблиці й має такий синтаксис:

DELETE FROM <ім'я таблиці>

[**WHERE** умова]

Залежно від наявності та змісту фрази WHERE можна видалити один рядок, множину рядків, усі рядки або не видалити жодного.

Назвемо кілька особливостей використання оператора DELETE.

• Оператор не дає змоги видаляти, окремі поля (використовуйте для цього оператор DATE), видаляючи рядок повністю.

• Застосування оператора DELETE, як і INSERT та UPDATE, може призвести до порушення цілісності бази даних.

• Якщо у фразі WHERE використовується вкладений підзапит, то у фразі FROM цього підзапиту не можна зазначати таблицю, з якої видаляються рядки. Це стосується також операторів INSERT та UPDATE.

• Оператор видаляє лише рядки таблиці, а не саму таблицю. Для видалення всієї таблиці слід застосувати оператор DROP TABLE. Наведемо приклади.

Запит.Видалити відомості про лекції, що читаються в суботу та неділю.

DELETE FROM ЛЕКЦІЯ

WHERE День IN ("субота", “неділя")

Запит. Видалити всі рядки з реляційного відношення ПРЕДМЕТ.

DELETE FROM ПРЕДМЕТ

Запит. Видалити з таблиці ПРЕДМЕТ ті предмети, з яких не читається лекцій.

DELETE FROM ПРЕДМЕТ

WHERE #S!= **ALL (SELECT** #S **FROM** ЛЕКЦІЯ)

Операції над схемою бази даних

У цьому підрозділі ми вивчимо можливості мови SQL щодо створення баз даних і таблиць, а також розглянемо основні операції маніпулювання даними: зміна структури таблиці, додавання та видалення стовпців і рядків, видалення таблиці та бази даних.

Створення бази даних. Оператор CREATE DATABASE

Операція створення бази даних може бути або елементарною дією, або ж вимагати складних маніпуляцій, залежно від потреб користувача та обраної СКБД. Багато сучасних СКБД оснащені графічними засобами, що дають змогу визначати схему бази, проте універсальним способом створення бази даних є застосування операторів SQL. Синтаксис типового оператора створення бази даних є таким:

CREATE DATABASE <ім’я бази даних>

Оскільки синтаксис цього оператора змінюється залежно від системи, ми його не будемо деталізувати. У більшості СКБД оператор CREATE DATABASE дає змогу виконати такі основні дії:

- делегувати повноваження на створення бази даних тому чи іншому користувачу;
- визначити місце (диск, каталог), де розташовуватиметься база даних;
- зарезервувати певний обсяг дискового простору для подальшого зберігання рядків таблиць та іншої інформації.

Створення таблиці. Оператор CREATE TABLE

Процедура створення таблиці є більш стандартизованою. Базовий синтаксис відповідного оператора є таким:

CREATE TABLE <ім'я таблиці>

(<поле 1><тип даних 1>[**NOT NULL**]

[, <поле 2><тип даних 2> [**NOT NULL**]]...)

Після ключових слів CREATE TABLE задаються ім'я таблиці та імена стовпців з типами даних і деякими іншими характеристиками. Специфікатор NOT NULL забороняє введення у стовпець NULL - значень і застосовується зокрема до ключових полів.

Типи даних

Кожна система має свої типи даних. Наведемо типи даних у СКБД Oracle.

Таблиця 10.1 – Типи даних у СКБД Oracle

Тип даних	Коментарі
CHAR	Текстові рядки довжиною від 1 до 255 символів. Довжина рядків фіксована
VARCHAR2	Текстові рядки довжиною від 1 до 2000 символів. Довжина рядків змінна
DATE	Дата: рік, місяць, день, година, хвилина, секунда
LONG	Рядок символьних даних змінної довжини до 2 Гбайт
LONGRAW	Двійкові дані довжиною до 2 Гбайт
NUMBER	Додатні чи від'ємні числа з фіксованою чи плаваючою комою
RAM	Двійкові дані довжиною до 255 байтів
R0WID	Шістнадцятковий рядок, з унікальною адресою рядка в таблиці.

Унікальні значення

Певні системи надають можливість зазначати, що те чи інше поле має містити унікальні (не повторювані) значення. Це особливо важливо для ключових полів. Інші системи, наприклад Oracle та SQLServer, дають змогу оголошувати унікальні (UNIQUE) індекси. Є системи, в яких визначено тип даних, та його значення автоматично надаються створюваним рядкам. Ці значення є унікальними в межах таблиці протягом усього часу її існування. В Oracle таким типом даних є ROWID. Окрім того, в Oracle модифікатор UNIQUE застосовується для позначення полів, що разом забезпечують унікальність записів у межах таблиці.

Запит.Створити таблицю ФАКУЛЬТЕТ (застосовуємо типи даних Oracle).

CREATE TABLE ФАКУЛЬТЕТ
(#F NUMBER (10).
Назва CHAR(SO) NOT NULL.
Декан CHAR(25),
Корпус CHARC5),
Фонд NUMBER(6,2))

Створення таблиці на базі існуючої

Для створення таблиць найчастіше використовують команду CREATE TABLE. Однак деякі системи надають альтернативний спосіб

означення таблиць із використанням формату та даних наявної таблиці. Цей метод корисний для вибирання даних із таблиці з метою тимчасового зберігання та модифікації. В СКБД Oracle синтаксис цієї команди виглядає так:

CREATE TABLE <нова таблиця> (<список полів>)

AS (SELECT <список полів>

FROM <стара таблиця> [**WHERE**..])

Ця команда дає змогу створити нову таблицю з типами даних полів наявної таблиці та за необхідності перейменувати поля.

Модифікація таблиці. Оператор ALTER TABLE

Можливі ситуації, коли вже створена таблиця не відповідає зміненим вимогам до бази даних. Команда ALTER TABLE дає змогу змінити структуру таблиці після її створення: додати до вже визначеної таблиці стовпець або змінити наявний.

Синтаксис команди такий:

ALTER TABLE <ім’я таблиці>

[**ADD** <поле><означення поля>|

ALTER <поле><параметри>|

DROP <поле><параметри>]

Видалення таблиці. Оператор DROP TABLE

У SQL є команда, призначена для видалення з бази даних усієї таблиці. Оператор DROP TABLE видаляє таблицю з усіма зв'язаними з нею віртуальними таблицями й індексами. Найчастіше він застосовується до тимчасових таблиць, які видаляються через певний час після створення.

Синтаксис команди такий:

DROP TABLE<ім'я таблиці>

Видалення бази даних. Оператор DROP DATABASE

Команда DROP DATABASE застосовується для видалення всієї бази даних і має такий синтаксис:

DROP DATABASE<ім'я бази даних>

Віртуальні таблиці та індекси

У цьому підрозділі ми розглянемо два засоби SQL, які дають змогу зображувати дані не в тому вигляді, в якому вони зберігаються в БД. Мова йтиме про віртуальні таблиці та індекси.

Використання віртуальних таблиць

Віртуальна таблиця - це пойменована таблиця, одержана в результаті виконання оператора SELECT з можливою заміною імен стовпців.

Віртуальна таблиця застосовується для створення складних запитів. Вона може використовуватись в операторах SELECT, INSERT, UPDATE, DELETE, хоча фізично не займає місця в базі даних, як звичайна таблиця. Синтаксис створення віртуальної таблиці такий:

CREATE VIEW<віртуальна таблиця> [(<список полів>)] AS
SELECT<список полів>
FROM<імена таблиць>
[WHERE ...]
[WITH CHECK OPTION]

Необов'язкова фраза WITH CHECK OPTION (з контролем) вказує на те, що для операцій INSERT та UPDATE над цією таблицею має здійснюватися контроль, який забезпечує виконання умов, заданих у фразі WHERE підзапиту.

Проста віртуальна таблиця

Це віртуальна таблиця, що є копією вихідної, але має інше ім'я:

CREATE VIEW КОПІЯ_ФАКУЛЬТЕТУ AS
SELECT*
FROM ФАКУЛЬТЕТ

У результаті виконання наведеного оператора створюється віртуальна таблиця КОПІЯ_ФАКУЛЬТЕТУ, що є точною копією таблиці ФАКУЛЬТЕТ. Можна також створювати віртуальні таблиці на базі інших віртуальних таблиць.

Вибір стовпців

До віртуальної таблиці можна копіювати окремі стовпці вихідної таблиці:

CREATE VIEW ДЕКАНИ_ФАКУЛЬТЕТІВ (Назва.Декан) AS
SELECT *
FROM ФАКУЛЬТЕТ

Складні конструкції

Фраза WHERE у складі оператора CREATE VIEW може містити підзапити. Наведемо приклад.

Запит. Створити віртуальну таблицю зі списком викладачів факультету інформатики, які працюють також на інших факультетах.

CREATE VIEW ВИКЛАДАЧ_2 (Прізвище, Посада)
AS SELECT ВИКЛАДАЧ. Прізвище. Посада

FROM ФАКУЛЬТЕТ, КАФЕДРА, ВИКЛАДАЧ
WHERE ФАКУЛЬТЕ.#F = КАФЕДРА.#F**AND**
КАФЕДРА.#D = ВИКЛАДАЧ.#D**AND**
ФАКУЛЬТЕТ.Назва = "інформатики" **AND**
ВИКЛАДАЧ. Прізвище **IN**
(**SELECT** ВИКЛАДАЧ. Прізвище
FROM ФАКУЛЬТЕТ, КАФЕДРА. ВИКЛАДАЧ
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F**AND**
КАФЕДРА.#D = ВИКЛАДАЧ.#D
ФАКУЛЬТЕТ. Назва <>"інформатики")

Обмеження, накладені на структуру запиту в операторі CREATE VIEW.

SQL накладає певні обмеження на структуру запиту, що використовується для означення віртуальної таблиці: в ньому забороняється застосовувати оператор UNION фразу ORDER BY.

Зміна даних через віртуальні таблиці

Віртуальні таблиці можна використовувати в операторах UPDATE, INSERT, DELETE для зміни даних у БД, але в цьому разі слід дотримуватися таких правил:

•забороняється застосовувати оператор DELETE до віртуальних таблиць, визначених на багатьох базових таблицях;

•директиву INSERT дозволяється використовувати лише в тому випадку, якщо віртуальнатаблиця містить усі значення NOT NULL - стовпця базової таблиці;

•під час додавання чи оновлення через віртуальну таблицю всі записи, що оновлюються, мають належати одній і тій самій фізичній таблиці;

•забороняється додавати чи оновлювати записи через віртуальну таблицю, визначену іззастосуванням модифікатора DISTINCT;

•не дозволяється оновлювати віртуальні стовпці (тобто стовпці, значення в яких є результатами обчислення виразів).

Застосування віртуальних таблиць

Віртуальна таблиця може використовуватися для захисту, конвертування й забезпечення логічної незалежності даних, а також для спрощення складних запитів.

•Забезпечення логічної незалежності. Одне з основних завдань, яке дають змогувирішити віртуальні таблиці, полягає в забезпеченні незалежності користувацьких програм від змін у логічній структурі

бази даних, зумовлених розширенням таблиці та/або зміною розташування стовпців (наприклад, підчас розщеплення таблиць).

•Захист даних. Надаючи користувачам доступ до певної інформації лише через віртуальнітаблиці, можна вирішити проблему захисту даних.

•Конвертування даних. Віртуальні таблиці стають у нагоді, коли постає потреба падатикористувачу дані не в тому форматі, в якому вони зберігаються в базі даних.

•Спрощення конструкцій складних запитів. Складний запит, що має ієрархічну структуру, легше сформулювати, використовуючи віртуальні таблиці як допоміжні.

Видалення віртуальної таблиці

Для видалення віртуальної таблиці застосовується оператор DROP VIEW. Його синтаксис такий:

DROP VIEW<Ім'я віртуальної таблиці>

Слід пам'ятати, що під час видалення віртуальної таблиці видаляються також всі інші таблиці, у визначеннях яких дана таблиця використовувалась.

Використання індексів

Одним зі способів подання даних не в тому порядку, в якому вони зберігаються, є використання індексів. Вони забезпечують:

• задоволення вимоги унікальності записів:

• підтримку логічної упорядкованості даних відповідно до значень одного чи кількох полів;

• оптимізацію виконання запитів.

З погляду користувача, індекс — це перелік стовпців таблиці, за значеннями яких записи логічно впорядковуються. З погляду СКБД, індекс — це механізм, що дає змогу значно підвищити швидкість доступу до записів за індексованими полями та забезпечує ефективну перевірку унікальності значень індексованих полів.

Визначення індексу

Базовий синтаксис оператора визначення індексу є таким:

CREATE INDEX<ім'я індексу>

ON <ім'я таблиці> (<поле 1> [.<поле 2>1...)

У багатьох СКБД оператор визначення індексу доповнюється іншими конструкціями.

Правила використання індексів

Використовуючи індекси, варто брати до уваги такі міркування.

• У таблицях невеликих розмірів індекси майже не забезпечують підвищення продуктивності.

• Продуктивність значно підвищується в тих випадках, коли стовпці містять переважно неповторювані дані чи багато NULL-значень.

• Завдяки індексам оптимізується виконання запитів, що видають невелику кількість рядків (до 25 %).

• Пам'ятайте, що індекси прискорюють пошук даних, однак сповільнюють процес їхньогооновлення, що стає особливо відчутним під час одночасного оновлення великої кількості рядків. У подібних випадках перед оновленням індекс потрібно видаляти, а після завершення даної операції - відновлювати.

• Зберігання індексів потребує значних обсягів пам'яті. Якщо СКБД дає змогу керуватипам'яттю, слід відвести частину пам'яті під індекси.

• Потрібно завжди індексувати поля, що використовуються для з'єднання таблиць. Цезначно прискорює виконання запитів.

• Не слід індексувати поля, які регулярно оновлюються.

• Не бажано зберігати індекси разом із таблицями на одному фізичному пристрої. Розподіл цих об'єктів між носіями інформації знижує навантаження на них та прискорює виконання запитів.

Складені індекси

SQL дає змогу створювати індекс за кількома полями. Наприклад, оператор

CREATE INDEX КАФЕДРА_ЗАВІДУВАЧ_НАЗВА
ON КАФЕДРА (#ЗАВІДУВАЧ. Назва)

створює індекс у таблиці КАФЕДРА за полями #ЗАВІДУВАЧ і Назва.

У складених індексах слід спочатку зазначати поля, що використовуються найчастіше. Складені індекси потрібно застосовувати тоді, коли зазначені в них поля використовуються для опису умови вибирання даних.

Використання фрази UNIQUE

Фраза UNIQUE вказує, що значення індексу мають бути унікальними. Наприклад, оператор

CREATE UNIQUE INDEX ГРУПА_ІД
ON ГРУПА (#G)

вимагає, щоб у таблиці ГРУПА значення поля #Gбули неповторюваними.

Порядок сортування полів у індексі

У деяких СКБД надається можливість зазначати порядок сортування полів, щоіндексуються. Наприклад, оператор

CREATE INDEX ФАКУЛЬТ_НАЗВА_ДЕКАН

ON ФАКУЛЬТЕТ (Назва DESC. Декан)

вказує на необхідність індексування таблиці ФАКУЛЬТЕТ за стовпцем Назва у порядку спадання, а за стовпцем Декан - у порядку зростання. За замовчуванням сортування здійснюється в порядку зростання.

Видалення індексу

Видалення індексу виконується командою DROP INDEX, що має такий синтаксис:

DROP INDEX < ім’я індексу>

Транзакції

Транзакція - це сукупність операцій з маніпулювання базою даних, що мають розглядатися як атомарна дія. Під терміном «атомарна дія» ми маємо на увазі таку дію, що повністю виконується або скасовується як єдине ціле, тобто щодо неї підтримується принцип «усе або нічого». Різні СКБД мають свою специфіку щодо підтримки транзакцій. Тому ми використовуватимемо гіпотетичний узагальнений синтаксис таких операцій.

Початок і завершення транзакції

Транзакція активізується командою:

BEGIN TRANSACTION

[<ім’я транзакції>]

і завершується командою

COMMIT [TRANSACTION]

за якою відбувається фактична зміна стану бази даних згідно з командами, що містяться в транзакції. Наприклад, якщо є така транзакція:

BEGIN TRANSACTION

INSERT INTO ФАКУЛЬТЕТ VALUES (1, "інформатики". "Іванов". "2/b", 42000)

INSERT INTO ФАКУЛЬТЕТ VALUES (1, "економіки", "Петров". "3/a", 47000)

COMMIT

то в таблицю ФАКУЛЬТЕТ будуть вставлені або два рядки, або жодного - залежно від успішності виконання всієї транзакції.

Скасування транзакції. Точки збереження

Після здійснення транзакції зазвичай надається можливість з'ясувати, чи успішним було її виконання. Транзакцію можна скасувати навіть у разі її успішного виконання, застосувавши команду ROLLBACK (але вона має бути виконана до команди COMMIT). У цьому випадку база даних набуває того ж вигляду, який вона мала до виконання транзакції. Можна також вказати так звані точки збереження (save-point), або контрольні точки. Загальний синтаксис команди ROLLBACK є таким:

ROLLBACK [TRANSACTION] [TOSAVEPOINT<Ім'я точки збереження>]

Розглянемо приклад.

BEGIN TRANSACTION

INSERT INTO ФАКУЛЬТЕТ **VALUES** (1 "інформатики", "Іванов". Ж 42000) **ROLLBACK TRANSACTION**

INSERT INTO ФАКУЛЬТЕТ **VALUES** (1, "економіки". "Петров". Ж 47000) **COMMIT**

У таблицю ФАКУЛЬТЕТ буде занесено лише відомості про факультет економіки, оскільки після додавання першого рядка відбувається скасування транзакції (додавання першого рядка анулюється).

Після застосування команди COMMIT усі дії, що є в транзакції, виконуються. Відміна транзакції призводить до скасування всіх дій, виконаних до моменту застосування **ROLLBACK TRANSACTION.**

За допомогою **точок збереження** можна скасовувати не всі виконані з початку транзакції дії, а лише їх частину. Синтаксис оголошення точки збереження є таким:

SAVEPOINT <ім'я точки збереження>

Розглянемо приклад.

BEGIN TRANSACTION

UPDATE КАФЕДРА

SET Фонд = Фонд + Фонд/10 WHERE Корпус = "2/3"

SAVEPOINT save_it

DELETE FROM КАФЕДРА WHERE Корпус = "2/3"

ROLLBACK TO SAVEPOINT save_it

COMMIT

Видалення рядків не відбудеться, оскільки здійснюється повернення до точки збереження save_it, що означена до команди видалення.

Тригери

Тригер - це SQL-оператор, що активізується під час виконання певних операцій над об'єктами бази даних. Об'єктами бази даних є таблиці, а операціями — додавання, видалення та заміна рядків. Тригери — це один із механізмів підтримки цілісності бази даних.

У найпростішому випадку синтаксис оголошення тригера є таким:

CREATE TRIGGER < ім'я тригера>

{BEFORE I AFTER}<операції над таблицею> [OF <список полів>] ON <ім'я таблиці>

[**WHEN** (<умова>)] <оператори SQL>

Якщо умова у фразі WHEN є істинною або ця фраза відсутня, до (BEFORE) або після (AFTER) виконання операції INSERT, UPDATE чи DELETE над таблицею, зазначеною після слова ON, буде виконано вказані нижче оператори SQL. Коли таких операторів кілька, їх слід помістити між ключовими словами BEGIN ATOMIC та END. Конструкція другого рядка означення тригера **називається реченням ініціювання**.WHEN - **умовою ініціювання**, а <оператори SQL> - **дією тригера**.

Розглянемо приклади застосування тригерів.

Запит

Після видалення інформації про кафедру видалити інформацію про всіх викладачів кафедри.

CREATE TRIGGER Кафедрat_Видалення
AFTER DELETE ON КАФЕДРА
DELETE FROM ВИКЛАДАЧ
WHERE ВИКЛАДАЧ.## = КАФЕДРА #D

Запит

Після видалення рядка з відомостями про викладача встановити значення NULL в полі #Куратор тих записів із таблиці ГРУПА, що відповідають групам, де згаданий викладач був куратором.

CREATE TRIGGER Викладач_Видалення
AFTER DELETE ON ВИКЛАДАЧ
UPDATE ГРУПА
SET #Куратор =NULL
WHERE ГРУПА.#Т = ВИКЛАДАЧ.#Т

Доступ до старих і нових значень рядків

Якщо виконується оновлення рядків таблиці, то в тригері допускається звернення до старих і нових значень рядків, що

оновлюються. Для звернення до нового (старого) рядка слід вказати кваліфікатор NEW (OLD) перед іменем стовпця. Такі кваліфікатори дозволяється використовувати як в умові тригера, так і в описі його дії. Розглянемо приклади.

Запит. Після додавання чи оновлення рядка в таблиці КАФЕДРА встановити значення поля Фонд у таблиці ФАКУЛЬТЕТ рівним сумі фондів усіх кафедр відповідного факультету.

CREATE TRIGGER ФАКУЛЬТЕТ_Фонд
AFTER INSERT OR UPDATE ON КАФЕДРА
BEGIN
UPDATE ФАКУЛЬТЕТ
SET Фонд = **SELECT SUM** (Фонд)
FROM КАФЕДРА
WHERE КАФЕДРА.#F = КАФЕДРА. NEW.#F
AND ФАКУЛЬТЕТ.#F = КАФЕДРАРА.NEW.#F

Тригери й транзакції

Дії, які виконуються під час основної операції та в тригері, становлять єдину транзакцію.

1. Перед виконанням додавання, оновлення та видалення неявно ініціюється команда BEGIN TRANSACTION.

2 Реалізується операція додавання/оновлення/видалення.

3. Ініціюється та виконується тригер;

4. Тригер скасовує транзакцію або за замовчуванням його дія завершується.

Запит. Дозволити додавання рядка з відомостями про кафедру лише в тому випадку, коли існує рядок із даними про факультет, якому належить кафедра (кафедри без факультету не буває).

CREATE TRIGGER Кафедра_Додавання **BEFORE INSERT ON** КАФЕДРА
WHEN NOT EXISTS (SELECT*
FROM ФАКУЛЬТЕТ
WHERE ФАКУЛЬТЕТ.#F = КАФЕДРА.#F **BEGIN**
ROLLBACK TRANSACTION END

Вкладеність тригерів

Тригери бувають вкладеними. Це означає, що маніпулювання рядком однієї таблиці може ініціювати тригер, який маніпулює рядками іншої таблиці. У свою чергу, маніпулювання рядками другої таблиці може ініціювати тригер, що маніпулює рядками третьої таблиці тощо.

Вкладеність тригерів може призвести до «зациклення». Обмеження на використання тригерів:

• тригери не визначаються для віртуальних таблиць;

• після видалення таблиці всі зв'язані з нею тригери також видаляються;

• тригери визначаються лише для створених таблиць.

В усіх наведених прикладах тригери використовувалися для підтримки цілісності бази даних. Але існує багато інших можливостей використання тригерів. Наведемо лише деякі з них:

- автоматичне обчислення значень похідних стовпців;
- запобігання виконанню недозволених транзакцій;
- виконання складних перевірок з метою захисту даних;
- підтримка складних обмежень цілісності;
- реєстрація подій, що відбуваються в базі даних;
- збирання статистики щодо доступу до таблиць бази даних.

Додаткові можливості

Тимчасові таблиці

Деякі СКБД надають можливість створювати тимчасові таблиці, які є звичайними таблицями, що існують лише до завершення сеансу роботи користувача з базою даних. Як правило, такі таблиці входять до складу спеціальної службової бази даних, призначеної для їхнього зберігання. Синтаксис директив для роботи з тимчасовими таблицями аналогічний синтаксису відповідних команд для звичайних таблиць, за винятком того, що є конструкції, призначені для створення тимчасових таблиць. Тимчасові таблиці потрібні для зберігання тимчасових даних, які потрібні лише протягом поточного сеансу зв'язку із СКБД.

Курсори

Курсор є покажчиком на окремий запис тієї чи іншої таблиці. Курсори використовуються переважно в збережених процедурах, а також у програмах, що працюють з базою даних й ініціюють виконання операторів SQL. Використання курсорів дає можливість:

- вибрати певну сукупність даних (на зразок тимчасової таблиці, що містить результати виконання оператора SELECT);
- ініціювати послідовний перегляд сукупності записів;
- проаналізувати конкретний запис, на який вказує курсор;
- виконати зовнішню операцію над поточним записом перш ніж перейти до наступного.

Ще одним варіантом застосування курсору є тимчасове зберігання результатів запиту для подальшого використання. Якщо зовнішня програма вимагає багаторазового використання певного набору записів, то створюється курсор, яким оперують замість багаторазового виконання оператора SELECT. Для використання курсору потрібно:

• створити курсор;

• відкрити курсор для використання в застосуванні чи збереженій процедурі;

• ініціювати за допомогою курсору послідовне перебирання записів з метою їхньої обробки:

• закрити курсор після завершення роботи з ним (з можливим наступним відкриттям у разіпотреби);

• видалити курсор.

На відміну від таблиць, індексів, тригерів та збережених процедур, курсори не є об'єктами бази даних.

Збережені процедури

Збережені процедури - це об'єкти бази даних, які зазвичай містять велику кількість директив SQL. Такі процедури можуть містити сукупність команд (складних запитів, операцій оновлення, додавання та видалення), що часто використовуються як єдине ціле. Збережена процедура дає змогу звернутися до неї як до функції, замість того, щоб послідовно записувати команди SQL, які вона містить. Значне зниження навантаження на канали зв'язку під час роботи користувачів із сервером теж є важливою перевагою збережених процедур, адже замість окремих багаторазових звернень до бази даних виконується єдине звернення до збереженої процедури.

Розширення SQL

Будь-яка СКБД розширює стандарт мови SQL, причому в багатьох випадках SQL трансформується в мову програмування, що надає усі можливості SQL. Наприклад, у MicrosoftSQLServer таким розширенням є мова Transact-SQL. в Oracle — мова PL/SQL. Кожна з них має переваги звичайних мовпрограмування, підтримує всі можливості стандарту ANSISQL, а також розширює його.

Крім того, багато СКБД надають засоби для застосування SQL у традиційних мовах програмування. Існують два методи такого використання мови.

Статична SQL припускає вкладання безпосередньо в програмний код речення, що не може бути змінене під час виконання програми.

Зазвичай статична SQL вимагає використання предкомпілятора. Наприклад, Oracle має предкомпілятори для використання SQL у мовах C, Pascal, Ada, COBOL та FORTRAN.

Динамічна SQL дає змогу програмувати побудову SQL- запитів, що створюються під час виконання програми і передаються СКБД, яка, в свою чергу, передає знайдені дані змінним програми.

Контрольні запитання:

1. Назвіть кілька особливостей використання оператора DELETE?
2. Що таке транзакція?
3. Що таке тригер?
4. Що потрібно для використання курсорів?
5. Які можливості надає використання курсорів?
6. Що таке збережені процедури?
7. Що таке динамічна SQL?
8. Що таке статична SQL?

Тестові завдання:

1.Яка команда застосовується для видалення всієї бази даних?
а) DROP DATABASE
б) DELETE Ф
в) CREATE table
г) Правильної відповіді немає

2. Які два засоби SQL, які дають змогу зображувати дані не в тому вигляді, в якому вони зберігаються в БД?
а) віртуальні таблиці та індекси
б) індекси та курсори
в) віртуальні таблиці та курсори
г) Правильноївідповіднемає

3. Який з перелічених нижче оперторів є оператором для створення бази даних?
а) CREATE TABLE
б) CREATE DATABASE
в) DROP DATABASE
г) Правильної відповіді немає

4.Який з перелічених нижче оперторів є оператором для модифікації таблиць?

а) ALTER TABLE

б) CREATE TABLE

в) DROP TABLE

г) Правильної відповіді немає

5. Скільки існує методів, які надають можливість застосування SQL у традиційних мовах програмування?

а)Один (статична SQL)

б) Один (динамічна SQL)

в) Два (статична SQL та динамічна SQL)

г) Правильної відповіді немає

Розділ 11

Проектування і використання баз даних

- **Проектування баз даних**
- **Надмірне дублювання даних**
- **Аномалії**
- **Формування початкового відношення**
- **Метод нормальних форм**

Проектування баз даних

Проблеми проектування

Проектування інформаційних систем, що включають бази даних, здійснюється на фізичному і логічному рівнях. Рішення проблем проектування на фізичному рівні багато в чому залежить від СКБД, що використовується, часто автоматизовано і приховано від користувача. У ряді випадків користувачу надається можливість настройки окремих параметрів системи, що не складає великої проблеми.

Логічне проектування полягає у визначенні числа і структури таблиць, формуванні запитів до БД, визначенні типів звітних документів, розробці алгоритмів обробки інформації, створенні форм для введення і редагування даних в базі і рішенні ряду інших задач.

Рішення задач логічного проектування БД в основному визначається специфікою задач наочної області. Найважливішою тут є проблема структуризації даних.

При проектуванні **структур даних** для автоматизованих систем можна виділити три основні підходи:

1. Збір інформації про об'єкти задачі в рамках однієї таблиці (одного відношення) і подальша декомпозиція її на взаємозв'язані таблиці на основі процедури нормалізації відношень.

2. Формування знань про систему (визначення типів початкових даних і їх взаємозв'язків) і вимог до обробки даних, отримання за допомогою CASE-системи (системи автоматизації проектування і розробки баз даних) готової схеми БД або навіть готової прикладної інформаційної системи.

3. Структуризація інформації для використання в інформаційній системі в процесі проведення системного аналізу на основі сукупності правил і рекомендацій.

Розглянемо перший з названих підходів, він є класичним і історично першим. Перш за все охарактеризуємо основні проблеми, що мають місце при визначенні структур даних у відношеннях реляційної моделі.

Надмірне дублювання даних і аномалії

Слід розрізняти просте (ненадмірне) і надмірне дублювання даних. Наявність першого з них допускається в базах даних, а надмірне дублювання даних може приводити до проблем при обробці даних. Наведемо приклади обох варіантів дублювання.

Приклад, **ненадмірного дублювання** даних представляє відношення С_Т з атрибутами Співробітник і Телефон. Для співробітників, що знаходяться в одному приміщенні, номери телефонів співпадають. Номер телефону 4328 зустрічається кілька разів, хоча для кожного службовця номер телефону унікальний. Тому жоден з номерів не є надмірним. Дійсно, при видаленні одного з номерів телефонів буде загублена інформація про те, по якому номеру можна додзвонитися до одного із службовців.

С_Т (Співробітник_Телефон)

Співробітник	Телефон
Іванов	3721
Петров	4328
Сидоров	4328
Єгоров	4328

Ненадмірне дублювання

Приклад **надмірного дублювання (надмірності)** представляє відношення С_Т_Н, яке, на відміну від відношення С_Т, доповнено атрибутом Н_кімн (номер кімнати співробітника). Природно припустити, що всі службовці в одній кімнаті мають один і той же телефон. Отже, в даному відношенні є надмірне дублювання даних. Так, у зв'язку з тим, що Сидоров і Єгоров знаходяться в тій же кімнаті, що і Петров, їх номери можна взнати з кортежу з відомостями про Петрова.

а)

С_Т_Н (Співробітник_Телефон_Номер кімнати)

Співробітник	Телефон	Н_кім
Іванов	3721	109
Петров	4328	111
Сидоров	4328	111
Єгоров	4328	111

б)

С_Т_Н (Співробітник_Телефон_Номер кімнати)

Співробітник	Телефон	Н_кім
Іванов	3721	109
Петров	4328	111
Сидоров	—	111
Єгоров	—	111

Надмірне дублювання

На мал. вище наведений приклад невдалого відношення С_Т_Н, в якому замість телефонів Сидорова і Єгорова поставлені прочерки (невизначені значення).

Невдалість подібного способу виключення надмірності полягає в наступному. По-перше, при програмуванні доведеться витратити додаткові зусилля на створення механізму пошуку інформації для прочерків таблиці. По-друге, пам'ять все одно виділяється під атрибути з прочерками, хоча дублювання даних і виключено. По-третє, що особливо важливо, при виключенні з колективу Петрова кортеж з відомостями про нього буде виключений з відношення, а значить знищена інформація про телефон 111-й кімнати, що неприпустимо.

Можливий спосіб виходу з даної ситуації приведений на мал. нижче. Тут показані два відношення С_Н і Н_Т, отримані шляхом декомпозиції початкового відношення С_Т_Н. Перше з них містить інформацію про номери кімнат, в яких розташовуються співробітники, а друге - інформацію про номери телефонів в кожній з кімнат. Тепер, якщо Петрова і звільнять з установи і, як наслідок цього, видалять всяку інформацію про нього з баз даних установи, це не приведе до втрати інформації про номер телефону в 111-й кімнаті.

Т_Н (Телефон_Номер кімнати) С_Н (Співробітник_Номер кімнати)

Телефон	Н_кім
3721	109
4328	111

Співробітник	Н_кімн
Іванов	109
Петров	111
Сидоров	111
Єгоров	111

Виключення надмірного дублювання

Процедура декомпозиції відношення С_Т_Н на два відношення С_Н і Н_Т є основною процедурою нормалізації відношень.

Надмірне дублювання даних створює проблеми при обробці кортежів відношення, названі Е. Коддом «аномаліями оновлення відношення». Він показав, що для деяких відношень проблеми виникають при спробі видалення, додавання або редагування їх кортежів.

Аномаліями називатимемо таку ситуацію в таблицях БД, яка приводить до суперечностей в БД, або істотно ускладнює обробку даних. Виділяють три основні види аномалій: аномалії модифікації (або редагування), аномалії видалення і аномалії додавання. **Аномалії модифікації** виявляються в тому, що зміна значення одного даного може спричинити за собою перегляд всієї таблиці і відповідну зміну деяких інших записів таблиці.

Так, наприклад, зміна номера телефону в кімнаті 111, що представляє собою один єдиний факт, приводить до необхідності перегляд всієї таблиці С_Т_Н і зміну поля Н_кімн згідно поточного вмісту таблиці в записах, що відносяться до Петрова, Сидорова і Єгорова.

Аномалії видалення полягають в тому, що при видаленні якого-небудь даного з таблиці може пропасти і інша інформація, яка не пов'язана напряму з тим, що видаляється.

В тій же таблиці С_Т_Н видалення запису про співробітника Іванова (наприклад, внаслідок звільнення або відходу на заслужений відпочинок) приводить до зникнення інформації про номер телефону, встановленого в 109-й кімнаті.

Аномалії додавання виникають у випадках, коли інформацію в таблицю не можна помістити до тих пір, поки вона неповна, або вставка нового запису вимагає додаткового перегляду таблиці.

Прикладом може служити операція додавання нового співробітника все в ту ж таблицю С_Т_Н. Очевидно, буде протиприродним зберігання

відомостей в цій таблиці тільки про кімнату і номер телефону в ній, поки ніхто із співробітників не розташований в ній. Більш того, якщо в таблиці С_Т_Н поле Співробітник є ключовим, то зберігання в ній неповних записів з відсутнім прізвищем службовця просто неприпустимо через невизначеність значення ключового поля.

Другим прикладом виникнення аномалії додавання може бути ситуація включення в таблицю нового співробітника. При додаванні таких записів для виключення суперечностей бажано перевірити номер телефону і відповідний номер кімнати хоча б з одним із співробітників, що сидять з новим співробітником в тій же кімнаті. Якщо ж виявиться, що у декількох співробітників, що сидять в одній кімнаті, є різні телефони, то взагалі не ясно, що робити (чи то в кімнаті декілька телефонів, чи то якийсь з номерів помилковий).

Формування початкового відношення

Проектування БД починається з визначення всіх об'єктів, відомості про які будуть включені в базу, і визначення їх атрибутів. Потім атрибути зводяться в одну таблицю - початкове відношення.

Приклад. Формування початкового відношення.

Припустимо, що для учбової частини факультету створюється БД про викладачів. На першому етапі проектування БД в результаті спілкування із замовником (завідуючим учбовою частиною) повинні бути визначені відомості, тобто те, як вона повинна використовуватися і яку інформацію замовник хоче одержувати в процесі її експлуатації. В результаті встановлюються атрибути, які повинні міститися у відношеннях БД, і зв'язки між ними. Перерахуємо імена виділених атрибутів і їх короткі характеристики:

ПІБ - прізвище і ініціали викладача. Виключаємо можливість збігу прізвища і ініціалів у викладачів.

Посада - посада, займана викладачем.

Оклад - оклад викладача.

Стаж - викладацький стаж.

Н_Стаж - надбавка за стаж.

Каф - номер кафедри, на якій числиться викладач.

Предм - назва предмету (дисципліни), що читається викладачем.

Група - номер групи, в якій викладач проводить заняття.

ВидЗан - вид занять, що проводяться викладачем в учбовій групі.

Одна з вимог до відношень полягає в тому, щоб всі атрибути відношення мали атомарні (прості) значення. В початковому

відношенні кожний атрибут кортежу також повинен бути простим. Приклад початкового відношення ВИКЛАДАЧ наведений на мал. 11.1 нижче.

ВИКЛАДАЧ

{ПІБ+Предм+Група} – складний ключ

ПІБ	Посада	Оклад	Стаж	Н_Стаж	Каф	Предм	Група	ВидЗан
Іванов І.М.	викл.	500	5	100	25	СКБД	256	Практ
Іванов І.М.	викл.	500	5	100	25	СІ	123	Практ
Петров М.І.	ст.викл	800	7	100	25	СКБД	256	Лекція
Петров М.І.	ст.викл	800	7	100	25	Паскаль	256	Практ
Сидоров Н.Г	викл.	500	10	150	25	СІ	123	Лекція
Сидоров Н.Г.	викл.	500	10	150	25	Паскаль	256	Лекція
Єгоров В. В.	викл.	500	5	100	24	ПЕОМ	244	Лекція

Рисунок 11.1 – Початкове відношення ВИКЛАДАЧ

Вказане відношення має наступну схему ВИКЛАДАЧ (ПІБ, Посада, Оклад, Стаж, Н_Стаж, Каф, Предм, Група, ВидЗан).

Початкове відношення ВИКЛАДАЧ містить надмірне дублювання даних, яке і є причиною аномалій редагування. Розрізняють надмірність явну і неявну.

Явна надмірність полягає в тому, що у відношенні ВИКЛАДАЧ рядка з даними про викладачів, проведені заняття в декількох групах, повторюються відповідне число раз. Наприклад, всі дані Іванова повторюються двічі. Тому, якщо Іванов І.М. стане старшим викладачем, то цей факт повинен бути відображений в обох рядках. В іншому випадку матиме місце суперечність в даних, що є прикладом аномалії редагування, обумовленою явною надмірністю даних у відношенні.

Неявна надмірність у відношенні ВИКЛАДАЧ виявляється в однакових окладах у всіх викладачів і в однакових добавках до окладу за однаковий стаж. Тому, якщо при зміні окладів з 500 на 510 це значення змінять у всіх викладачів, окрім, наприклад, Сидорова, то база

стане суперечливою. Це приклад аномалії редагування для варіанту з неявною надмірністю.

Засобом виключення надмірності у відношеннях і, як наслідок, аномалій є нормалізація відношень, розглянемо її більш детально.

Метод нормальних форм

Проектування БД є одним з етапів життєвого циклу інформаційної системи. Основною задачею, вирішуваною в процесі проектування БД, є задача нормалізації її відношень. Метод нормальних форм є класичним методом проектування реляційних БД. Цей метод заснований на фундаментальному в теорії реляційних баз даних понятті залежності між атрибутами відношень.

Залежність між атрибутами

Розглянемо основні види залежності між атрибутами відношень: функціональні, транзитивні і багатозначні. Поняття функціональної залежності є базовим, оскільки на його основі формулюються визначення решти видів залежностей.

Атрибут B **функціонально залежить** від атрибута A, якщо кожному значенню A відповідає в точності одне значення B. Математично функціональна залежність B від A позначається записом A->B. Це означає, що у всіх кортежах з однаковим значенням атрибута A атрибут B матиме також одне і те ж значення. Відзначимо, що A і B можуть бути складовими - складатися з двох і більш атрибутів.

У відношенні можна виділити функціональну залежність між атрибутами ПІБ—>Каф, ПІБ—>Посада, Посада-оклад і інші. Наявність функціональної залежності у відношенні визначається природою речей, інформація про які представлена кортежами відношення. У відношенні ключ є складним і складається з атрибутів ПІБ, Предмет, Група.

Функціональна взаємозалежність. Якщо існує функціональна залежність вигляду A->B і B->A, то між A і B є взаємно однозначна відповідність, або функціональна взаємозалежність. Наявність функціональної взаємозалежності між атрибутами A і B позначимо як A<->B або B<->A.

Приклад. Припустимо є деяке відношення, що включає два атрибути, функціонально залежні один від одного. Це серія і номер паспорта (N) і прізвище, ім'я і по батькові власника (ПІБ). Наявність функціональної залежності поля ПІБ від N означає не тільки той факт, що значення поля N однозначно визначає значення поля ПІБ, але і те, що одному і тому ж значенню поля N відповідає тільки єдине значення

поля ПІБ. Зрозуміло, що в даному випадку діє і зворотня ФЗ: кожному значенню поля ПІБ відповідає тільки одне значення поля N. В даному прикладі передбачається, що ситуація наявності повного збігу прізвищ, імен і по батькові двох людей виключена.

Якщо відношення знаходиться в 1НФ, то всі неключові атрибути функціонально залежать від ключа з різним ступенем залежності.

Частковою залежністю (частковою функціональною залежністю) називається залежність неключового атрибута від частини складного ключа. В даному відношенні атрибут Посада знаходиться у функціональній залежності від атрибута ПІБ, що є частиною ключа. Тим самим атрибут Посада знаходиться в частковій залежності від ключа відношення.

Альтернативним варіантом є **повна функціональна залежність** неключового атрибута від всього складного ключа. В нашому прикладі атрибут ВидЗан знаходиться в повній функціональній залежності від складного ключа.

Атрибут B залежить від атрибута A **транзитивно** (існує транзитивна залежність), якщо для атрибутів A, B, C виконуються умови A—>B і B—>C, але зворотня залежність відсутня. У відношенні транзитивною залежністю зв'язані атрибути:

ПІБ ->Посада-Оклад

Між атрибутами може мати місце багатозначна залежність.

Відносно R атрибут B **багатозначно залежить** від атрибута A, якщо кожному значенню A відповідає безліч значень B, не пов'язаних з іншими атрибутами з R.

Багатозначна залежність може бути «один до багатьох» (1:M), «багато до одного» (M: 1) або «багато до багатьох» (M:M), що позначаються відповідно: A=>B, A<=B і A<=>B.

Наприклад, припустимо викладач веде декілька предметів, а кожний предмет може вестися декількома викладачами, тоді має місце залежність ПІБ - Предмет. Так, видно, що викладач Іванов І.М. веде заняття по двох предметах, а дисципліна СКБД - читається двома викладачами: Івановим І.М. і Петровим М.І.

Зауваження. В загальному випадку між двома атрибутами одного відношення може існувати залежність: 1:1,1:M, M:1 і M:M. Оскільки залежність між атрибутами є причиною аномалій, прагнуть розчленувати відношення із залежністю атрибутів на декілька відношень. В результаті утворюється сукупність зв'язаних відношень

(таблиць) із зв'язками вигляду 1:1,1:M, M:1 і M:M. Зв'язки між таблицями відображають залежність між атрибутами різних відношень.

Взаємно незалежні атрибути. Два або більш атрибута називаються взаємно незалежними, якщо жоден з цих атрибутів не є функціонально залежним від інших атрибутів.

У разі двох атрибутів відсутність залежності атрибута A від атрибута B можна позначити так: A¬>B. Випадок, коли A¬>B і B¬> A, можна позначити A¬ =B.

Виявлення залежності між атрибутами

Виявлення залежності між атрибутами необхідне для виконання проектування БД методом нормальних форм.

Основний спосіб визначення наявності функціональної залежності — уважний аналіз семантики атрибутів. Для кожного відношення існує, але не завжди, певна множина функціональної залежності між атрибутами. Причому якщо в деякому відношенні існує одна або декілька функціональних залежностей, можна вивести іншу функціональну залежність, існуючу в цьому відношенні.

Виявимо залежність між атрибутами відношення ВИКЛАДАЧ. При цьому врахуємо наступну умову, яка виконується в даному відношенні: один викладач в одній групі може проводити один вид занять (лекції або практичні заняття).

В результаті аналізу відношення одержуємо залежність між атрибутами.

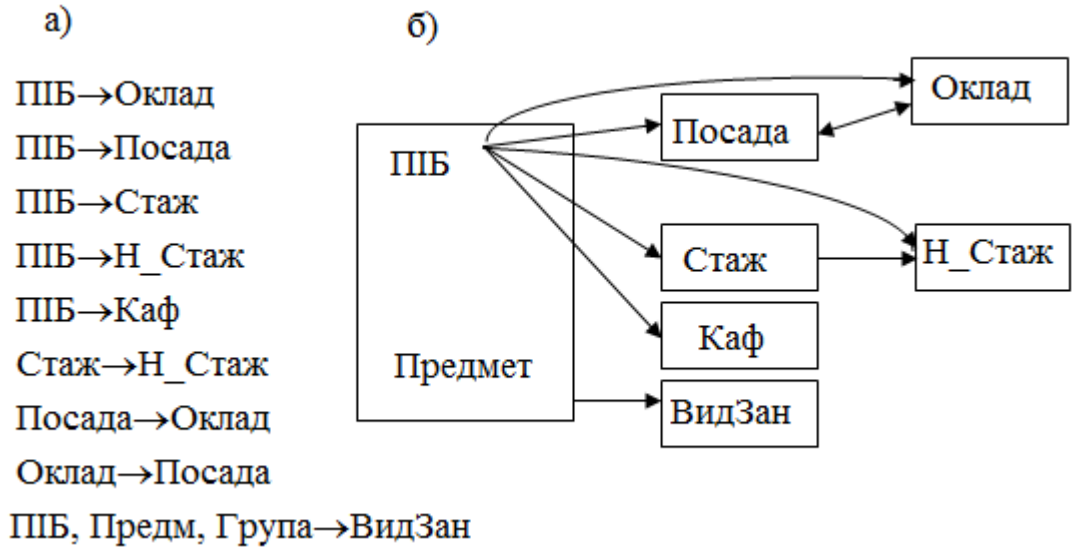


Рисунок 11.2 – Залежність між атрибутами

До виділення цих функціональних залежностей для даного прикладу приводять наступні міркування.

Прізвище, ім'я і по батькові у викладачів факультету унікальні. Кожному викладачу однозначно відповідає його стаж, тобто має місце функціональна залежність ПІБ→Стаж. Зворотне твердження невірно, оскільки однаковий стаж може бути у різних викладачів.

Кожний викладач має певну добавку за стаж, тобто має місце функціональна залежність ПІБ→Н_Стаж, але зворотна функціональна залежність відсутня, оскільки одну і ту ж надбавку можуть мати декілька викладачів.

Кожний викладач має певну посаду (викл., ст.викл., доцент, професор), але одну і ту ж посаду можуть мати декілька викладачів, тобто має місце функціональна залежність ПІБ→Посада, а зворотна функціональна залежність відсутня.

Кожний викладач є співробітником однієї і лише однієї кафедри. Тому функціональна залежність ПІБ→Каф має місце. З другого боку, на кожній кафедрі багато викладачів, тому зворотної функціональної залежності немає.

Кожному викладачу відповідає конкретний оклад, який однаковий для всіх педагогів з однаковими посадами, що враховується залежністю ПІБ→Оклад і Посада→Оклад. Немає однакових окладів для різних посад, тому має місце функціональна залежність Оклад→Посада.

Один і той же викладач в одній групі по різних предметах може проводити різні види занять. Визначення виду занять, які проводить викладач, неможливе без вказівки предмету і групи, тому має місце функціональна залежність ПІБ, предм, Група→ВидЗан. Дійсно, Петров М.І. в групі читає лекції і проводить практичні заняття. Але лекції він читає по СКБД, а практику проводить по Паскалю.

Нами не була виділена залежність між атрибутами ПІБ, предм і Група, оскільки вони утворюють складний ключ і не враховуються в процесі нормалізації початкового відношення.

Після того, як виділена вся функціональна залежність, слід перевірити їх узгодженість з даними початкового відношення ВИКЛАДАЧ.

Наприклад, Посада='викл' і Оклад=500 завжди відповідають один одному у всіх кортежах, тобто підтверджується функціональна залежність Посада-оклад. Так само слід варифікувати і решту функціональних залежностей, не забуваючи про обмеженість відносно даних.

Контрольні запитання:

1. Назвіть підходи до проектування структур даних.
2. В чому полягає надмірне і ненадмірне дублювання даних?
3. Назвіть і охарактеризуйте основні види аномалій.
4. Як формується початкове відношення при проектуванні БД?
5. Наведіть приклади явної і неявної надмірності.
6. Назвіть основні види залежності між атрибутами відношень.
7. Наведіть приклади функціональної і часткової функціональної залежності.
8. Наведіть приклади відношень із залежними атрибутами.

Тестові завдання:

1. Яке проектування полягає у визначенні числа і структури таблиць, формуванні запитів до БД, визначенні типів звітних документів, розробці алгоритмів обробки інформації, створенні форм для введення і редагування даних в базі і рішенні ряду інших задач?

а) Фізичне проектування
б) Логічне проектування
в) Логічно-фізичне проектування
г) Правильної відповіді немає

2. Аномалії, які виникають у випадках, коли інформацію в таблицю не можна помістити до тих пір, поки вона неповна, або вставка нового запису вимагає додаткового перегляду таблиці називають аномаліями:

а) Модифікації
б) Видалення
в) Додавання
г) Правильної відповіді немає

3. Залежність неключового атрибута від частини складного ключа називають:

а) Повною функціональною залежністю
б) Частковою залежністю
в) Транзитивною залежністю
г) Правильної відповіді немає

4. В загальному випадку між двома атрибутами одного відношення може існувати залежність:

а) 1:1,1:M, M:1 і M:M

б) 1:1, 1:2, 2:1, 2:2

в) 1:1, 1:M, M:1, 2:2

г) Правильної відповіді немає

5. Ненадмірне дублювання даних ще називають:

а) Складним

б) Повним

в) Простим

г) Правильної відповіді немає

Розділ 12

Нормальні форми

- **Перша нормальна форма**
- **Друга нормальна форма**
- **Третя нормальна форма**
- **Четверта нормальна форма**
- **П'ята нормальна форма**

Процес проектування БД з використанням методу нормальних форм є ітераційним і полягає в послідовному переведені відношень з першої нормальної форми до бінормальних форм більш високого порядку за певними правилами. Кожна наступна нормальна форма обмежує певний тип функціональної залежності, усуває відповідні аномалії при виконанні операцій над відношеннями БД і зберігає властивості попередніх нормальних форм.

Виділяють наступну послідовність нормальних форм:

• перша нормальна форма (1НФ);

• друга нормальна форма (2НФ);

• третя нормальна форма (ЗНФ);

• посилена третя нормальна форма, або нормальна форма Бойса-Кодда (БКНФ);

• четверта нормальна форма(4НФ);

• п'ята нормальна форма (5НФ).

Перша нормальна форма. Відношення знаходиться в 1НФ, якщо всі його атрибути є простими (мають єдине значення). Початкове відношення будується так, щоб воно було в 1НФ. Переведення відношення в наступну нормальну форму здійснюється методом «декомпозиції без втрат». Така декомпозиція повинна забезпечити те, що запити (вибірка даних по умові) до початкового відношення і до відношень, одержуваних в результаті декомпозиції, дадуть однаковий результат.

Основною операцією методу є операція проекції. Пояснимо її на прикладі. Припустимо, що відношенню R(A, B, C, D, E...) усунення функціональної залежності C—>D дозволить перевести його в наступну нормальну форму. Для вирішення цієї задачі виконаємо декомпозицію відношення R на два нові відношення R1(A,B,C,E...) і R2(C,D). Відношення R2 є проекцією відношення R на атрибути C і D.

Початкове відношення ВИКЛАДАЧ, що використовується для ілюстрації методу, має складний ключ **ПІБ, Предм, Група** і знаходиться в 1НФ, оскільки всі його атрибути прості. В цьому відношенні можна виділити часткову залежність атрибутів Стаж, Н_Стаж, Каф, Посада, Оклад від ключа - вказані атрибути знаходяться у функціональній залежності від атрибута ПІБ, що є частиною складного ключа.

Ця часткова залежність від ключа приводить до наступного:

1. У відношенні присутнє явне і неявне надмірне дублювання даних, наприклад:

• повторення відомостей про стаж, посаду і оклад викладачів, що проводять заняття в декількох групах або по різних предметах;

• повторення відомостей про оклади для однієї і тієї ж посади або про надбавкиза однаковий стаж.

2. Наслідком надмірного дублювання даних є проблема їх редагування. Наприклад, зміна посади у викладача Іванова І.М. тягне за собою проглядання всіх кортежів відношення і внесення змін в ті з них, які мають відомості про даного викладача.

Частина надмірності усувається при переведені відношення в 2НФ.

Друга нормальна форма. Відношення знаходиться в 2НФ, якщо воно знаходиться в 1НФ і кожний неключовий атрибут функціонально повно залежить від первинного ключа (складного). Для усунення часткової залежності і переведення відношення в 2НФ необхідно, використовуючи операцію проекції, розкласти його на декілька відношень таким чином:

• побудувати проекцію без атрибутів, що знаходяться в частковій функціональнійзалежності від первинного ключа;

• побудувати проекції на частини складного первинного ключа і атрибути, залежні від цих частин.

В результаті отримаємо два відношення R1 і R2 в 2НФ.

а)

R1

ПІБ	Предм	Група	ВидЗан
Іванов І.М.	СКБД	256	Практ
Іванов І.М.	Сі	123	Практ
Петров М.І.	СКБД	256	Лекція
Петров М.І.	Паскаль	256	Практ
Сидоров Н.Г.	Сі	123	Лекція
Сидоров Н.Г.	Паскаль	256	Лекція
Єгоров В.В.	ПЕОМ	244	Лекція

R2

ПІБ	Посада	Оклад	Стаж	Н_Стаж	Каф
Іванов І.М.	Викл	500	5	100	25
Петров М.І.	Ст.викл	800	7	100	25
Сидоров Н.Г.	Викл	500	10	150	25
Єгоров В.В.	Викл	500	5	100	24

б)

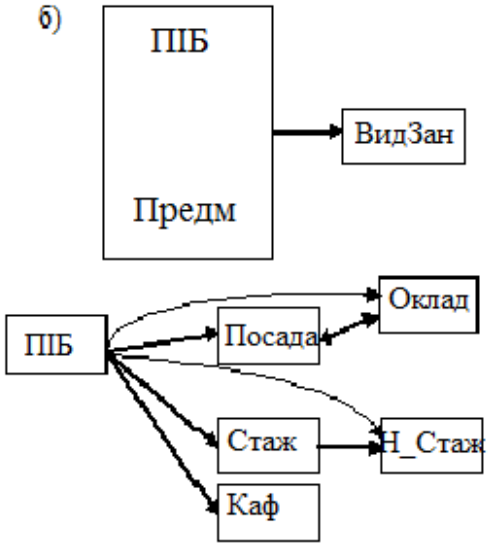


Рисунок 12.1 - Відношення БД в 2НФ

У відношенні R1 первинний ключ є складним і складається з атрибутів **ПІБ, Предм, Група.** Ключ відношення R1 отриманий з припущенням, що кожний викладач в одній групі по одному предмету може або читати лекції, або проводити практичні заняття. Відношення R2 має ключ **ПІБ.** Дослідження відношень R1 і R2 показує, що перехід до 2НФ дозволив виключити явну надмірність даних в таблиці R2 - повторення рядків з відомостями про викладачів. В R2 як і раніше має місце неявне дублювання даних. Для подальшого вдосконалення відношення необхідно перетворити його в ЗНФ.

Третя нормальна форма.

Визначення 1. Відношення знаходиться в ЗНФ, якщо воно знаходиться в 2НФ і кожний неключовий атрибут нетранзитивно залежить від первинного ключа. Існує і альтернативне визначення.

Визначення 2. Відношення знаходиться в ЗНФ в тому і лише в тому випадку, якщо всі неключові атрибути відношення взаємно незалежні і повністю залежать від первинного ключа.

Довести справедливість цього твердження нескладно. Дійсно, те, що неключові атрибути повністю залежать від первинного ключа, означає, що дане відношення знаходиться у формі 2НФ. Взаємна незалежність атрибутів (визначення приведено вище) означає відсутність всякої залежності між атрибутами відношення, у тому числі і транзитивної залежності між ними. Таким чином, друге визначення ЗНФ зводиться до першого визначення.

Якщо в відношенні R1 транзитивна залежність відсутня, то в відношенні R2 вони є:

ПІБ>Посада>Оклад, ПІБ>Оклад>Посада, ПІБ>Стаж>Н_Стаж

Транзитивна залежність також породжує надмірне дублювання інформації у відношенні. Усунемо їх. Для цього використовуючи операцію проекції на атрибути, що є причиною транзитивної залежності, перетворимо відношення R2, отримавши при цьому відношення R3, R4 і R5, кожне з яких знаходиться в ЗНФ. Помітимо, що відношення R2 можна перетворити по-іншому, а саме: у відношенні R3 замість атрибута Посада взяти атрибут Оклад. На практиці побудова ЗНФ схем відношень в більшості випадків є достатньою і приведенням до них, процес проектування реляційної БД закінчується. Розглянемо рисунок 12.2:

а)

R3

ПІБ	Посада	Стаж	Каф
Іванов І.М.	Викл	5	25
Петров М.І.	Ст.викл	7	25
Сидоров Н.Г.	Викл	10	25
Єгоров В.В.	Викл	5	24

R4

Посада	Оклад
Викл	500
Ст.викл	800

R5

Стаж	Н_Стаж
5	100
7	100
10	150

б)

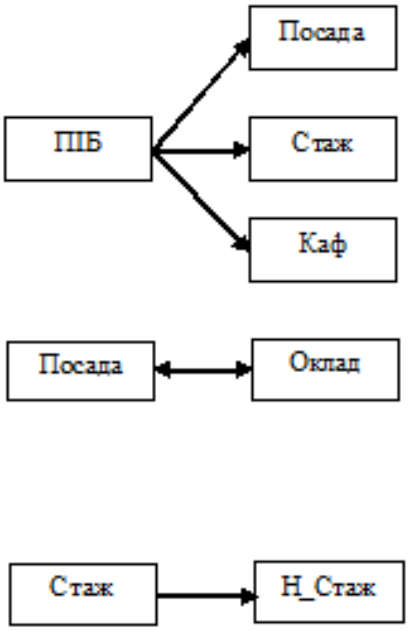


Рисунок 12.2 - Відношення БД в ЗНФ

Дійсно, приведення відношень до ЗНФ в нашому прикладі, привело до усунення надмірного дублювання. Якщо у відношенні є залежність атрибутів складного ключа від неключових атрибутів, то необхідно перейти до посиленої ЗНФ. **Посилена ЗНФ** або нормальна форма **Бойса-Кодда** (БКНФ). Відношення знаходиться в БКНФ, якщо воно знаходиться в ЗНФ і в ньому відсутня залежність ключів (атрибутів складного ключа) від неключових атрибутів.

У нас подібної залежності немає, тому процес проектування на цьому закінчується. Результатом проектування є БД, що складається з наступних таблиць: Rl, R3, R4, R5. В отриманій БД має місце необхідне дублювання даних, але відсутнє надмірне.

Четверта нормальна форма.

Розглянемо приклад нового відношення ПРОЕКТИ, схема якого виглядає таким чином: ПРОЕКТИ (Номер_проекта, Код_співробітника, Завдання_співробітника). Первинним ключем відношення є вся сукупність атрибутів: Номер_проекта, Код_співробітника і Завдання_співробітника. У відношенні містяться номери проектів, для кожного проекту - список кодів співробітників-виконавців, а також список завдань, передбачених кожним проектом. Співробітники можуть брати участь в декількох проектах, і різні проекти можуть містити однакові завдання. Передбачається, що кожний співробітник, що бере участь в деякому проекті, виконує всі завдання за цим проектом (припущення не завжди справедливо, але бажане для нашого прикладу).

При такій постановці питання єдиним можливим ключем відношення є складний атрибут Номер_проекта, Код_співробітника, Завдання_співробітника. Віні став первинним ключем відношення. Звідси витікає, що відношення ПРОЕКТИ, знаходиться у формі БКНФ. Нехай вхідна інформація в цьому відношенні виглядає таким чином:

ПРОЕКТИ

Номер_проекта	Код_співробітника	Завдання_співробітника
001	05	1
001	05	2
001	05	3
004	02	1
004	02	2
004	03	1
004	03	2
004	05	1
004	05	2
007	06	1

Головний недолік відношення ПРОЕКТИ полягає в тому, що при підключенні/вилученні від проекту деякого співробітника доводиться додавати/ виключати з відношення стільки кортежів, скільки завдань є в проекті. Внесення або виключення відносно одного факту про деякого

співробітника вимагає серії елементарних операцій через дублювання значень в кортежах.

Звідси виникають питання: навіщо зберігати в кортежах значення кодів співробітників, що повторюються? Чи потрібно перераховувати всі завдання за кожним проектом, та ще для кожного співробітника-виконавця цього проекту? Чи не можна інформацію про прив'язку завдань до проектів помістити в окрему таблицю і виключити повторення в основній таблиці?

Помітимо, що непряма ознака аномалії, як і раніше, - дублювання інформації в таблиці. Викажемо припущення, що причиною аномалії є наявність деякої залежності між атрибутами відношення (як побачимо далі - багатозначної залежності).

Дійсно, у відношенні ПРОЕКТИ існують наступні дві багатозначні залежності:

Номер_проекта=>Код_співробітника Номер_проекта=>Завдання_ співробітника

В довільному відношенні R(A, B, C) може одночасно існувати багатозначна залежність A=>B і A=>C. Цю обставину позначимо як A=>B |C.

Подальша нормалізація відношень, схожих з відношенням Проекти, ґрунтується на наступній теоремі.

Теорема Фейджіна (Fagin R). Відношення R(A, B, C) можна спроектувати без втрат у відношення R1(A, B) і R2(A, C) в тому і лише тому випадку, коли існує залежність A=>B |C.

Під проектуванням без втрат тут розуміється такий спосіб декомпозиції відношення, при якому початкове відношення повністю і без надмірності відновлюється шляхом **природного з'єднання** отриманих відношень.

Результатом **операції з'єднання** бінарних відношень R1(A, B) і R2(A, C) по атрибуту A є тетрарне відношення з атрибутами A, B і C, кортежі якого виходять шляхом зв’язування відношень R1 і R2 по типу 1:M на основі співпадання значень атрибута A.

Визначення четвертої нормальної форми. Відношення R знаходиться в четвертій нормальній формі (4НФ) в тому і лише у тому випадку, коли існує багатозначна залежність A=>B, а вся решта атрибутів R функціонально залежить від A.

Приведене вище відношення ПРОЕКТИ можна представити у вигляді двох відношень: ПРОЕКТИ-СПІВРОБІТНИКИ і ПРОЕКТИ-ЗАВДАННЯ. Ці відношення мають наступну структуру:

ПРОЕКТИ-СПІВРОБІТНИКИ (Номер_проекта, Код_співробітника). **Первинний ключ відношення**: Номер_проекта, Код_співробітника. ПРОЕКТИ-ЗАВДАННЯ (Номер_проекта, Завдання_ співробітника). Первинний **ключ відношення:** Номер_проекта, Завдання_ співробітника.

Обидва ці відношення знаходяться в 4НФ і вільні від помічених недоліків. Дублювання значень атрибутів кодів співробітників пропало. Якщо з'єднати ці відношення, то вийде відношення ПРОЕКТИ.

В загальному випадку не всяке відношення можна відновити до початкового. В нашому випадку відновлення можливо тому що кожний співробітник, що бере участь в деякому проекті, виконував всі завдання за цим проектом (саме це укладається в принцип 1:М з'єднання відношень). Самі ж співробітники брали участь в декількох проектах, і різні проекти могли містити однакові завдання.

П'ята нормальна форма.

У всіх розглянутих раніше нормальних формах нормалізація проводилась декомпозицією одного відношення у два, а іноді і більше відношень.

Припустимо, що один і той же співробітник може працювати в кількох відділах і в кожному відділі може брати участь в кількох проектах.

Розглянемо відношення

СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ (Спів_номер, Від_номер, Про_номер)

Первинним ключем цього відношення є повна сукупність його атрибутів.

У відношенні немає функціональних і багатозначних залежностей. Відношення знаходиться в 4НФ.

Але у ньому можуть існувати аномалії, які можна усунути шляхом декомпозиції у три відношення. Для відношення R із n атрибутів можна побудувати n проекцій по кожному з цих атрибутів. З’єднання отриманих відношень (всіх або деяких) являє собою нове відношення. Таке відношення може співпадати з первинним (тобто відновлюватись без втрат) або ні.

Визначення: Відношення R(X,Y,…,Z) задовільняє залежності сполучення *(X,Y,…,Z) в тому і тільки в тому випадку, коли R відновлюється без втрат шляхом сполучення своїх проекцій на X,Y,…,Z. Інакше кажучи відношення залежить від сполучення своїх проекцій.

Визначення. П’ята нормальна форма.

Відношення R знаходиться у п’ятій нормальній формі (нормальній формі проекції-сполучення PJ/NF) в тому і тільки і в тому випадку, коли будь-яка залежність сполучення в R створюється з проекцій за ключовими атрибутами, які в сукупності є можливим ключем відношення.

Розглянемо складні атрибути:

СВ=(Спів_номер, Від_номер)

СП=(Спів_номер, Про_номер)

ВП=(Від_номер, Про_номер)

Припустимо, що у відношенні

СПІВРОБІТНИКИ_ВІДДІЛИ_ПРОЕКТИ існує залежність сполучення: (СВ, СП, ВП).

Для того щоб уникнути проблем під час вставлення та вилучення кортежів потрібно шляхом декомпозиції вхідного відношення у три нових відношення:

СПІВРОБІТНИК_ВІДДІЛИ (Спів_номер, Від_номер)

СПІВРОБІТНИКИ_ ПРОЕКТИ (Спів_номер, Про_номер)

ВІДДІЛИ_ПРОЕКТИ (Від_номер, Про_номер)

П’яту нормальну форму можна отримати шляхом декомпозиції з 4НФ. На практиці 5НФ не використовується. Залежність сполучення є узагальненням як багатозначної так і функціональної залежності.

На практиці обмежуються структурою БД, відповідної ЗНФ або БКНФ. Тому процес нормалізації відношень методом нормальних форм припускає послідовне видалення з початкового відношення наступної між атрибутної залежності:

- часткової залежності неключових атрибутів від ключа (задоволення вимог 2НФ);
- транзитивної залежності неключових атрибутів від ключа (задоволення вимог 3НФ);
- залежність ключів (атрибутів складових ключів) від неключових атрибутів (задоволення вимог БКНФ).

Окрім методу нормальних форм Кодда, що використовується для проектування невеликих БД, застосовують і інші методи, наприклад, метод ER-діаграм (метод «Суттєвість-зв'язок). Цей метод використовується при проектуванні великих БД, на ньому заснований ряд засобів проектування БД.

Контрольні запитання:

1. Охарактеризуйте нормальні форми.
2. Дайте визначення першої нормальної форми.
3. Дайте визначення другої нормальної форми.
4. Дайте визначення третьої нормальної форми.
5. Дайте визначення посиленої третьої нормальної форми.
6. Поясніть на прикладі що використовуються в розділі таблиць вимоги 4НФ.
7. Поясніть на прикладі що використовуються в розділі таблиць вимоги 5НФ.

Тестові завдання:

1. Початкове відношення будується так, щоб воно було в 1НФ. Яким методом здійснюється переведення відношення в наступну нормальну форму?

а) Методом «декомпозиції без втрат»
б) Методом «декомпозиції з витратами»
в) Такого методу не існує
г) правильної відповіді немає

2. Яка основна операція методу декомпозиції?

а) Операція проекції
б) Операція перенесення
в) Операція додавання
г) правильної відповіді немає

3. Нормальна форма Бойса-Кодда – це:

а) Посилена 1НФ
б) Посилена 2НФ
в) Посилена 3НФ
г) правильної відповіді немає

4. Відношення R знаходиться в такій формі в тому і лише у тому випадку, коли існує багатозначна залежність A=>B, а вся решта атрибутів R функціонально залежить від A. Про яку нормальну форму йдеться мова?

а) 5НФ

б) 4НФ

в) 3НФ

г) БКНФ

5. Яку нормальну форму можна отримати шляхом декомпозиції з 4НФ?

а) 5НФ

б) 4НФ

в) 3НФ

г) 1НФ

Розділ 13

Рекомендації з розробки структур

- **Таблиці суттєвостей**
- **Організація зв’язку суттєвостей**
- **Забезпечення цілісності**

Як узагальнення матеріалу попередньогорозділу приведемо найважливіші рекомендації, зневага до яких може привести до аномалій при обробці даних в базах даних. Зупинимося на двох питаннях: виходячи з яких міркувань потрібно створювати відношення (таблиці) і яким чином слідує їх зв'язувати.

Якими повинні бути таблиці суттєвостей

Основне правило при створенні таблиць суттєвостей - це «кожній суттєвості - окрему таблицю».

Поля таблиць суттєвостей можуть бути двох видів: **ключові і неключові.** Введення ключів в таблиці практично у всіх реляційних СКБД дозволяє забезпечити унікальність значень в записах таблиці по ключу, прискорити обробку записів таблиці, а також виконати автоматичне сортування записів за значеннями в ключових полях.

Звичайно достатньо визначення **простого** ключа, рідше - вводять **складний** ключ. Таблицею з складним ключем може бути, наприклад, таблиця зберігання списку співробітників (прізвище, ім'я і по батькові), в якому зустрічаються однофамільці. В деяких СКБД користувачам пропонується визначити автоматично створюване ключове поле нумерації (в Access - це поле типу «лічильник»), яке спрощує рішення проблеми унікальності записів таблиці.

Іноді в таблицях є поля опису властивостей або характеристик об'єктів. Якщо в таблиці є значне число повторень по цих полях і ця інформація має істотний об'єм, то краще їх виділити в окрему таблицю (дотримуючись правила: «кожній суттєвості - окрему таблицю»). Тим більше, слід утворити додаткову таблицю, якщо властивості взаємозв'язані.

В більш загальному вигляді останні рекомендації можна сформулювати так: інформацію про суттєвості слід представити так, щоб неключові поля в таблицях були взаємно незалежними і повністю залежали від ключа (див. визначення третьої нормальної форми).

В процесі обробки таблиць суттєвостей треба мати на увазі наступне. Нову суттєвість легко додати і змінити, але при видаленні -

слід знищити всі посилання на неї з таблиць зв'язків, інакше таблиці зв'язків будуть некоректними. Багато сучасних СКБД блокують некоректні дії в подібних операціях.

Організація зв'язку суттєвостей

Записи **таблиці зв'язків** призначені для відображення зв'язків між суттєвостями, інформація про які знаходиться у відповідних таблицях суттєвостей.

Звичайно одна таблиця зв'язків описує взаємозв'язок двох суттєвостей. Оскільки таблиці суттєвостей в найпростішому випадку мають по одному ключовому полю, то таблиця зв'язків двох таблиць для забезпечення унікальності записів про зв'язки повинна мати два ключі. Можна створити таблицю зв'язків, як і таблицю об'єктів, і без ключів, але тоді функції контролю за унікальністю записів лягають на користувача.

Складніші зв'язки (не бінарні) слід зводити до бінарних. Для опису взаємозв'язків N об'єктів потрібно N-1 таблиць зв'язків. Транзитивних зв'язків не повинно бути. Надлишок зв'язків приводить до суперечностей (див. приклад відношень СПІВРОБІТНИКИ-ВІДДІЛИ, СПІВРОБІТНИКИ-ПРОЕКТИ і ВІДДІЛИ-ПРОЕКТИ попереднього підрозділу).

Не слід включати в таблиці зв'язків характеристики суттєвостей, інакше неминучі аномалії. Їх краще берегти в окремих таблицях суттєвостей.

За допомогою таблиць зв'язків можна описувати і дещо специфічний вид зв'язку - лінійний зв'язок, або слабкий зв'язок. Прикладом лінійного зв'язку можна вважати відношення приналежності суттєвостей деякої іншої суттєвості більш високого порядку (системи, що складаються з вузлів; ліки, що складаються з компонентів; сплави металів і т. д.). В цьому випадку для опису зв'язків достатньо однієї таблиці зв'язків.

При роботі з таблицями зв'язків слід мати на увазі, що будь-який запис з таблиці зв'язків легко може бути видалений, оскільки суттєвості якийсь час можуть обійтися і без зв'язків. При додаванні або зміні вмісту записів таблиці треба контролювати правильність посилань на існуючі об'єкти, оскільки зв'язок без об'єктів існувати не може. Більшість сучасних СКБД контролює правильність посилань на об'єкти.

Забезпечення цілісності

Під **цілісністю** розуміють властивість бази даних, що означає, що вона містить повну, несуперечливу і адекватно відображаючу предметну область інформацію.

Розрізняють **фізичну** і **логічну** цілісність. Фізична цілісність означає наявність фізичного доступу до даних і те, що дані не втрачені. Логічна цілісність означає відсутність логічних помилок в базі даних, до яких відносяться порушення структури БД або її об'єктів, видалення або зміна встановлених зв'язків між об'єктами і т.д. Надалі моваіде про логічну цілісність. Підтримка цілісності БД включає перевірку (контроль) цілісності і її відновлення у разі виявлення суперечностей в базі. Цілісний стан БД задається за допомогою **обмежень цілісності** у вигляді умов, яким повинні задовільняти збережені в базі дані.

Серед обмежень цілісності можна виділити два основні типи обмежень: **обмеження значень** атрибутів відношень і **структурні обмеження** на кортежі відношень.

Прикладом **обмежень значень** атрибутів відношень є вимога неприпустимості порожніх значень або тих що повторюються в атрибутах, а також контроль належності значень атрибутів заданому діапазону. Так, в записах відношень про кадри значення атрибута Дата_народження не можуть перевищувати значень атрибута Дата_прийому.

Найгнучкішим засобом реалізації контролю значень атрибутів є **збережені процедури і тригери,** щоє в деяких СКБД.

Структурні обмеження визначають вимоги **цілісності** суттєвостей і **цілісності посилань.** Кожному екземпляру суттєвості, представленому у відношенні, відповідає тільки один його кортеж. Вимога **цілісності** суттєвостей полягає в тому, що будь-який кортеж відношення повинен бути відмінним від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення повинне володіти первинним ключем.

Формування вимоги цілісності посилань тісно пов'язано з поняттям **зовнішнього ключа.** Нагадаємо, що зовнішні ключі служать для зв'язку відношень (таблиць БД) між собою. При цьому атрибут одного відношення (батьківського) називається **зовнішнім ключем даного** відношення, якщо він є первинним **ключеміншого відношення (дочірнього)**. Говорять, що відношення, в якому визначений зовнішній

ключ, посилається на відношення, в якому цей же атрибут є первинним ключем.

Вимога цілісності посилань полягає в тому, що для кожного значення зовнішнього ключа батьківської таблиці повинен знайтися рядок в дочірній таблиці з таким же значенням первинного ключа. Наприклад, якщо в відношенніR1 містяться відомості про співробітників кафедри, а атрибут цього відношення Посада є первинним ключем відношення R2, то в цьому відношенні для кожної посади з R1 повинен бути рядок з відповідним їй окладом.

В багатьох сучасних СКБД є засоби забезпечення контролю цілісності БД.

На рисунку 13.1 зображено зв'язок відношень за допомогою зовнішнього ключа.

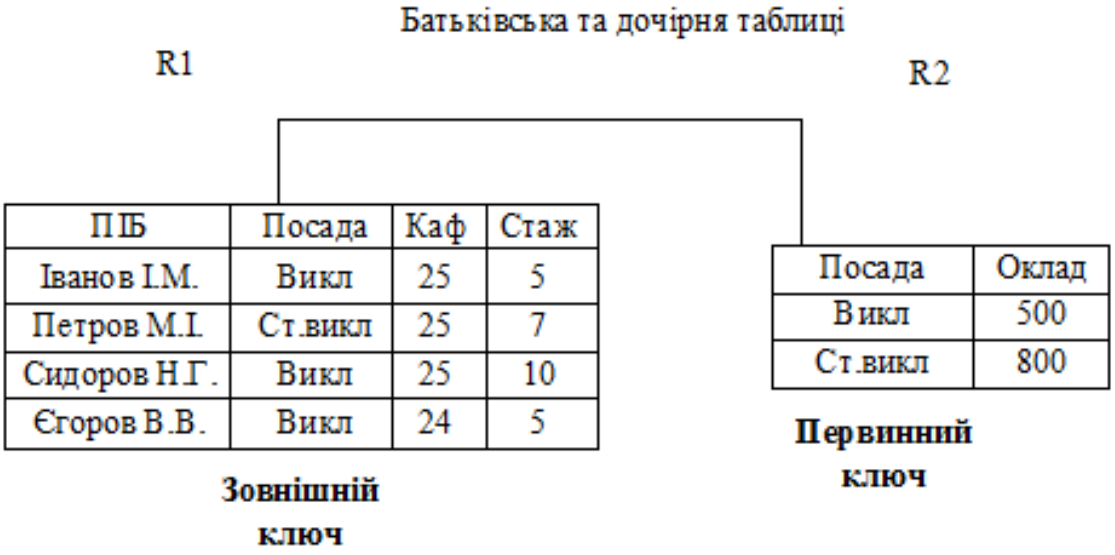


Рисунок 13.1 - Зв'язок відношень за допомогою зовнішнього ключа

Контрольні запитання:

1. Сформулюйте основне правило створення таблиць суттєвостей.
2. Назвіть рекомендації по організації зв'язку суттєвостей.
3. Дайте визначення фізичної і логічної цілісності БД.
4. Наведіть приклади обмежень значень і структурних обмежень.
5. Поясніть поняття зовнішнього і первинного ключів таблиць.
6. Розробіть схему БД, що дозволяє переглядати і редагувати інформацію про автомобільний парк організації. БД повинна містити відомості про водіїв машин (категорія транспортних засобів, стаж водія, закріплені автомобілі і т. д.), а також автомобілях автопарку (марка, рік випуску, технічний стан і т. д.).

Запропонуйте і обґрунтуйте вибір структур таблиць, їх взаємозв'язок і вкажіть вид нормальних форм таблиць.

Тестові завдання:

1. Скількох видів можуть бути поля таблиць суттєвостей?

а) Двох

б) Трьох

в) П’ятьох

г) Десятьох

д) Правильної відповіді немає

2. Які є поля таблиць суттєвостей?

а) Тільки ключові

б) Тільки неключові

в) Ключові та не ключові

г) Дескрипторні

д) Правильної відповіді немає

3. Цілісність розподіляють на:

а) Фізичну і логічну

б) Фізичну і бінарну

в) Логічну і бінарну

г) Віртуальну і невіртуальну

д)Правильної відповіді немає

4. Скільки ключів повинна мати таблиця зв'язків двох таблиць для забезпечення унікальності записів про зв'язки?

а) 1

б) 2

в) 3

г) 4

д) Правильної відповіді немає

5. З яким поняттям тісно пов’язано формування вимоги цілісності посилань?

а) Поняттям зовнішнього ключа

б) Поняттям первинного ключа

в) Поняттям жорсткого ключа
г) Поняттям дескриптора
д)Правильної відповіді немає

6. Що означає поняття цілісності БД?
а) Властивість бази даних, яка визначає, що в ній знаходиться повна, адекватно відображаюча ПО інформація без протиріч
б) Інструкція, яка описує всі кроки для виконання частини роботи з базою даних
в)Логічна структура бази даних
г) Захист бази даних від некоректного користування та руйнування
д) Немає правильної відповіді.

Розділ 14

Семантичне моделювання даних. ER – діаграми

- **Обмеженість реляційних моделей**
- **Семантичні моделі даних**
- **Побудова семантичної моделі бази даних**
- **Визначення призначення та кола завдань інформаційної системи**
- **Відокремлення зовнішніх сутностей**
- **Визначення функціональних процесів предметної області**
- **Побудова діаграм потоків**

Обмеженість реляційних моделей

Широке розповсюдження реляційних СКБД і їх використання у різноманітних додатках свідчать, що реляційна модель даних є достатньою для моделювання ПО. Однак проектування реляційної БД у термінах відношень на базі розглянутого механізму нормалізації часто являє собою дуже складний і незручний для проектувальника процес.

При цьому виявляється обмеженість реляційної моделі даних у таких аспектах:

- Модель не надає достатніх засобів для подання сеансу даних. Семантика реальної ПО має відображатися незалежним від моделі засобом. Зокрема, це стосується проблеми подання обмежень цілісності;
- Для багатьох додатків важко моделювати ПО на підставі плоских таблиць. На початковій стадії проектування проектувальнику важко описати ПО у вигляді однієї (можливо, навіть ненормалізованої) таблиці;
- Незважаючи на те, що весь процес проектування відбувається з урахуванням залежностей, реляційна модель не надає будь-яких засобів для подання цих залежностей.
- Незважаючи на те, що процес проектування починається з виділення деяких істотних для додатка об’єкта ПО («сутностей») і виявлення зв’язків між цими сутностями, реляційна модель даних не пропонує відповідного апарату для розподілу сутностей і зв’язків.

Сематичні моделі даних

Потреби проектувальників БД у більш зручних і потужних засобах моделювання ПО зумовили виникнення семантичних моделей даних.

При тому, що будь-яка розвинена семантична модель даних, так само як і реляційна модель, містить структурну, маніпуляційну і цілісну частини, головним призначенням семантичних моделей є забезпечення можливості відображення семантики даних.

Розглянемо можливі застосування семантичних моделей. Найчастіше на практиці семантичне моделювання використовують на першій стадії проектування БД. При цьому в термінах семантичної моделі виробляється концептуальна схема БД, яка після цього перетворюється на реляційну (або будь-яку іншу) схему. Цей процес виконується за методиками, в яких достатньо чітко обумовлені всі етапи перетворення.

Рідше реалізується автоматизована компіляція концептуальної схеми в реляційну. Відомо два підходи: підхід, що базується на явному поданні концептуальної схеми як вхідної інформації для компіляції, і підхід, орієнтований на побудову інтегрованих систем проектування з автоматизованим створенням концептуальної схеми на основі інтерв'ю з експертами ПО.

В обох випадках у результаті виробляється реляційна схема БД у третій нормальній формі.

Нарешті третя можливість, яка ще не вийшла (або тільки виходить) за межі дослідних і експериментальних проектів, - це безпосередня робота з БД в семантичній моделі, тобто СКБД, що базується на семантичних моделях даних. При цьому знову розглядають два варіанти: забезпечення інтерфейсу користувача на основі семантичної моделі даних з автоматичним відображенням конструкцій у реляційну модель даних (це завдання приблизно такого самого рівня складності, як і автоматична компіляція концептуальної схеми БД у реляційну схему) і пряма реалізація СКБД, що ґрунтується на будь-якій семантичній моделі даних. Найбільш близькими до другого підходу є сучасні об'єктно-орієнтовані СКБД, моделі даних яких за своїми параметрами подібні до семантичних моделей (хоча в деяких випадках вони є більш потужними, а в деяких – слабкішими).

Побудова семантичної моделі бази даних

За основу моделювання за методологією Gane/Sarson узято побудову ІС. Відповідно до методології модель системи визначають як ієрархію діаграм потоків даних (ДПД, або DFD – Date Flou Diagrames), які описують асинхронний процес перетворення інформації від її

введення у систему до видачі користувачу. Розглядаються перетворення, які є необхідними для виконання функцій системи.

Діаграми верхніх рівнів ієрархії (контекстні діаграми) визначають основні процеси із зовнішніми входами і виходами. Вони деталізуються за допомогою діаграм нижнього рівня. Така декомпозиція за допомогою діаграм нижнього рівня. Така декомпозиція продовжується, створюючи багаторівневу ієрархію діаграм, доти, доки деталізувати їх далі не потрібно.

Джерела інформації (зовнішні сутності) породжують інформаційні потоки (потоки даних), що переносять інформацію до процесів. Останні, у свою чергу, перетворюють інформацію і породжують нові потоки, що переносять інформацію до інших процесів, накопичувачів даних чи зовнішніх сутностей – споживачів інформації. Отже, основні компоненти діаграм потоків даних такі:

- Зовнішні сутності;
- Процеси;
- Накопичувачі даних;
- Потоки даних.

Побудову семантичної моделі БД виконують поетапно:

- Визначення призначення та кола завдань ІС;
- Відокремлення зовнішніх сутностей;
- Визначення процесів ПО;
- Побудова діаграм потоків даних;
- Проектування структури БД (концептуальної схеми).

Визначення призначення та кола завдань інформаційної системи

Для визначення призначення проводять обстеження реального об’єкта господарювання, для керування яким створюється ІС, в ході якого визначають основні проблеми, які має вирішити система. Залежно від ПО існує такий розподіл ІС:

- Інформаційні системи організаційного типу, які проектують з метою вирішення завдань керування процесами організації виробничих процесів;
- Навчальні інформаційні системи, призначенням яких є організація навчальних процесів;
- Інформаційні системи керування технологічними процесами на виробництві, тобто керування безпосередньо процесами виготовлення різноманітної продукції;

- Інформаційні системи, метою функціонування яких є створення технічної документації;
- Інформаційні системи, завданням яких є організація обміну інформацією між її користувачами, наприклад, різноманітні системи класу Intranet.

Визначення призначення має складатися з одного речення, в якому стисло, але повно сформульовано призначення системи. Наприклад, «Система обслуговує керування орендою нерухомості». Після формування визначення, його необхідно погодити з користувачем системи.

Для визначення завдань аналізують документацію, яка регламентує функціонування реальної системи, а також співбесіди з майбутніми користувачами різного рівня. Останнім кроком попереднього вивчення є фіксація переліку запитань до системи. Після визначення головних завдань необхідно визначити функції, за допомогою яких система вирішуватиме ці завдання. Перелік функцій відповідає переліку завдань, не виходячи за межі призначення системи.

До визначення функцій формулюють такі вимоги:

- Виробничі, наприклад, періодичність виконання;
- Функціональні, наприклад, визначення вигляду звітів та екранних форм;
- Технічні, наприклад, обмеження на тип операційної системи, інструментальне середовище або не на використання протоколів мережевого сполучення.

Відокремлення зовнішніх сутностей

Зовнішня сутність являє собою матеріальний предмет або фізичну особу, яка є джерелом або приймачем інформації, наприклад, замовники, персонал, постачальники, клієнти, склад. Визначення деякого об’єкта або системи як зовнішньої сутності вказує на те, що вона перебуває за межами аналізованої ІС. У процесі аналізу деякі зовнішні сутності можуть бути перенесені всередину діаграми ІС, якщо це необхідно, або навпаки, частину процесів ІС можна винести за межі діаграми і подати як зовнішню сутність.

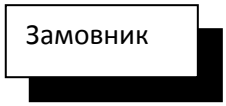


Рисунок 14.1 – Зовнішня сутність

Зовнішню сутність позначають квадратом, розташованим немов би «над» діаграмою з тінню, для того, щоб можна було виділити цей символ серед інших позначень.

Визначення функціональних процесів предметної області

Функціональний процес у ПО являє собою перетворення вхідних потоків даних на вихідні відповідно до визначеного алгоритму, для виконання деякої функції, притаманної даній області. Фізично процес може бути реалізований різними способами: у підрозділі організації (відділі), що виконує обробку вхідних документів і випуск звітів, програма, що виконує обробку даних, апаратно реалізований логічний пристрій і тощо.

Процес на діаграмі потоків даних показано на рисунку.

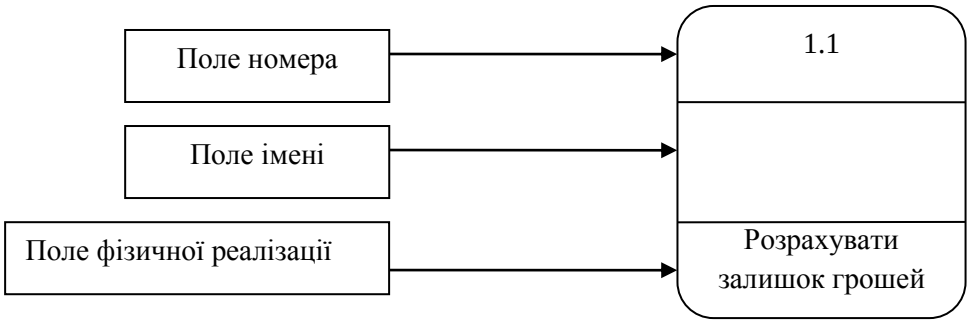


Рисунок 14.2 – Процес

Номер процесу використовують для його ідентифікації. У поле імені вводять найменування процесу у вигляді пропозиції з активним недвозначним дієсловом у неозначеній формі (обчислити, розрахувати, перевірити, визначити, створити, одержати), за яким йдуть іменники у знахідному відмінку, наприклад:

- «Ввести відомості про клієнтів»;
- «Видати інформацію про поточні витрати»;
- «Перевірити кредитоспроможність клієнта».

Використання таких дієслів, як «обробити», «модернізувати» або «відреагувати» означає, як правило, недостатньо глибоке розуміння цього процесу і вимагає подальшого аналізу.

Інформація в полі фізичної реалізації показує, який підрозділ організації, яка програма або апаратний пристрій виконують цей процес.

Побудова діаграм потоків

У процесі побудови діаграм потоків даних використовують поняття **накопичувача даних.**

Накопичувач даних являє собою абстрактний пристрій для збереження інформації, яку можна вбудь-який момент занести в нього і через якийсь час витягти, причому способи занесення і отримання можуть бути різними.

Накопичувач даних може бути реалізований фізично у вигляді мікрофіші, шухляди в картотеці, таблиці в оперативній пам'яті, файлу на магнітному носії тощо. Накопичувач даних на діаграмі потоків даних наведено на рисунку.

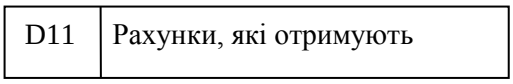


Рисунок 14.3 – Накопичувач даних

Накопичувач даних ідентифікують літерою D і довільним числом. Ім'я накопичувача вибирають з точки зору найбільшої інформативності для проектувальника.

Накопичувач даних у загальному випадку є прообразом майбутньої таблиці БД і опису даних, що зберігаються в ній, та має бути ув'язаний з інформаційною моделлю.

Потік даних визначає інформацію, передану через з'єднання від джерела до приймача. Реальний потік даних може бути, наприклад:

- Інформацією, переданою по кабелю між двома пристроями;
- Листами, що пересилаються поштою;
- Магнітними стрічками або дискетами, які переносять з одного комп'ютера на інший;
- Іншими носіями.

Потік даних на діаграмі зображують лінією, яка закінчується стрілкою та показує напрямок потоку. Кожний потік даних має ім'я, що відображає його зміст.

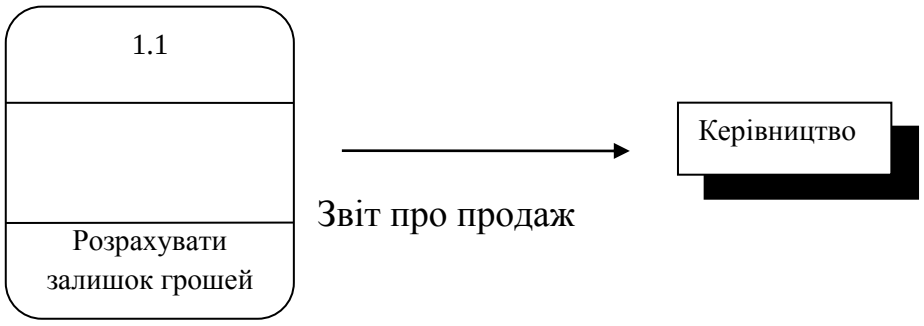


Рисунок 14.4 – Потік даних

Далі будують **ієрархію діаграм потоків даних.** Першим кроком під час побудови ієрархії ДПД є побудова контекстних діаграм. Звичайно,

під час проектування простих ІС будують єдину контекстну діаграму із зіркоподібною топологією, у центрі якої знаходиться так званий **головний процес,** з'єднаний із приймачами і джерелами інформації, за допомогою яких із системою взаємодіють користувачі й інші зовнішні системи.

Якщо для складної системи обмежитися єдиною контекстною діаграмою, то вона буде містити занадто велику кількість джерел і приймачів інформації, що важко розташувати на аркуші паперу нормального формату і, крім того, єдиний головний процес не розкриває структури розподіленої системи. Ознаками складності (за контекстом) можуть бути:

- Наявність великої кількості зовнішніх сутностей (десять і більше);
- Розподілена природна система;
- Багатофункціональність системи із сформованим або виявленим групуванням функцій в окремій підсистемі.

Для складних ІС будується ієрархія контекстних діаграм. При цьому контекстна діаграма верхнього рівня містить не єдиний головний процес, а набір підсистем, з'єднаних потоками даних. Контекстні діаграми наступного рівня деталізують контекст і структуру підсистем.

Ієрархія контекстних діаграм визначає взаємодію основних функціональних підсистем проектованої ІС як між собою, так і з зовнішніми вхідними і вихідними потоками даних та зовнішніми об'єктами (джерелами і приймачами інформації), із якими взаємодіє ІС.

Розробка контекстних діаграм вирішує проблему визначення функціональної структури ІС на початковій стадії її проектування, що особливо важливо для складних багатофункціональних систем, у розробці яких беруть участь різні організації і колективи розробників.

Після побудови контекстних діаграм отриману модель варто перевіряти на повноту вихідних даних про об'єкти системи та ізольованість об'єктів (відсутність інформаційних зв'язків з іншими об'єктами).

Для кожної системи, що розміщується на контекстних діаграмах, виконують її деталізацію за допомогою ДПД. Кожний процес на ДПД, у свою чергу, може бути деталізований за допомогою ДПД або мініспецифікації.

Під час деталізації мають виконувати такі правила:

- **Правило балансування** – означає, що під час деталізації підсистеми або процесу діаграма, яка деталізує як зовнішні джерела/приймачі даних, може мати тільки такі компоненти (підсистеми, процеси, зовнішні сутності, накопичувачі даних), із якими має інформаційний зв'язок підсистема або процес на батьківській діаграмі, що деталізується;
- **Правило нумерації** – означає, що під час деталізації процесів має підтримуватися їхня ієрархічна нумерація. Наприклад, процеси, що деталізують процеси із номером 12, одержують номери 12.1, 12.2, 12.3 і т.д.

Мініспецифікація (опис логіки процесу) має формулювати його основні функції так, щоб надалі фахівець, який виконує реалізацію проекту, зміг виконати їх або розробити відповідну програму.

Мініспецифікація є кінцевою вершиною ієрархії ДПД. Рішення щодо завершення деталізації процесу і використання мініспецифікації приймає аналітик, виходячи з таких критеріїв:

- Наявність у процесі незначної кількості вхідних і вихідних потоків даних (2-3 потоки);
- Можливість опису перетворення даних процесом у вигляді послідовного алгоритму;
- Виконання процесом єдиної логічної функції перетворення вхідної інформації на вихідну;
- Можливість опису логіки процесу за допомогою мініспецифікації невеликого обсягу (не більше 20-30 рядків).

Під час побудови ієрархії ДПД переходити до деталізації процесів необхідно тільки після визначення змісту всіх потоків і накопичувачів даних, що описують за допомогою структур даних. **Структури даних** будують з елементів даних і можуть містити альтернативи, умовні входження та ітерації. Умовне входження означає, що даного компонента може не бути у структурі. Альтернатива означає, що в структуру може входити один із перерахованих елементів; ітерація – входження будь-якої кількості елементів у зазначеному діапазоні.

Після побудови кінцевої моделі системи її необхідно верифікувати (перевіряти на повноту і узгодженість). У повній моделі усі її об’єкти (підсистеми, процеси, потоки даних) мають бути докладно описані і деталізовані. Виявлені недеталізовані об’єкти необхідно деталізувати, повернувшись на попередні кроки розробки. В узгодженій моделі для всіх потоків і накопичувачів даних має виконуватися правило

збереження інформації: усі дані, що надходять, мають бути зчитані, а всі зчитані дані слід записати.

Контрольні запитання:

1. В яких аспектах виявляється обмеженість реляційної моделі даних?
2. Опишіть можливості застосування семантичних моделей даних.
3. Які основні компоненти діаграм потоків даних?
4. Які етапи побудови семантичної моделі БД?
5. Що таке зовнішня сутність?
6. Опишіть побудову діаграм потоків.
7. Назвіть, які правила виконуються під час деталізації та дайте їм визначення.
8. Опишіть що таке мініспецифікація та її призначення.

Тестові завдання:

1.Вірним визначенням правила балансування є:

а) формулюються його основні функції так, щоб надалі фахівець, який виконує реалізацію проекту, зміг виконати їх або розробити відповідну програму.

б) під час деталізації підсистеми або процесу діаграма, яка деталізує як зовнішні джерела/приймачі даних, може мати тільки такі компоненти (підсистеми, процеси, зовнішні сутності, накопичувачі даних), із якими має інформаційний зв'язок підсистема або процес на батьківській діаграмі, що деталізується;

в) під час деталізації процесів має підтримуватися їхня ієрархічна нумерація. Наприклад, процеси, що деталізують процеси із номером 12, одержують номери 12.1, 12.2, 12.3 і т.д.

г) правильної відповіді немає

2. Яке правило означає, що під час деталізації процесів має підтримуватися їхня ієрархічна нумерація. Наприклад, процеси, що деталізують процеси із номером 12, одержують номери 12.1, 12.2, 12.3 і т.д.

а) Правило нумерації

б) Правило балансування

в) Мініспецифікація

г) Правильної відповіді немає

3. Накопичувачі даних ідентифікують:
а) літерою D і довільним числом
б) літерою F і довільним числом
в) довільною літерою і цифрою 5
г) правильної відповіді немає

4. До визначення функцій формулюють такі вимоги:
а) Виробничі та функціональні
б) Технічні
в) Варіант а та б
г) Правильної відповіді немає

5. Скільки потрібно мати зовнішніх сутностей, щоб це являлося ознакою складності ІС:
а) Від 5 до 10
б) Десять і більше
в) Більше 3, але не більше 10
г) Правильної відповіді немає

Розділ 15

Проектування концептуальної схеми бази даних. ER – моделювання даних

- **Мета моделювання даних**
- **Таблиця основних термінів моделювання сутностей і зв'язків**
- **Основні типи зв'язків**
- **ER – моделювання за методикою Чена**
- **Приклади діаграм ER – типів**

Метою моделювання даних є забезпечення розробника ІС концептуальною схемою БД у формі однієї моделі або декількох локальних моделей, що відносно легко можуть бути відображені у будь-якій СКБД.

Моделювання сутностей і зв'язків належить до моделювання логічних структур даних, яке базується на припущенні, що всі елементи ПО можна подати у вигляді ідеальних прототипів – **сутностей.** Такі концептуальні сутності описуються в термінах їхніх характеристик, або **атрибутів (attribute).** Сутності пов'язані через дії, які вони виконують відносно одна до одної. Ці дії встановлюють **зв'язки** (relationship) між сутностями.

Таблиця 15.1 – **Основні терміни моделювання сутностей і зв'язків**

Таблиця	Визначення
Сутність (entity)	Реальний об'єкт (фізична особа, підприємство, подія, предмет), дані про який зберігаються. Множину об'єктів однієї суттєвості також називають класами сутностей.
Екземпляр сутності (entitycopy)	Конкретний представник класу сутностей. Наприклад, клієнт Кравченко є екземпляром сутностей КЛІЄНТ.
Підтип (subtype)	Клас сутностей, який є підмножиною більш великого типу сутностей, який називають **супертипом.** Наприклад, клас сутностей ПОЖЕЖНИК може бути підтипом СЛУЖБОВЕЦЬ.
Супертип (superlype)	Клас сутностей, що є надмножиною більш дрібних і вузьких класів сутностей, які називають **підтипами.** Наприклад, клас сутностей АВТОМОБІЛЬ може бути супертипом ФОРД_АВТО, КРАЙСЛЕР_АВТО і т.д.
Атрибут (attribute)	Характеристика сутності або зв'язку, призначена для класифікації, ідентифікації, кількості характеристики

	або стану. Атрибути докладно описують сутності.
Екземпляр атрибута (attributecopy)	Екземпляр атрибута визначається типом характеристики окремого екземпляра сутності і її значенням, яке називається **значенням атрибута.**
Домен (domain)	Вказаний тип даних або діапазон значень, який може набувати атрибут. Наприклад, домен **TDate** для атрибута Дата_Прийому.
Цілісність домену (domain integrity)	Правила, що визначають типи даних, які дозволяються доменом.
Унікальний ідентифікатор (unique identifier)	Атрибут або сукупність атрибутів, призначена для унікальної ідентифікації кожного екземпляра сутності.
Ідентифікатор сутності (entityidentifier)	Один із унікальних ідентифікаторів, обраний для унікальної ідентифікації кожного екземпляра сутності.
Зв'язок (relationship)	Пойменована асоціація (з'єднання) двох сутностей, за якої кожний екземпляр однієї (батьківської) сутності асоційований із деякою кількістю екземплярів іншої (дочірньої) сутності. При цьому зміна стану батьківської сутності приводить до зміни стану дочірньої сутності. Кожний екземпляр сутності-нащадка є асоційованим з одним екземпляром сутності-батька. Отже, екземпляр сутності-нащадка може існувати тільки у разі існування екземпляра сутності батька. Зв'язку подається ім'я виражене дієсловом, яке розміщується біля лінії зв'язку. Ім'я кожного зв'язку між двома сутностями має бути унікальним.
Цілісність зв'язків (relations integrity)	Правила, які забезпечують підтримку зв'язку між сутностями. Наприклад, цілісність зв'язків може перешкодити видаленню екземпляра сутності КЛІЄНТ у якого є зв'язок з екземпляром сутності-РАХУНОК.
Потужність зв'язків (connectivity)	Відображення характеристики зв'язку між екземплярами зв'язаних сутностей. Наприклад, за визначенням потужності сутностей РАХУНОК і КЛІЄНТ можна зазначити, що кожний екземпляр сутності-КЛІЄНТ може мати багато відповідних йому екземплярів класу сутностей РАХУНОК.

За характером з’єднання розглядають чотири види зв’язків: «один до одного», «один до багатьох», «багато до одного», «багато до багатьох».

Розглядають такі типи зв’язків:

- **Повний** – у зв’язку беруть участь усі екземпляри сутності;
- **Необов’язковий** – у зв’язку беруть участь не всі екземпляри сутності;
- **Обов’язковий** – екземпляри однієї сутності (залежної) можуть існувати тільки за наявності екземплярів іншої сутності (незалежної);
- **Слабкий** – екземпляр дочірньої «слабкої» сутності можна ідентифікувати тільки за допомогою екземпляра батьківської «сильної» сутності, тобто ключ «сильної сутності» є частиною ключа «слабкої сутності»;
- **«Супертип-підтип»** – загальні характеристики (атрибути) визначаються в батьківській сутності – супертипі, а дочірня сутність – підтип успадковує атрибути супертипу;
- **Асоціативний** – кожний екземпляр (асоціативний об’єкт) може існувати тільки за умови існування окремо визначених екземплярів кожної із взаємозалежних сутностей. Асоціативний об’єкт – це об’єкт, що є одночасно сутністю і зв’язком. Асоціативний зв'язок – зв'язок між декількома «незалежними» сутностями і однією «залежною». Зв'язок між незалежними сутностями має атрибути, які визначаються у залежній сутності. Отже, залежна сутність визначається в термінах атрибутів зв’язку між іншими сутностями;
- **Взаємовиключний** – зв'язок однієї сутності з декількома, за яких факт участі екземпляра в одному зв’язку веде до неможливості участі його в іншому;
- **Рекурсивний** – зв'язок об’єкта із самим собою;
- **Незсувний** – зв'язок, у якому екземпляр сутності не може бути зсунутий з одного екземпляра зв’язку в інший;
- **Ідентифікуючий** – екземпляр сутності-нащадка однозначно визначається своїм зв’язком із сутністю-батьком;
- **Не ідентифікуючий** – екземпляр сутності-нащадка не визначається однозначно своїм зв’язком із сутністю-батьком.

ER-моделювання проводиться з метою отримання зручного візуального представлення об’єктів і зв’язків між ними як концептуальної моделі БД. На базі цієї моделі проводиться наступне формування фізичної БД засобами СКБД.

ER-моделювання за методикою Чена

Найпоширенішим засобом семантичного моделювання даних є діаграми **сутність-зв'язок** (Entity Relations Diagram – ERD). За їх допомогою визначають важливі для ПО об’єкти (сутності), їх властивості (атрибути) та взаємовідношення (відношення між собою, тобто зв’язки). ERD можна безпосередньо використовувати для проектування реляційних БД.

Пітер Чен (Peter Chen) 1976 року опублікував специфікацію щодо підходу до реляційних структур як до набору зв’язків між сутностями. Ця праця та праці інших теоретиків, зокрема Хаммера (Hammer) та Мак-Леода (McLeod), які створили семантичну модель даних, стали основою для виникнення ERD, тобто ER-діаграм (діаграм **сутність-зв'язок**), які в наочній формі представляють зв’язки між сутностями і сьогодні є фундаментом моделювання логічних структур даних.

Існують два основні типи ER-діаграми, запропонованих П. Ченом. Перший тип використовує **стандартну нотацію Чена**. У діаграмі сутність зображено прямокутником, у середині якого наведено ім’я сутності. Зв'язок показано ромбом, усередині якого – ім’я зв’язку. Сутність поєднано із зв’язком за допомогою дуги, над якою вказано потужність зв’язку. Діаграми такого типу дуже спеціалізовані – кожна зазвичай представляє один зв'язок між двома сутностями. Приклад простої ER-моделі, що використовує стандартну нотацію Чена, показано на малюнку.

Рисунок 15.1 – ER-діаграма стандартного типу Чена

Другий тип діаграми називають **деталізованою ER-діаграмою.** Під час проектування громіздкої БД застосування методики Чена призводить до ER-моделей.Цього, як правило, можна уникнути, тому що всі зв’язки окремої сутності наводяться на одній діаграмі. Вказівки діаграми ґрунтуються на сутностях, а не на зв’язках. Особливістю деталізованих діаграм також є інформації про атрибути (характеристики) сутностей. Сутності, які визначають за допомогою ER-діаграм, відповідають відношенням (таблицям) реляційної моделі БД та її наступної реалізації як фізичного сховища структурованої інформації, діаграми використовують не тільки елементи моделювання сутностей і зв’язків, але й елементи реляційної моделі та фізичного проектування.

Слід мати на увазі, що в результаті проектування може бути отримано декілька варіантів однієї БД. Так, два різні проектувальники, розглядаючи одну і ту ж проблему з різних точок зору, можуть отримати різні набори суттєвостей і зв'язків. При цьому обидва варіанти можуть бути робочими, а вибір кращого з них буде результатом особистих переваг.

З метою підвищення наочності і зручності проектування для представлення суттєвостей, екземплярів і зв'язків між ними використовуються наступні графічні засоби:

- діаграми ER-екземплярів
- діаграми ER-типу, або ER-діаграми.

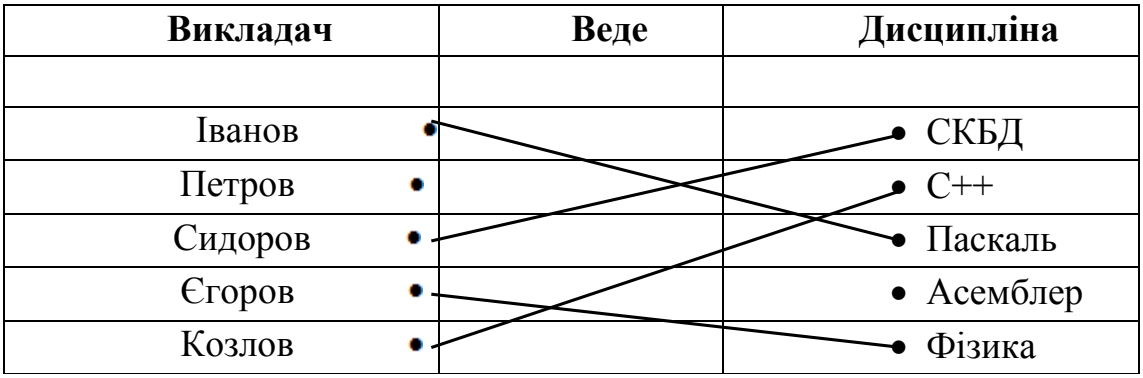


Рисунок 15.2 – Діаграма ER-екземплярів

На мал. приведена діаграма ER-екземплярів для суттєвостей ВИКЛАДАЧ і ДИСЦИПЛІНА із зв'язком ВЕДЕ.

Діаграма ER-екземплярів показує, яку конкретно дисципліну (СКБД, С++ і т.д.) веде кожний з викладачів. На рисунку 15.3 представлена діаграма ER-типу, відповідна розглянутій діаграмі ER-екземплярів.

Рисунок 15.3 – Діаграма ER-типу

На початковому етапі проектування БД виділяються атрибути, що становлять ключі суттєвостей.

На основі аналізу діаграм ER-типу формуються відношення БД. При цьому враховується **ступінь зв'язку** суттєвостей і **клас їх належності**, які, у свою чергу, визначаються на основі аналізу діаграм ER-екземплярів відповідних суттєвостей.

Ступінь зв'язку є характеристикою зв'язку між суттєвостями, який може бути типу: 1:1, 1:M, M:1, M:M.

Клас приналежності (КН) суттєвості може бути: **обов'язковим** і **необов'язковим.**

Клас приналежності суттєвості є **обов'язковим**, якщо всі екземпляри цієї суттєвості обов'язково беруть участь в даному зв'язку, в іншому випадку клас належності суттєвості є **необов'язковим.**

Варіюючи класом належності суттєвостями для кожного з названих типів зв'язку, можна отримати декілька варіантів діаграм ER-типу. Розглянемо приклади деяких з них.

Приклад 1. Зв'язки типу 1:1 і необов'язковий клас належності (рис. 15.3).

В приведеній на рисунку діаграмі ступінь зв'язку між суттєвостями 1:1, а клас належності обох суттєвостей необов'язковий. Дійсно, з рисунку видно наступне:

• кожний викладач веде не більше однієї дисципліни, а кожна дисциплінаведеться не більше ніж одним викладачем (ступінь зв'язку 1:1);

• деякі викладачі не ведуть жодної дисципліни і є дисципліни, які не веде жоден з викладачів (клас належності обох суттєвостей необов’язковий).

Приклад 2. Зв'язки типу 1:1 і обов'язковий клас приналежності. На рисунку приведені діаграми, у яких ступінь зв'язку між суттєвостями 1:1, а клас належності обох суттєвостей обов'язковий.

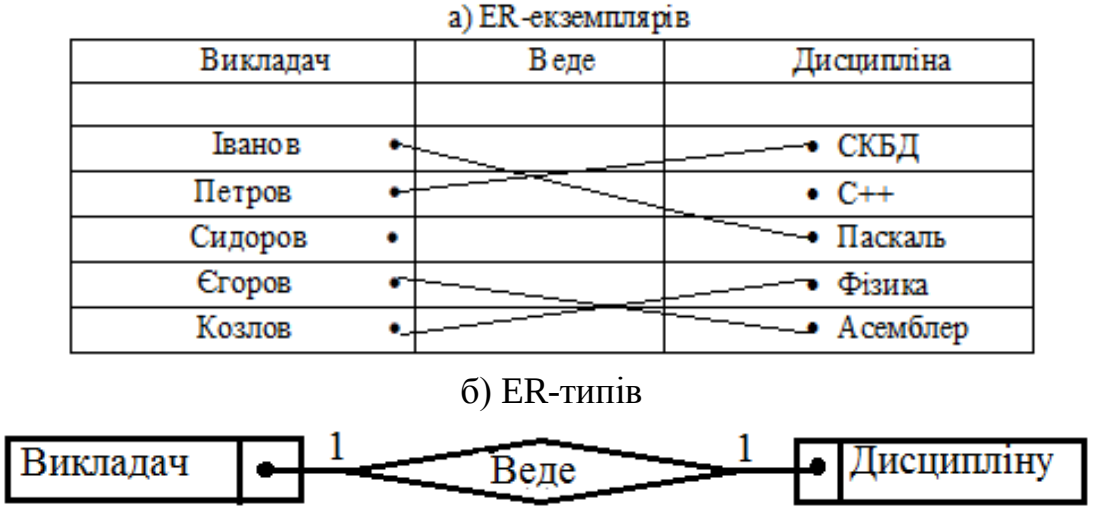


Рисунок 15.4 – Діаграми для зв'язку 1:1 і обов'язковим КП обох суттєвостей

В цьому випадку кожний викладач веде одну дисципліну і кожна дисципліна ведеться одним викладачем.

Можливі два проміжні варіанти з необов'язковим класом належності одної з суттєвостей.

Зауваження.

• На діаграмах ER-типу **обов'язкова** участь в зв'язку екземплярів суттєвостей наголошується блоком з крапкою всередині, суміжним з блоком цієї суттєвості.

• При **необов'язковій** участі екземплярів суттєвостей в зв'язку додатковий блок до блоку суттєвостей не приставляється, а крапка розміщується на лінії зв'язку.

• Символи на лінії зв'язку вказують на ступінь зв'язку.

Під кожним блоком, відповідним деякій суттєвості, вказується її ключ, що виділяється підкресленням. Багатокрапка за ключовими атрибутами означає, що можливі інші атрибути суттєвостей, але жоден з них не може бути частиною її ключа. Ці атрибути виявляються після формування відношень.

На практиці ступінь зв'язку і клас належності суттєвостей при проектуванні БД визначається специфікою наочної області. Розглянемо приклади варіантів із ступенем зв'язку 1:M або M:1.

Приклад 3. Зв'язки типу 1:M.

Кожний викладач може вести декілька дисциплін, але кожна дисципліна ведеться одним викладачем.

Приклад 4. Зв'язки типу M:1.

Кожний викладач може вести одну дисципліну, але кожну дисципліну можуть вести декілька викладачів.

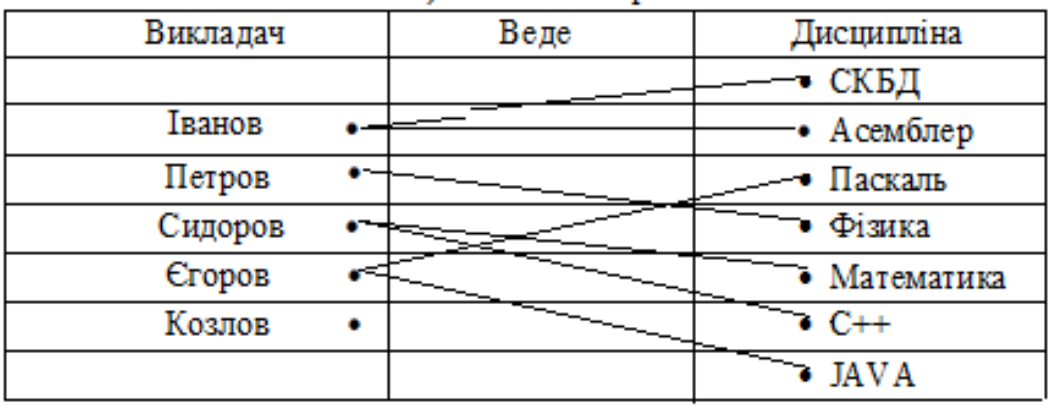
Приклади з типом зв'язку 1:M або M:1 можуть мати ряд варіантів, відмінних класом належності однієї або обох суттєвостей. Позначимо обов'язковий клас належності символом «О», а необов'язковий - символом «Н», тоді варіанти для зв'язку типу 1:M умовно можна представити як: О-О, О-Н, Н-О, Н-Н. Для зв'язку типу M:1 також є 4 аналогічні варіанти.

Приклад 5. Зв'язки типу 1:M варіант Н-О.

Кожний викладач може вести декілька дисциплін або жодної, але кожна дисципліна ведеться одним викладачем.

Аналогічно легко скласти діаграми і для решти варіантів.

а) ER-екземплярів



б)ER-типів

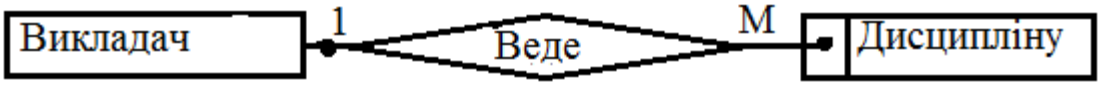


Рисунок 15.5 – Діаграми для зв'язку типу 1:M варіанту Н-О

Приклад 6. Зв'язки типу M:M.

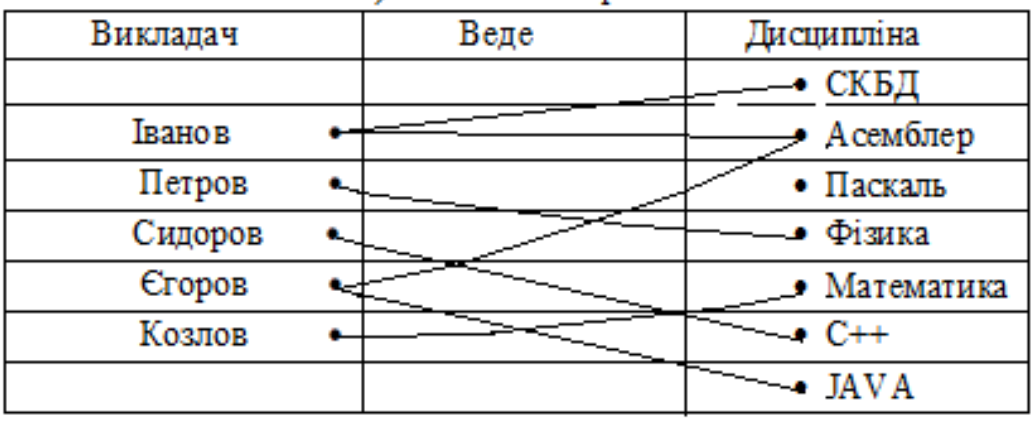
Кожний викладач може вести декілька дисциплін, а кожна дисципліна може вестися декількома викладачами.

Як і в разі інших типів зв'язків, для зв'язку типу M:M можливі 4 варіанти, відмінні класом належності суттєвостей.

Приклад 7. Зв'язки типу M:M і варіант класу належності О-Н.

Припустимо, що кожний викладач веде не менше однієї дисципліни, а дисципліна може вестися більш ніж одним викладачем, є і такі дисципліни, які ніхто не веде. Відповідні цьому випадку діаграми приведені на рисунку.

а) ER-екземплярів



б) ER-типів

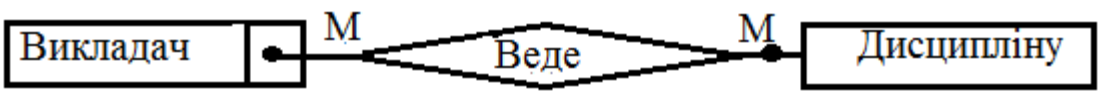


Рисунок 15.6 – Діаграми для зв'язку типу M : M і варіанту О-Н

Виявлення суттєвостей і зв'язків між ними, а також формування на їх основі діаграм ER-типу виконується на початкових етапах методу **суттєвість-зв'язок.**

Контрольні запитання:

1. Яка мета моделювання даних?
2. Напишіть основні терміни моделювання сутностей і зв’язків.
3. Що таке сутність, екземпляр сутності, підтип та супер тип?
4. Дайте визначення термінам: атрибут та екземпляр атрибут.
5. Дайте визначення термінам: домен, цілісність домену, унікальний ідентифікатор, ідентифікатор сутності.
6. Що таке зв'язок, цілісність та потужність зв’язків?
7. Як існують типи зв’язків?
8. Наведіть приклад простої ER-діаграми стандартного типу Чена?

Тестові завдання:

1. Реальний об’єкт (фізична особа, підприємство, подія, предмет), дані про який зберігаються. Множину об’єктів однієї суттєвості також називають класами сутностей. Цьому визначенню відповідає термін:

а) Екземпляр сутності
б) Домен
в) Сутність
г) Правильної відповіді немає

2. Унікальний ідентифікатор – це:

а) Правила, що визначають типи даних, які дозволяються доменом.
б)Один із унікальних ідентифікаторів, обраний для неунікальної ідентифікації кожного екземпляра сутності.
в) Атрибут або сукупність атрибутів, призначена для унікальної ідентифікації кожного екземпляра сутності.
г) Правильної відповіді немає

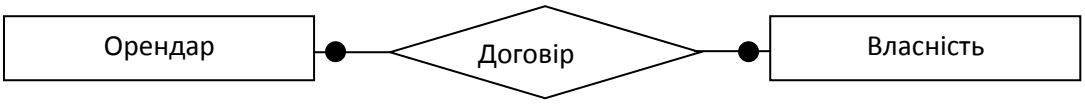
3. Зв'язок об’єкта із самим собою – це зв'язок:

а) Взаємовиключний
б) Асоціативний
в) Повний

г) Правильної відповіді немає

4. Екземпляр сутності – це:
а) Конкретний представник класу сутностей
б) Реальний об’єкт, дані про який зберігаються
в) Клас сутностей, що є надмножиною більш дрібних і вузьких класів сутностей
г) Клас сутностей, який є підмножиною більш великого типу сутностей

5. На малюнку зображено:



а) Діаграма ER-типу
б) Діаграма ER-екземпляру
в) Діаграми для зв'язку типу M : M
г) Правильної відповіді немає

Розділ 16

Етапи проектування баз даних

- **Етапи проектування**
- **Правила формування відношень для зв’язку 1:1**
- **Правила формування відношень для зв’язку 1:М**
- **Правила формування відношень для зв’язку М:М**
- **Приклад проектування БД учбової частини**
- **Засоби автоматизації проектування. Основні визначення**
- **Перспективна Case– система**

Процес проектування бази даних є ітераційним - допускаючи повернення до попередніх етапів для перегляду раніше ухвалених рішень і включає наступні етапи:

1.Виділення суттєвостей і зв'язків між ними.

2.Побудова діаграм ER-типу з урахуванням всіх суттєвостей і їх зв'язків.

3.Формування набору попередніх відношень з вказівкою передбачуваного первинного ключа для кожного відношення і використанням діаграм ER-типу.

4.Додавання не ключових атрибутів у відношення.

5.Приведення попередніх відношень до нормальної форми Бойса-Кодда, наприклад, за допомогою методу нормальних форм.

6. Перегляд ER-діаграм в наступних випадках:

• деякі відношення не приводяться до нормальної форми Бойса-Кодда;

• деяким атрибутам не знаходиться логічно обґрунтованих місць в попередніх відношеннях.

Після перетворення ER-діаграм здійснюється повторне виконання попередніх етапів проектування (повернення до етапу 1).

Одним з ключових етапів проектування є етап формування відношень. Розглянемо процес формування попередніх відношень, що становлять первинний варіант схеми БД.

В розглянутих вище прикладах зв'язок ВЕДЕ завжди сполучає дві суттєвості і тому є **бінарним.** Сформульовані нижче правила формування відношень з діаграм ER-типу розповсюджуються саме на бінарні зв'язки. Тому, коли йдеться про зв'язки, слово «бінарні» далі опускається.

Правила формування відношень

Правила формування відношень ґрунтуються на обліку наступного:

• ступені зв'язку між суттєвостями (1:1, 1:M, M:1, M:M);

• класу приналежності екземплярів суттєвостей (обов'язковий і необов’язковий).

Розглянемо формулювання шести правил формування відношень на основі діаграм ER-типу.

Формування відношень для зв'язку 1:1

Правило 1. Якщо ступінь бінарного зв'язку 1:1 і клас належності обох суттєвостей обов'язковий, то формується одне відношення. Первинним ключем цього відношення може бути ключ будь-якої з двох суттєвостей.

На мал. приведені діаграма ER-типу і відношення, сформоване за правилом 1 на її основі.

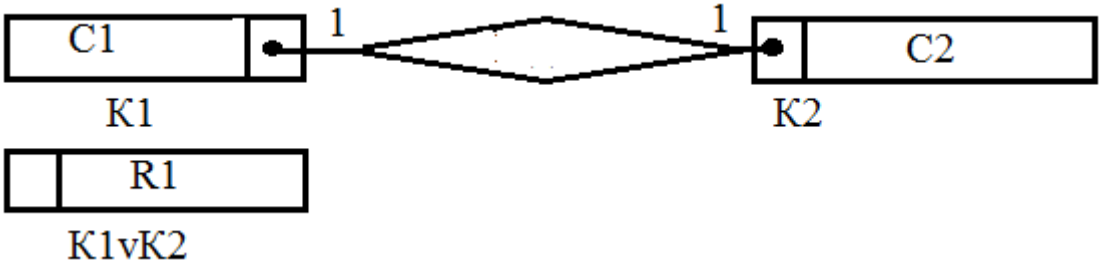


Рисунок 16.1 – Діаграма і відношення для правила 1

На малюнку використовуються наступні позначення:

Cl, C2 - суттєвості 1 і 2;

Kl, K2 - ключі першої і другої суттєвостей відповідно;

R1 - відношення 1, сформоване на основі першої і другої суттєвостей;

KlvK2... означає, що ключем сформованого відношення може бути або К1, або К2.

Перевіримо і інші правила, розглядаючи різні варіанти зв'язку ВИКЛАДАЧ ВЕДЕ ДИСЦИПЛІНУ. Припустимо суттєвість ВИКЛАДАЧ характеризується атрибутами НП (ідентифікаційний номер викладача), ПІБ(прізвище, ім'я і по батькові), Стаж (стаж викладача). Суттєвість ДИСЦИПЛІНА характеризується відповідно атрибутами КД (код дисципліни), Години (години, що відводяться на дисципліну). Тоді схема відношення, що містить інформацію про обидві суттєвості, і саме відношення для випадку, коли ступінь зв'язку

рівний 1:1, а КН (клас належності) обов'язковий для всіх суттєвостей, можуть мати вигляд, показаний на рисунку 16.2.

ВИКЛАДАЧ_ДИСЦИПЛІНА(**НП, ПІБ**, Стаж, КД, Години)

ВИКЛАДАЧ__ДИСЦИПЛИНА

НВ	ПІБ	Стаж	КД	Години
В1	Іванов	5	К1	62
В2	Петров	7	К2	74
В3	Сидоров	10	К3	102
В4	Єгоров	5	К4	80

Рисунок 16.2 – Отримані за правилом 1 схема і відношення

Сформоване відношення містить повну інформацію про викладачів, дисципліни і про те, як вони зв'язані між собою. Так, викладач Іванов веде тільки дисципліну з кодом К1, а дисципліна К1 ведеться тільки Івановим (зв'язок 1:1). В цьому відношенні відсутні порожні поля (КН обов'язковий для всіх суттєвостей), оскільки немає викладачів, які б щось не вели, і немає дисциплін, які ніхто не веде. Таким чином, одного відношення в даному випадку достатньо. Як первинний ключ може бути вибраний ключ першого відношення НВ або ключ другого відношення КД.

Правило 2: якщо ступінь зв'язку 1:1 і клас належності однієї суттєвості обов'язковий, а друге - необов'язковий, то під кожну з суттєвостей формується по відношенню з первинними ключами відповідних суттєвостей. Далі до відношення, суттєвість якого має обов'язковий КН, додається як атрибут ключ суттєвості з необов'язковим КН.

На рисунку приведені діаграма ER-типу і відношення, сформовані за правилом 2 на її основі.

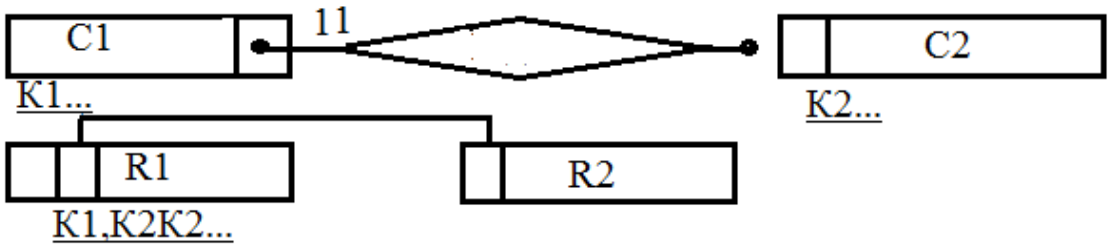


Рисунок 16.3 – Діаграма і відношення для правила 2

Щоб переконатися в справедливості правила, розглянемо наступний приклад. На мал. приведено початкове відношення, що містить інформацію про викладачів і дисципліни. Воно представляє варіант, в

якому клас суттєвості ВИКЛАДАЧ є обов'язковим, а в суттєвості ДИСЦИПЛІНА - необов'язковим. При цьому пропуски «—» (порожні поля) присутні у всіх кортежах з інформацією про дисципліни, які не ведуться жодним з викладачів.

ВИКЛАДАЧ_ДИСЦИПЛІНА

НВ	ПІБ	Стаж	КД	Години
В1	Іванов І.М.	5	К1	62
В2	Петров М.І.	7	К2	74
В3	Сидоров Н.Г.	10	К3	102
...	...	...	К4	80

Рисунок 16.4 – Початкове відношення

Уникнути цієї ситуації можна, застосувавши **правило 2,** відповідно до якого, виділяються два відношення, приведені на рисунку.

ВИКЛАДАЧ(**НП**, ПІБ, Стаж, КД)

ВИКЛАДАЧ

НВ	ПІБ	Стаж	КД
В1	Іванов І.М.	5	К1
В2	Петров М.І.	7	К2
В3	Сидоров Н.Г.	10	К3
В4	Єгоров	5	К4

ДИСЦИПЛІНА(**КД,** Години)

ДИСЦИПЛІНА

КД	Години
К1	62
К2	74
К3	102
К4	80

Рисунок 16.5 – Відношення, отримані за правилом 2

В результаті ми уникнули порожніх полів у відношеннях, не втративши даних. Додавши атрибут КД - ключ суттєвості ДИСЦИПЛІНА (з необов'язковим КН) як зовнішній ключ у відношення, відповідної суттєвості ВИКЛАДАЧ (з обов'язковим КН), ми зв'язали відношення.

НВ	ПІБ	Стаж	КД

КД	Годинник

Рисунок 16.6 – Зв'язок відношень за зовнішнім ключем

Точніше кажучи, ми створили умови для зв’язку відношень. Цей зв'язок при роботі з базою даних дозволяє, наприклад, одержувати одночасно дані про викладача і про дисципліни, що ними ведуться (години).

Правило 3. Якщо ступінь зв'язку 1:1 і клас належності обох суттєвостей є необов'язковим, то необхідно використовувати три відношення. Два відношення відповідають зв'язуваним сутностям, ключі яких є первинними в цих відношеннях. Третє відношення є зв'язним між першими двома, тому його ключ об'єднує ключові атрибути зв'язуваних відношень.

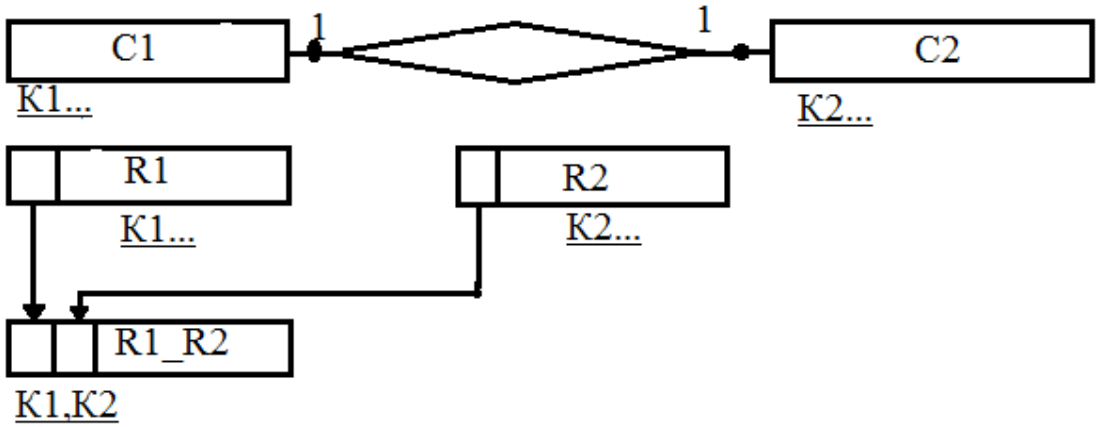


Рисунок 16.7 – Діаграма і відношення для правила 3

На рисунку приведені діаграма ER-типу і відношення, сформовані за правилом 3 на її основі.

Нарисунку наведені приклади відношень, підтверджуючі необхідність використання трьох відношень за наявності необов'язкового КН для обох зв'язуваних суттєвостей.

а) одне відношення

ВИКЛАДАЧ_ДИСЦИПЛІНА

НВ	ПІБ	Стаж	КД	Години
В1	Іванов	5	К1	62
В2	Петров	7	…	…
В3	Сидоров	10	К2	74
…	…	…	К3	102

б) два відношення

ВИКЛАДАЧ

ВП	ПІБ	Стаж	КД
В1	Іванов	5	К1
В2	Петров	7	…
В3	Сидоров	10	К2

ДИСЦИПЛІНА

КД	Години	НВ
К1	62	В1
К2	74	В2
К3	102	…

в) три відношення

ВИКЛАДАЧ

НВ	ПІБ	Стаж
В1	Іванов	5
В2	Петров	7
В3	Сидоров	10

ВЕДЕ

НВ	КД
В1	К1
В2	К2

ДИСЦИПЛІНА

КД	Години
К1	62
К2	74
К3	102

Рисунок 16.8 – Варіанти відношень для правила 3

Використання одного відношення в даному випадку приводить до наявності небажаних порожніх полів в цьому відношенні. При використанні двох відношень нам довелося додати ключі кожної з суттєвостей у відношення, щоб не втратити інформацію про те, яку дисципліну веде кожний викладач і навпаки. При цьому також з'явилися порожні поля.

Вихід полягає у використанні трьох відношень, сформованих за правилом 3. Об'єктні відношення (з атрибутами суттєвостей) містять повну інформацію про всіх викладачів і дисципліни відповідно. Зв'язне відношення ВЕДЕ містить дані про викладачів, які ведуть дисципліни і про дисципліни, які ведуться викладачами. При цьому в ньому є тільки одна згадка про кожного викладача і дисципліну через зв'язок 1:1. Це відношення містить в даному випадку тільки ключові атрибути обох суттєвостей, але може мати і інші атрибути, що характеризують цей зв'язок. Наприклад, номер семестру, в якому викладач веде дисципліну.

Отже, сформульовано три правила, які дозволяють формувати відношення на основі ER-діаграм, для варіантів із ступенем зв'язку типу 1:1. Сформулюємо аналогічні два правила для варіантів, ступінь зв'язку між суттєвостями яких 1:M.

Формування відношень для зв'язку 1:M

Якщо дві суттєвості С1 і С2 зв'язано як 1:М, суттєвість С1 називатимемо однозв'язковою (1-зв'язковою), а суттєвість С2 - багатозв'язковою (М-зв'язковою). Визначальним чинником при формуванні відношень, зв'язаних цим видом зв'язку, є клас приналежності М-зв'язкової суттєвості. Так, якщо клас приналежності М-зв'язкової суттєвості обов'язковий, то в результаті застосування правила отримаємо два відношення, якщо необов'язковий - три відношення. Клас приналежності однозв'язної суттєвості не впливає на результат.

Щоб переконатися в цьому, розглянемо відношення ВИКЛАДАЧ_ДИСЦИПЛІНА, відповідне діаграмам, приведеним на рисунку, тобто випадку, коли: зв'язок типу 1:М, клас приналежності М-зв'язкової суттєвості обов'язковий, 1-зв'язкової - необов'язковий.

ВИКЛАДАЧ_ДИСЦИПЛІНА

НВ	ПІБ	Стаж	КД	Години
В1	Іванов	5	К1	62
В1	Іванов	5	К2	74
В2	Петров	7	К4	80
В3	Сидоров	10	К5	96
В3	Сидоров	10	К6	120
В4	Єгоров	5	К3	102
В4	Єгоров	5	К7	89
В5	Козлов	8	---	---

Рисунок 16.9 – Початкове відношення

З відношенням ВИКЛАДАЧ_ДИСЦИПЛІНА пов'язані наступні проблеми:

- є кортежі з порожніми полями (викладач не веде дисципліни)
- надмірне дублювання даних (повторюється стаж викладача) в кортежах з відомостями про викладачів, якщо ведуться декілька дисциплін.

Якби клас приналежності 1-зв'язної суттєвості був обов'язковим (немає викладача без дисципліни), то зникли б порожні поля, але дані в атрибутах викладача, що повторюються, збереглися б. Для усунення названих проблем відношення можуть бути сформовані за наступним правилом.

Правило 4. Якщо ступінь зв'язку між суттєвостями 1:М (або М:1) і клас приналежності М-зв'язкової суттєвості обов'язковий, то достатньо формування двох відношень (по одному на кожну з суттєвостей). При цьому первинними ключами цих відношень є ключі їх суттєвостей.

Крім того, ключ 1-зв'язної суттєвості додається як атрибут (зовнішній ключ) у відношення, відповідної М-зв'язкової суттєвості.

На рисунку приведені діаграма ER-типу і відношення, сформовані за правилом 4.

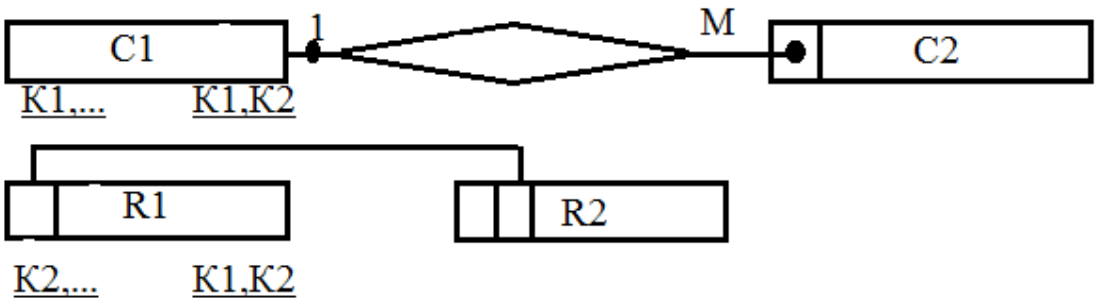


Рисунок 16.10 – Діаграма і відношення для правила 4

Відповідно до правила 4 перетворимо відношення на рисунку в два відношення. З малюнка видно, що порожні поля і дублювання інформації вдалося усунути.

ВИКЛАДАЧ

НВ	ПІБ	Стаж
В1	Іванов	5
В2	Петров	7
В3	Сидоров	10
В4	Єгоров	5
В5	Козлов	8

ДИСЦИПЛІНА

КД	Години	НВ
К1	62	В1
К2	74	В1
К3	102	В4
К4	80	В2
К5	96	В3
К6	120	В3
К7	89	В4

Рисунок 16.11 – Відношення, отримані за правилом 4

Втрати відомостей про те, хто з викладачів веде яку дисципліну, не відбулося завдяки введенню ключа НП суттєвостей ВИКЛАДАЧ як зовнішній ключ у відношення ДИСЦИПЛІНА.

Для формування і обґрунтування необхідності використання наступного правила розглянемо наступний приклад.

Правило 5. Якщо ступінь зв'язку 1:М (М:1) і клас належності М-зв'язкової суттєвості є необов'язковим, то необхідне формування трьох відношень.

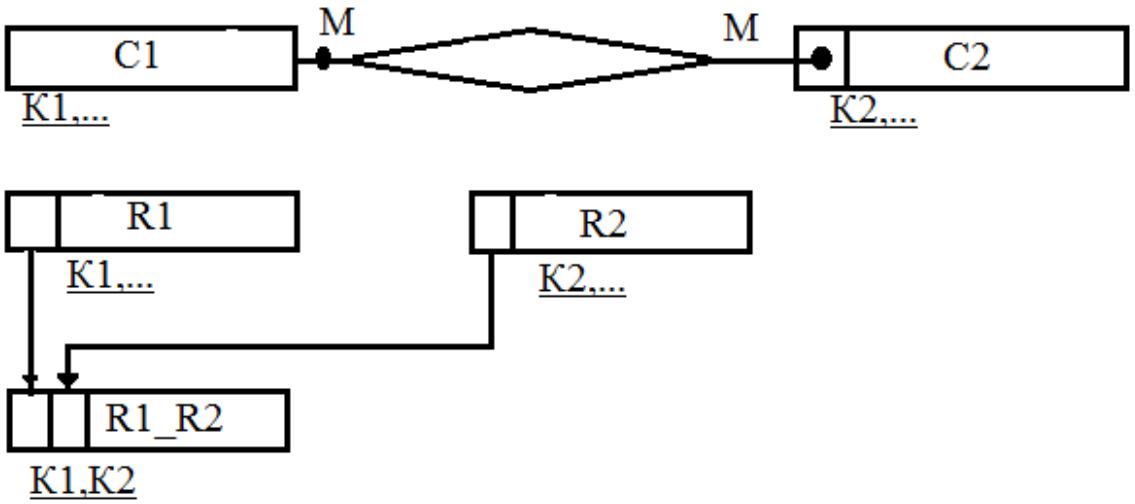


Рисунок 16.12 – Діаграма і відношення для правила 5

Два відношення відповідають зв'язуваним сутностям, ключі яких є первинними в цих відношеннях. Третє відношення є зв'язним між першими двома (його ключ об'єднує ключові атрибути зв'язуваних відношень).

В результаті використання правила 5 до даного відношення що містяться в ньому дані розподіляються по трьох відношеннях.

ВИКЛАДАЧ

НВ	ПІБ	Стаж
В1	Іванов	5
В2	Петров	7
В3	Сидоров	10
В4	Егоров	5
В5	Козл	8

ВЕДЕ ДИСЦИПЛІНА

НВ	КД
В1	К1
В1	К2
В2	К4
В3	К6
В4	К3
В4	К7

КД	Години
К1	62
К2	74
К3	102
К4	80
К5	96
К6	120
К7	89

Рисунок 16.13 – Відношення, отримані за правилом 5

Таким чином, вказані проблеми вдалося вирішити. Ключ в зв'язному відношенні ВЕДЕ є складним і включає ключові атрибути обох зв'язуваних відношень (суттєвостей). В практичних ситуаціях зв'язне відношення може містити і інші характеризуючі зв’язки атрибутів.

Підкреслимо, що визначальним чинником при виборі між 4-м або 5-м правилом є клас приналежності М-зв'язкової суттєвості.

Формування відношень для зв'язку М:М

За наявності зв'язку М:М між двома суттєвостями необхідні три відношення незалежно від класу належності будь-якої з суттєвостей. Використання одного або двох відношень в цьому випадку не позбавляє від порожніх полів або надмірно дубльованих даних.

Правило 6. Якщо ступінь зв'язку М:М, то незалежно від класу належності суттєвостей формуються три відношення. Два відношення відповідають зв'язуваним суттєвостям і їх ключі є первинними ключами цих відношень. Третє відношення є зв'язним між першими двома, а його ключ об'єднує ключові атрибути зв'язуваних відношень.

На мал. приведені діаграма ER-типу і відношення, сформовані за правилом 6. Нами показаний варіант з класом приналежності суттєвостей Н-Н, хоча, згідно правилу 6, він може бути довільним.

Застосуємо правило 6 наприклад, приведеному на мал. раніше. В ньому ступінь зв'язку рівний М:М, клас належності для суттєвості ВИКЛАДАЧ обов'язковий,а для суттєвості ДИСЦИПЛІНА - необов'язковий. Відповідно цьому прикладу початкове відношення показано на рисунку 16.14.

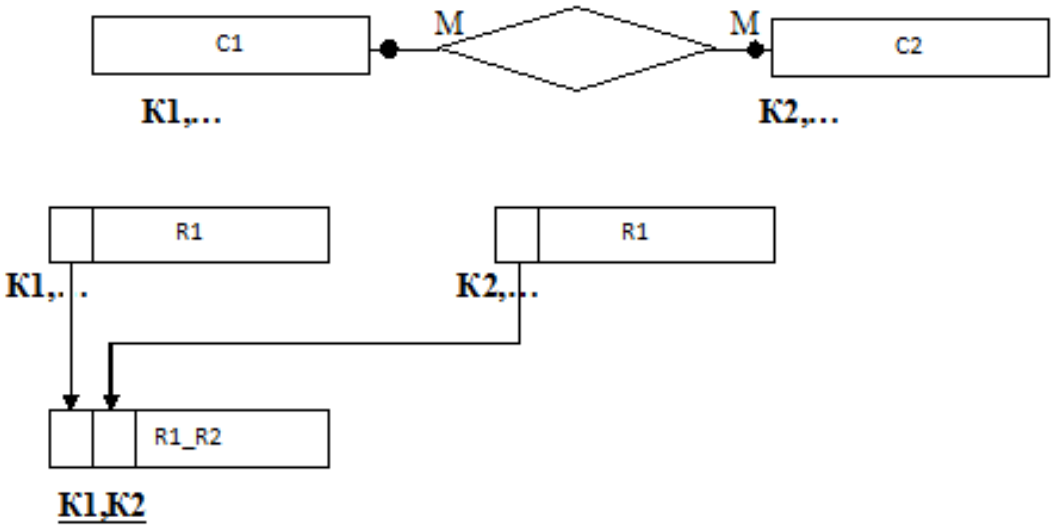


Рисунок 16.14 – Діаграма і відношення для правила 6

Розглянемо початкове відношення, що на рисунку 16.15.

ВИКЛАДАЧ_ДИСЦИПЛІНА

НВ	ПІБ	Стаж	КД	Години
В1	Іванов	5	К1	62
В1	Іванов	5	К2	74
В2	Петров	7	К4	80
---	---	---	К3	102
В3	Сидоров	10	К6	120
В4	Єгоров	5	К2	74
В4	Єгоров	5	К7	89
В5	Козлов	8	К5	96

Рисунок 16.15 – Початкове відношення

В результаті використання правила 6 виходять три відношення.

ВИКЛАДАЧ

НВ	ПІБ	Стаж
В1	Іванов	5
В2	Петров	7
В3	Сидоров	10
В4	Егоров	5
В5	Козл	8

ДИСЦИПЛІНА

КД	Годинник
К1	62
К2	74
К3	102
К4	80
К5	96
К6	120
К7	89

ВЕДЕ

НВ	КД
В1	К1
В1	К2
В2	К4
В3	К6
В4	К3
В4	К7

Рисунок 16.16 – Відношення, отримані за правилом 6

Аналогічні результати виходять і для трьох інших варіантів, що розрізняються класами належності їх суттєвостей. Розглянемо використання сформульованих правил на прикладах проектування БД методом сутність-зв'язок.

Приклад проектування БД учбової частини

Застосуємо описані правила на прикладі, розглянутому при вивченні методу нормальних форм. Нагадаємо, що йшлося про БД для учбової частини, що містить наступні відомості:

ПІБ - прізвище і ініціали викладача (можливість збігу прізвища і ініціалів у викладачів виключена).

Посада - посада, що займає викладач.

Ставка - оклад викладача.

Стаж - викладацький стаж.

Н_Стаж - надбавка за стаж.

Каф - номер кафедри, на якій працює викладач.

Предм - назва дисципліни (предмету), що ведеться викладачем.

Група - помер групи, в якій викладач проводить заняття.

ВидЗан - вид занять, що проводяться викладачем в учбовій групі (викладач в одній групі веде тільки один вид занять).

При проектуванні дотримуватимемося етапів проектування.

Перший етап проектування - виділення суттєвостей і зв'язків між ними.

Виділимо наступні суттєвості:

- ВИКЛАДАЧ (Ключ - ПІБ)
- ЗАНЯТТЯ (Ключ - Група, Предм)
- СТАЖ (Ключ - Стаж)
- ПОСАДА (Ключ - Посада).

Виділимо зв'язки між суттєвостями:

- ВИКЛАДАЧ МАЄ СТАЖ
- ВИКЛАДАЧ ВЕДЕ ЗАНЯТТЯ
- ВИКЛАДАЧ ПОСІДАЄ ПОСАДУ.

Другий етап проектування - побудова діаграми ER-типу з врахуванням всіх суттєвостей і зв'язків між ними. Діаграма ER-типу для даного прикладу.

Зв'язок МАЄ є зв'язком типу M:1, оскільки однаковий стаж можуть мати декілька викладачів. Суттєвість ВИКЛАДАЧ має обов'язковий клас належності, оскільки кожний викладач має свій стаж. Суттєвість СТАЖ має необов'язковий клас належності, оскільки можливі такі значення стажу, яких не має жоден з викладачів.

Зв'язок ВЕДЕ має тип M:M, оскільки викладач може вести декілька занять, а кожне заняття може проводитися декількома викладачами. Заняття може бути лекційним або практичним, проводиться викладачем

в учбовій групі по одній з дисциплін. Обидві суттєвості в даному зв'язку мають КН обов'язковий, в припущенні, що немає викладачів, які не проводять занять, і немає занять, які не забезпечені викладачами.

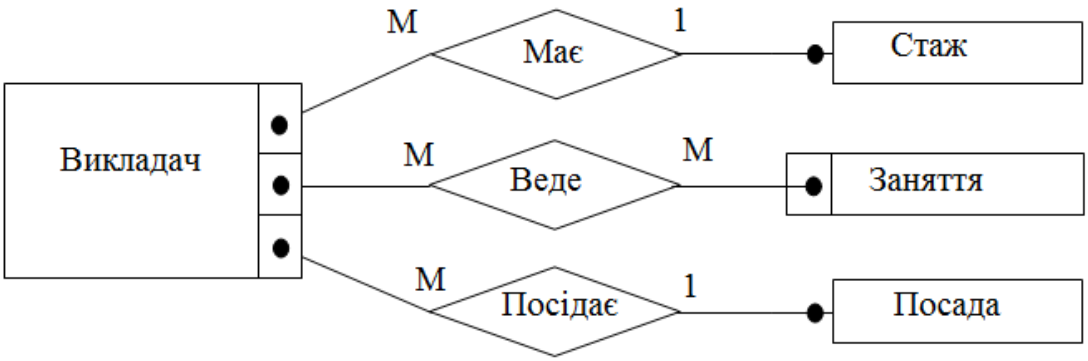


Рисунок 16.17 – Діаграма ER-типу

Зв'язок ЗАЙМАЄ має тип M:1, оскільки кожний викладач посідає певну посаду і однакові посади можуть займати декілька викладачів. Суттєвість ВИКЛАДАЧ має обов'язковий клас приналежності, оскільки припускаємо, що кожний викладач посідає посаду. Суттєвість ПОСАДУ має необов'язковий КН, оскільки не виключаємо, наприклад, відсутність посади професора на кафедрі, а значить і викладача, який її займає.

Третій етап проектування - формування набору попередніх відношень з вказівкою передбачуваного первинного ключа для кожного відношення, використовуючи діаграми ER-типу.

На основі аналізу діаграми ER-типу за допомогою шести сформульованих вище правил одержуємо набір попередніх відношень, представлених наступними схемами відношень.

Зв'язок *МАЄ* задовільняє умовам правила 4, відповідно до якого одержуємо два відношення:

1.ВИКЛАДАЧ (**ПІБ, Стаж**....) - додався ключовий атрибут **Стаж**.

2.СТАЖ (Стаж....).

Зв'язок *ВЕДЕ* задовільняє умовам правила 6, відповідно до якого одержуємо три відношення:

1.ВИКЛАДАЧ (**ПІБ, Стаж**. ...).

2.ЗАНЯТТЯ (Група. Предм. ...)

3.ВЕДЕ **(ФІО. Група. Предм. ...)**

Зв'язок ЗАЙМАЄ(Посідає) аналогічно зв'язку МАЄ задовільняє умовам правила 4, тому маємо наступні два відношення

1.ВИКЛАДАЧ (**ПІБ. Стаж, Пос**....) додався ключовий атрибут **Пос.**

2.ПОСАДА (**Пос**. ...).

Третій етап проектування - додавання не ключових атрибутів, які не були вибрані як ключові раніше, і призначення їх одному з попередніх відношень з тією умовою, щоб відношення відповідали вимогам нормальної форми Бойса-Кодда.

Після додавання не ключових атрибутів схеми відношень приймуть наступний вигляд:

ВИКЛАДАЧ (**ПІБ, Стаж. Пос**. Каф)

СТАЖ (**Стаж**, Д_Стаж)

ЗАНЯТТЯ (**Група. Предм**).

ВЕДЕ (**ПІБ, Група, Предм**. ВидЗан).

ПОСАДА (**Пос**, Оклад).

Після визначення відношеньслід перевірити їх на відповідність вимогам нормальної форми Бойса-Кодда. В нашому випадку ми отримали ті ж відношення, що і при проектуванні прикладу методом нормальних форм.

Отримана схема бази даних приведена на рисунку.

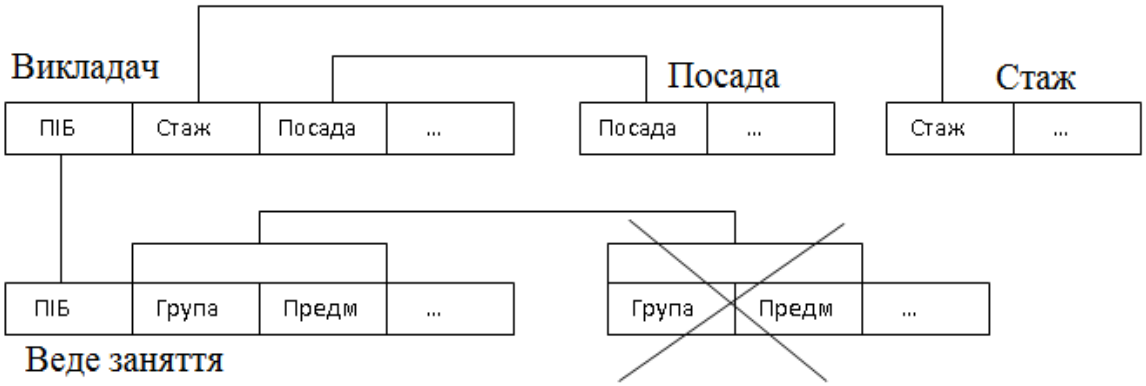


Рисунок 16.18 – Схема бази даних

Зауваження:

В нашому прикладі відношення ЗАНЯТТЯ, окрім ключових атрибутів (**Група.**Предм). інших атрибутів не має. Тому воно не несе додаткової інформації, окрім тієї, що міститься у відношенні ВЕДЕ. Дійсно, це відношення включає атрибути Група. предм. Тому відношення ЗАНЯТТЯ потрібно виключити з сформованої схеми БД (на малюнку воно є закресленим).

Засоби автоматизації проектування

Далі дається загальна характеристика CASE-засобів створення і супроводу інформаційних систем. Розглядаються моделі життєвого циклу програмних систем, описуються моделі структурного і об'єктно-орієнтованого проектування. Дається класифікація CASE-засобів і приводиться характеристика найпопулярніших систем.

Основні визначення

Для автоматизації проектування і розробки інформаційних систем раніше широко застосовувалася **структурна** методологія, що означає використання формалізованих методів опису системи і схвалюваних технічних рішень, що розробляються. При цьому використовувалися графічні засоби опису різних моделей інформаційних систем за допомогою схем і діаграм. При ручній розробці інформаційних систем такі графічні моделі розробляти і використовувати вельми трудомістко.

Відзначені обставини послужили однією з причин появи програмно-технологічних засобів, що отримали назву CASE-засобів і реалізовуючих CASE-технологію створення і супроводу інформаційних систем. Окрім структурної методології, у ряді сучасних CASE-засобів використовується об'єктно-орієнтована методологія проектування.

Термін CASE (Computer Aided Software Engineering) дослівно переводиться як розробка програмного забезпечення за допомогою комп'ютера. В даний час цей термін отримав більш широке значення, що означає автоматизацію розробки інформаційних систем.

CASE-засоби є програмними засобами, що підтримують процес створення або супроводу інформаційних систем, такі як: аналіз і формування вимог, проектування баз даних і додатків, генерація коду, тестування, забезпечення якості, управління конфігурацією і проектом.

CASE-систему можна визначити як набір CASE-засобів, що мають певне функціональне призначення і виконаних в рамках єдиного програмного продукту.

CASE-технологія звичайно визначається як методологія проектування інформаційних систем плюс інструментальні засоби, що дозволяють наочно моделювати наочну область, аналізувати її модель на всіх етапах розробки і супроводу інформаційної системи і розробляти додатки для користувачів.

CASE-індустрія об'єднує сотні фірм і компаній різної орієнтації. Практично всі серйозні зарубіжні програмні проекти здійснюються з використанням CASE-засобів, а загальне число поширюваних пакетів перевищує 500 найменувань.

Основна мета CASE-систем і засобів полягає в тому, щоб відділити проектування програмного забезпечення від його кодування і подальших етапів розробки (тестування, документування і ін.), а також автоматизувати весь процес створення програмних систем, або **інженерінг** (від англ, engineering - розробка).

Останнім часом все частіше розробка програм починається з деякого попереднього варіанту системи. Як такий варіант може виступати спеціально для цього розроблений **прототип**, або застаріла і така, що не задовільняєновим вимогам система. В останньому випадку для відновлення знань про програмну систему з метою подальшого їх використання застосовують повторну розробку **- реінженерінг.**

Повторна розробка зводиться до побудови початкової моделі програмної системи шляхом дослідження її програмних кодів. Маючи модель, можна її удосконалити, а потім знов перейти до розробки. Так часто і чинять при проектуванні. Одним з самих відомих принципів такого типу є принцип «поворотного проектування» - Round Trip Engineering (RTE).

Сучасні CASE-системи не обмежуються тільки розробкою, а частіше за все забезпечують і повторну розробку. Це істотно прискорює розробку додатків і підвищує їх якість.

Перспективна CASE-система

Динаміка зміни позицій структурних і об'єктно-орієнтованих систем дозволяє припустити, що перспективна CASE-система буде об'єктно-орієнтованою. Розглянемо вимоги до ідеальної перспективної CASE-системи.

Повнофункціональна об'єктно-орієнтована система повинна вирішувати задачі аналізу і моделювання, проектування, розробки (реалізації), а також мати ефективну інфраструктуру, що забезпечує сервісом перші три основні задачі.

Для вирішення задач аналізу і моделювання доцільно мати інтегроване середовище розробника, яке повинне забезпечувати можливості:

- динамічного моделювання подій в системі;
- динамічної і злагодженої корекції всієї сукупності діаграм;
- додавання написів пояснень до діаграм моделей і в документацію;
- автоматичної генерації документації;
- створення різних уявлень і приховання невживаних шарів системи;
- підтримка різних нотацій (ця вимога слабшає у зв'язку з переходом до єдиної мови моделювання).

Здійснення процесу проектування припускає наявність можливостей:

- визначення основної моделі прикладної задачі (бізнес-моделі, звичайно об’єктно-орієнтованої) і правил її поведінки (бізнес-правил);

• підтримки процесу проектування за допомогою бібліотек, оснащених засобами зберігання, пошуку і вибору елементів проектування (об'єктів і правил);

• створення призначеного для користувача інтерфейсу і підтримка поширених програмних інтерфейсів (підтримка стандартів OLE, OpenDoc, доступ до бібліотек HTML/Javai т. п.);

• створення різних розподілених клієнт-серверних додатків.

Засоби розробки у складі CASE-системи повинні забезпечувати побудову додатків за наслідками попередніх етапів розробки додатку. Максимально високою вимогою до засобів розробки можна рахувати генерацію готової до виконання програми.

CASE-системи найближчогомайбутнього повинні дозволяти створювати скелетні заготівки під класи, атрибути і методи, готові додатки на об'єктно-орієнтованих мовах програмування типу C++ або Smalltalk, або програми на мовах клієнт-серверних продуктів (PowerBuilder, Forte, VisualAge, VisualWorks і т. д.). До підтримуваних мов, мабуть, скоро додасться мова Java.

Для забезпечення зв'язку з іншими етапами життєвого циклу додатку засобу CASE-системи повинні включати і функції контролю програмного коду на синтаксичну коректність і відповідність моделям.

Ядром інфраструктури майбутніх CASE-систем повинен бути репозиторій, що відповідає за генерацію коду, реінженерінг і забезпечуючий відповідність між моделями і програмними кодами. Основу репозиторія складуть об'єктно-орієнтовані БД, хоча раніше для цього використовувалися реляційні БД. Іншими найважливішими функціями інфраструктури є функції контролю версій і управління частинами системи при колективній розробці.

Контрольні запитання:

1. Назвіть етапи проектування баз даних.
2. На чому ґрунтуються правила формування відношень?
3. Напишіть правила формування відношень для зв’язку 1:1 (правила 1, 2 та 3).
4. Напишіть правила формування відношень для зв’язку 1:M або M:1 (правила 4 та 5).
5. Напишіть правила формування відношень для зв’язку M:M (правило 6)

6. Дайте визначення термінам CASE-засоби, CASE-система та CASE-технологія.

7. Перелічіть можливості, які повинно забезпечувати інтегроване середовище?

8. Здійснення процесу проектування припускає наявність можливостей. Назвіть їх.

Тестові завдання:

1. Якщо ступінь зв'язку між суттєвостями 1:M (або M:1) і клас приналежності M-зв'язкової суттєвості обов'язковий, то достатньо формування двох відношень (по одному на кожну з суттєвостей). При цьому первинними ключами цих відношень є ключі їх суттєвостей. Крім того, ключ 1-зв'язної суттєвості додається як атрибут (зовнішній ключ) у відношення, відповідної M-зв'язкової суттєвості, це :

а) Правило 5

б) Правило 4

в) Правило 1

г) правильної відповіді немає

2. Якщо ступінь бінарного зв'язку 1:1 і клас належності обох суттєвостей обов'язковий, то формується одне відношення. Первинним ключем цього відношення може бути ключ будь-якої з двох суттєвостей, це правило:

а) Формування відношень для зв'язку M:M

б) Формування відношень для зв'язку 1:M

в) Формування відношень для зв'язку 1:1

г) правильної відповіді немає

3. Основна метаCASE-систем і засобів це:

а) усунення порожніх полів та дублювання

б) формування відношень на основі ER-діаграм

в) відділити проектування програмного забезпечення від його кодування і подальших етапів розробки

г) правильної відповіді немає

4. Набір CASE-засобів, що мають певне функціональне призначення і виконаних в рамках єдиного програмного продукту – це:

а) CASE-індустрія
б) CASE-технологія
в) CASE-система
г) CASE-відношення

5. *Скільки всього є правил формувань відношень?*
а) 10 правил
б) 6 правил
в) 3 правила
г) правильної відповіді немає

Розділ 17

Приклад побудови ER-моделі

- **Опис ПО**
- **Початкова контекстна діаграма**
- **Таблиця – список подій**
- **Таблиця – розподіл потоків**
- **Повна контекстна діаграма**
- **Діаграма структур даних**
- **Процес побудови модульної моделі**

Опис ПО

У цьому прикладі використовують методологію Yourdon, реалізовану в CASE- засобі VantageTeamBuilder.

Як ПО використовують опис роботи відео бібліотеки, яка одержує запити від клієнтів на фільми, котрі вони повертають. Запити розглядає адміністрація відео бібліотеки, використовуючи інформацію про клієнтів і фільми. При цьому перевіряють і поновлюють список орендованих фільмів, а також записи про членство в бібліотеці. Якщо необхідний фільм наявний у бібліотеці, адміністрація інформує клієнта про орендну плату. Але, якщо клієнт невчасно повернув фільми, йому не дозволяється брати нові. Коли стрічку повертають, розраховує орендну плату (плюс пеня за несвоєчасне повернення). Якщо клієнт не є членом бібліотеки, він не має права на оренду.

Відео бібліотека одержує нові фільми від своїх постачальників. Коли нові фільми надходять у бібліотеку, це фіксується. Інформація про членство у бібліотеці зберігається окремо від записів про оренду фільмів.

Адміністрація бібліотеки регулярно складає звіти за певний період часу про членів бібліотеки, нові надходження, оренду фільмів, їх постачальників.

Організація проекту. Увесь проект поділено на 4 фази: аналіз, глобальне проектування (проектування архітектури системи), детальне проектування і реалізація (програмування).

На фазі аналізу будується модель середовища, тобто проводиться:

- Аналіз поведінки системи (визначається призначення системи ІС, побудова початкової контекстної діаграми потоків даних

(DFD) і формування матриці списку подій (ELM – Event List Matrix), побудова контекстних діаграм;

- Аналіз даних (визначення складу потоків даних і побудова діаграм структур даних (DSD – Data Structure Diagrames), конструювання глобальної моделі даних у вигляді ER-діаграми;

У цьому разі призначення ІС формується в такий спосіб: ведення БД про членів бібліотеки, постачальників, фільми та оренду. При цьому керівництво бібліотеки повинно мати можливість одержувати різні види звітів для виконання своїх завдань.

Перед побудовою контекстної діаграми DFD необхідно проаналізувати зовнішні події (зовнішні об’єкти), що впливають на функціонування бібліотеки.

Ці об’єкти взаємодіють із ІС шляхом інформаційного обміну.

З опису ПО випливає, що в процесі роботи бібліотек беруть участь такі групи людей: клієнти, постачальники і керівництво. Ці групи є зовнішніми об’єктами. Вони не тільки взаємодіють із системою, а також визначають її межі і зображують на початковій контекстній діаграмі DFD як термінатори (зовнішні сутності).

Початкову контекстну діаграму зображено на рисунку 17.1. Зовнішні сутності позначено звичайними прямокутниками, а процеси – колом.

Рисунок 17.1 - Початкова контекстна діаграма

Список подій будується у вигляді матриці (ELM) і описує різні дії зовнішніх сутностей та реакцію ІС на ці дії. Вони являють собою зовнішні події, що впливають на бібліотеку.

Типи подій:

NC – нормальне керування;

ND – нормальні дані;

NCD – нормальне керування/дані;

TC – тимчасове керування;

TD – тимчасові дані;

TCD – тимчасове керування/дані.

Усі дії позначаються як нормальні дані. Ці дані є подіями, які ІС сприймає безпосередньо, наприклад, зміна адреси клієнта, що має бути відразу зареєстровано. Вони з’являються у DFD як вміст потоків даних.

Список подій наведено у таблиці.

Таблиця 17.1 – Список подій

Опис події	Тип	Реакція
1. Клієнт бажає стати членом бібліотеки	ND	Реакція клієнта як члена бібліотеки
2. Клієнт повідомляє про зміну адреси	ND	Реакція нової адреси клієнта
3. Клієнт замовляє фільм	ND	Розгляд запиту
4. Клієнт повертає фільм	ND	Реакція повернення
5. Керівництво надає повноваження новому постачальнику	ND	Реакція постачальника
6. Постачальник повідомляє про зміну адреси	ND	Реакція нової адреси постачальника
7. Постачальник направляє фільм у бібліотеку	ND	Одержання нового фільму
8. Керівництво запитує новий фільм	ND	Формування необхідного звіту для керівництва

Для завершення аналізу функціонального аспекту поведінки системи будується повна контекстна діаграма, яка містить діаграму нульового рівня. При цьому **Бібліотека** поділяється на 4 процеси, які відображають основні види адміністративної діяльності бібліотек. Існуючі «абстрактні» потоки даних між термінаторами і процесами трансформуються у потоки, які представляють обмін даними на більш контекстному рівні. Список подій показує, які потоки існують на цьому рівні: кожна подія із списку має формувати деякий потік (подія формує вхідний потік, реакція – вихідний потік).

Один «абстрактний» потік може бути поділений на більше ніж один «конкретний» потік. Розглянемо розподіл потоків.

Таблиця 17.2 – Розподіл потоків

Потоки на діаграмі верхнього рівня	**Потоки на діаграмі нульового рівня**
Інформація про клієнта	Дані про клієнта, Запит про оренду
Інформація для клієнта	Членська карта, Відповідь на запит про оренду
Інформація від керівництва	Запит звіту про нових членів, Новий постачальник, Запит звіту про постачальників, Запит звіту про оренду, Запит звіту про фільми
Інформація для керівництва	Звіт про нових членів, Звіт про постачальників, Звіт про оренду, Звіт про фільми
Інформація від постачальника	Дані про постачальника, Нові фільми

На наведеній повній контекстній діаграмі накопичувач даних **бібліотека** є глобальним або абстрактним поданням сховища даних.

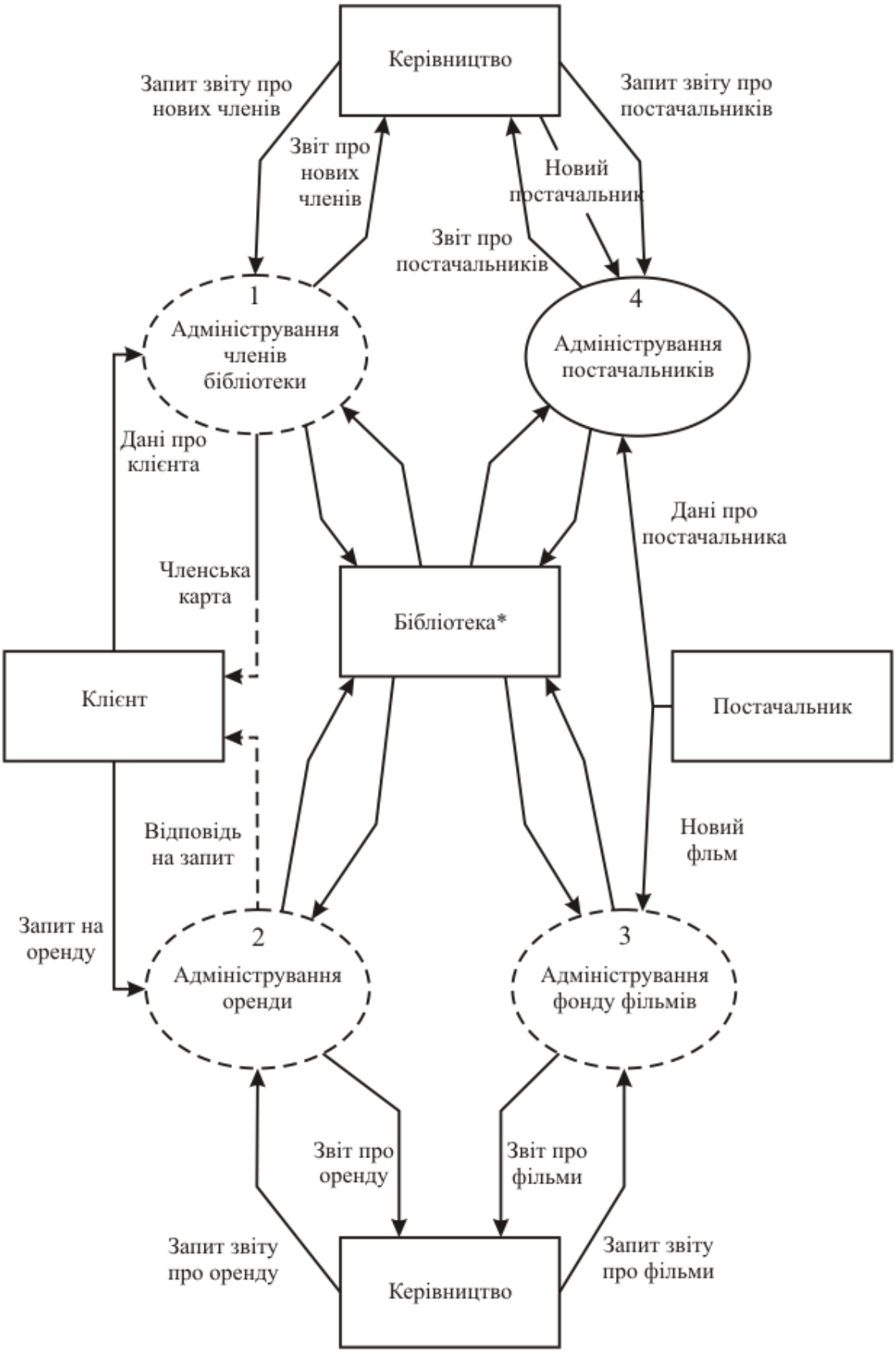


Рисунок 17.2 - Повна контекстна діаграма

Аналіз функціонального аспекту поведінки системи дає уявлення про обмін і перетворення даних у системі. Взаємозв’язок між

«абстрактними» і «конкретними» потоками даних на діаграмі нульового рівня виражається у діаграмах структур даних.

Рисунок 17.3 - Діаграма структур даних

На фазі аналізу будується глобальна модель даних, яка подається у вигляді діаграми «сутність-зв'язок».

Між різними типами діаграм існують такі взаємозв’язки:

ELM – DFD: події – вхідні потоки, реакції – вихідні потоки;

DFD – DSD: потоки даних – структури даних верхнього рівня;

DFD – ERS: накопичувачі даних – ER- діаграми;

DSD – ERD: структури даних нижнього рівня – атрибути сутностей.

На етапі проектування архітектури будується предметна модель. Процес побудови предметної області містить в собі:

- Детальний опис функціонування системи;
- Подальший аналіз використовуваних даних і побудова концептуальної моделі даних для наступного проектування БД;
- Визначення структури інтерфейсу користувача, специфікації форм і порядку їх появи;
- Уточнення діаграм потоків даних і списку подій, виділення серед процесів нижнього рівня інтерактивних і не інтерактивних, визначення для них міні специфікацій.

Результати проектування архітектури такі:

- Модель процесів (діаграми архітектури SAD – System Architecture Diagram) та міні специфікації структурованою мовою;
- Модель даних (ERD та її підсхеми);
- Модель інтерфейсу користувача (класифікація процесів та інтерактивні та не інтерактивні функції, діаграма послідовності форм (FSD – Form Sequence Diagram). Модель показує, які форми з’являються у додатку і в якому порядку. На FSD фіксується набір і структура викликів екранних форм. Діаграми FSD утворюють ієрархію, на вершині якої знаходиться головна форма додатка, що реалізує підсистему. На другому рівні знаходяться форми, які реалізують процеси нижнього рівня функціональної структури, зафіксованої на діаграмах SAD.

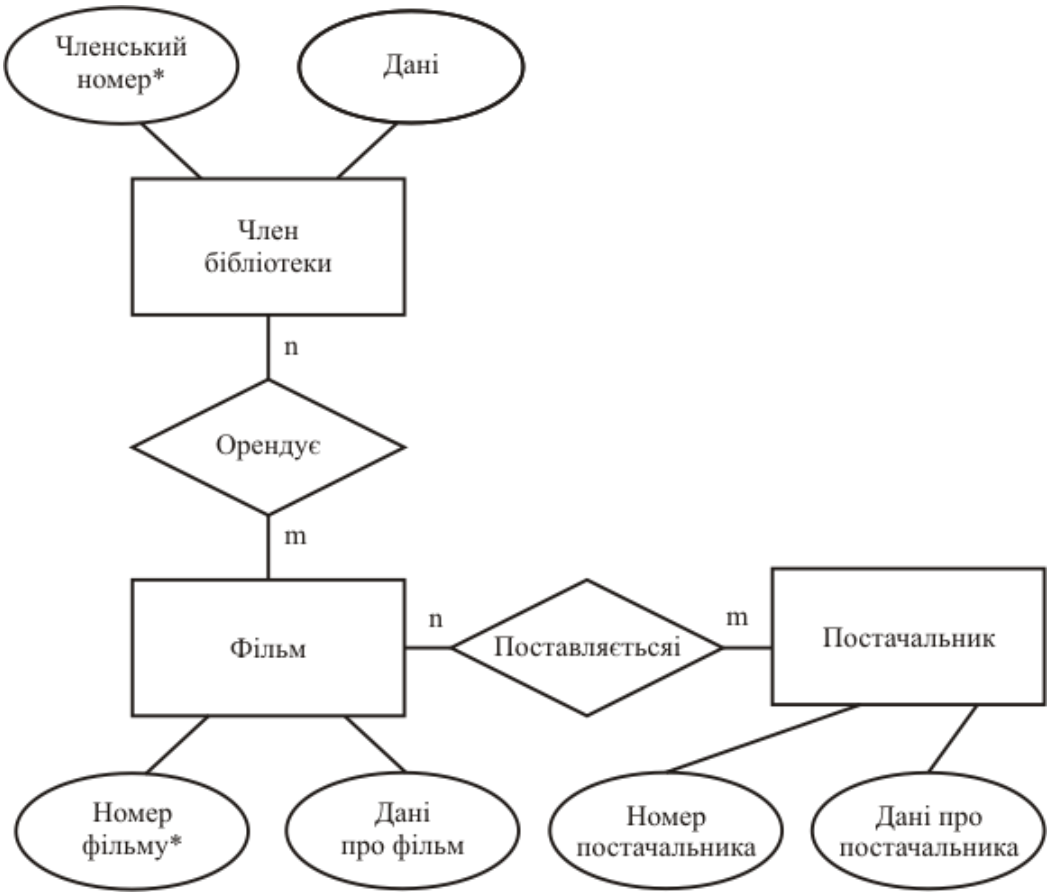


Рисунок 17.4 - Діаграма «сутність – зв'язок»

На фазі детального проектування будують модульну модель. Під **модульною моделлю** розуміють реальну модель прикладної системи, що проектується. **Процес її побудови поєднує у собі:**

- Уточнення моделі БД для наступної генерації SQL-пропозицій;
- Уточнення структури інтерфейсу користувача;
- Побудову структурних схем, які відображають логіку роботи інтерфейсу користувача і модель бізнес-логіки (SCD – Structure Charts Diagram), та прив’язку їх до форм.

Результатами детального проектування є:

- Модель процесів (структурні схеми інтерактивних і не інтерактивних функцій);
- Модель даних (визначення в ERD усіх необхідних параметрів для додатків);
- Модель інтерфейсу користувача (діаграма послідовності форм (FSD), яка показує, які форми з’являються у додатку та в якому, взаємозв’язок між кожною формою і визначеною структурною схемою,

взаємозв’язок між кожною формою й однією або більше сутностями в ERD).

На фазі реалізації будується реляційна модель. Процес її побудови такий:

- Генерація SQL- пропозицій, які визначають структуру цільової БД (таблиці, індекси, обмеження цілісності);
- Уточнення структурних схем (SCD) і діаграм послідовності форм (FSD) із наступною генерацією коду додатків.

На підставі аналізу потоків даних і взаємодії процесів зі сховищами даних здійснюється остаточне виділення підсистем (попереднє має бути зроблено та зафіксовано на етапі формулювання вимог у технічному завданні). Під час виділення підсистеми необхідно керуватися принципами функціональної зв’язаності і мінімізації інформаційної залежності. Слід урахувати, що на базі таких елементів підсистеми, які процеси і дані, на етапі розроблення має бути створений додаток, здатний функціонувати самостійно. Однак під час групування процесів і даних у підсистемі потрібно врахувати вимоги до конфігурування продукту, якщо вони були сформульовані на етапі аналізу.

Контрольні запитання:

1. На які фази було поділено проект?
2. Опишіть процес побудови модульної моделі.
3. Які результати детального проектування?
4. Наведіть типи подій, які були розглянуті у даному прикладі.
5. Що містить в собі процес побудови предметної області?

Тестові завдання:

1. На скільки фаз було поділено проект?

а) 5

б) 4

в) 8

г) правильної відповіді немає

2. Аналіз поведінки системи проводиться на фазі:

а) Аналізу

б) Детального проектування

в) Реалізації

г) правильної відповіді немає

3. DFD – ERS – це взаємозв'язок:
а) події – вхідні потоки, реакції – вихідні потоки
б) потоки даних – структури даних верхнього рівня
в) накопичувачі даних – ER- діаграми
г) правильної відповіді немає

4. Структури даних нижнього рівня – атрибути сутностей – це тип діаграми:
а) DSD – ERD
б) ELM – DFD
в) DFD – ERS
г) DFD – DSD

5. Один «абстрактний» потік може бути поділений на:
а) більше ніж один «конкретний» потік
б) тільки на 3 потоки
в) не може бути поділений зовсім
г) правильної відповіді немає

Розділ 18

Зберігання інформації у базах даних

- **Властивості реляційних СКБД**
- **Види об’єктів у зовнішній пам’яті БД**
- **Зберігання відношень**
- **Індекси**
- **Хешування**
- **Журнальна інформація**
- **Службова інформація**

Реляційні СКБД мають наступні властивості для впливу на організацію зовнішньої пам’яті:

1. Наявність двох рівнів системи: **рівня безпосереднього керування** даними у зовнішній пам’яті (а також, звичайно, керування буферами оперативної пам’яті, керування транзакціями і ведення журналу змін БД); **мовного рівня** (наприклад, рівня, що реалізує мова SQL). При такій організації підсистема нижнього рівня має підтримувати у зовнішній пам’яті набір базових структур, конкретна інтерпретація яких належить до функцій підсистеми верхнього рівня.

2. Підтримка відношень-каталогів. Інформація, пов’язана з найменуванням об’єктів БД і їхніми конкретними властивостями (наприклад, структура ключа індексу),підтримується підсистемою мовного рівня. Як відношення-каталог не відрізняється від звичайного відношення БД.

3. Регулярність структур даних. Оскільки основним об’єктом реляційної моделі даних є плоска таблиця, головний набір об’єктів зовнішньої пам’яті може мати дуже просту регулярну структуру. При цьому необхідно забезпечити ефективне виконання операторів мовного рівня як над одним відношенням (прості селекція і проекція), так і над декількома відношеннями (наприклад, сполучення декількох відношень). Для цього у зовнішній пам’яті мають підтримуватися додаткові керуючі структури – індекси.

4. Надмірність зберігання даних, яка необхідна для забезпечення надійного зберігання БД. Надмірність зазвичай реалізується у вигляді журналу змін БД.

Отже, розрізняють такі **види об’єктів у зовнішній пам’яті БД**:

- **Рядки відношень** – основна частина БД, більшою мірою безпосередньо доступна користувачам;
- **Керуючі структури** – індекси, створені з ініціативи користувача (адміністратора) або верхнього рівня системи з метою підвищення ефективності виконання запитів (як правило, автоматичні підтримуються нижнім рівнем системи);
- **Журнальна інформація**, яка підтримується для виконання внутрішніх потреб нижнього рівня системи (наприклад, інформація про вільну пам'ять).

Існують два альтернативні підходи до організації СКБД з точки зору розподілу функцій між різноманітними компонентами. Наприклад, у СКБД SystemR існувала інтегрована підсистема керування даними, транзакціями і журналами змін, а в СКБД Ingres – керування даними було відокремлено від керування транзакціями і журналами змін.

Зберігання відношень

Існують два принципові підходи до фізичного зберігання відношень. Найбільш розповсюдженим є покортежне зберігання відношень (кортеж з одиницею фізичного зберігання). Це забезпечує швидкий доступ до цілого кортежу, але при цьому у зовнішній пам’яті дублюються загальні значення різних кортежів одного відношення і виникають зайві обміни із нею, якщо потрібна частина кортежу.

Альтернативним (менш розповсюдженим) підходом є зберігання відношення стовпчика, тобто одиницею зберігання є стовпчик відношення з виключеними дублікатами. Зрозуміло, що за такої організації сумарно витрачається менше зовнішньої пам’яті, оскільки дублікати значень не зберігаються; за один обмін зовнішньої пам’яттю у загальному випадку можна одержати більше корисної інформації. Додатковою перевагою є можливість використання значень стовпчика відношення для оптимізації виконання операцій приєднання. Але при цьому потрібні істотні додаткові дії для збирання кортежу (або його частини).

Найпростішим рішенням є зберігання таких даних в окремих (поза БД) файлах із заміною «довгих» даних у кортежі на ім’я відповідного файлу. У деяких системах (наприклад, у одній із версій СКБД Informix) дані зберігалися в окремому наборі сторінок зовнішньої пам’яті, зв’язаному фізичними посиланнями. Обидва рішення дуже обмежують можливість роботи з «довгими» даними (наприклад, «Як усунути декілька байтів із середини двомегабайтного рядка?»). В інших

системах використовують засіб, запропонований декілька років тому у проекті Exodus, коли «довгі» дані організують у вигляді B-дерев послідовностей байтів.

Як правило, в одній сторінці даних зберігаються кортежі тільки одного відношення. Існують, однак, варіанти, коли можливе зберігання в одній сторінці кортежів декількох відношень. Це спричиняє деякі додаткові витрати щодо обробки службової інформації (при кожному кортежі потрібно зберігати інформацію про відповідне відношення), але інколи дозволяє різко скоротити кількість звернень до зовнішньої пам’яті під час виконання операції приєднання.

Розповсюдженим засобом підвищення ефективності СКБД є кластеризація відношення за значеннями одного або декількох стовпчиків. Корисною для оптимізації сполучень є спільна кластеризація декількох відношень.

З метою використання можливостей паралельного обміну із зовнішньою пам’яттю інколи застосовують схему декластеризованого зберігання відношень: кортежів із загальним значенням стовпчика декластиризації розміщують на різних дискових приладах, обмін з яким можна виконувати паралельно.

Що ж стосується зберігання відношень стовпчиками, то сутність полягає у спільному зберіганні всіх значень одного або декількох стовпчиків. Для кожного кортежу відношення зберігається кортеж одного і того самого ступеня, який складається з посилань на місця розташування відповідних значень стовпчиків. Одним з прийомів, що застосовують у розподілених системах БД, є так званий **вертикальний розподіл відношень**, коли у різних вузлах мережі зберігаються різні проекції даного відношення. Зберігання відношення стовпчика в деякому розумінні є крайнім випадком вертикального розподілу відношень.

Індекси

Основним призначенням індексів є забезпечення ефективного прямого доступу до кортежу відношення за ключем. Зазвичай індекс визначається для одного відношення, і ключем є значення атрибута (можливо, складного). Якщо ключем індексу можливий ключ відношення, то індекс має бути унікальним, тобто не містити дублікатів ключа. На практиці ситуація звичайно виглядає інакше: під час оголошення первинного ключа відношенню автоматично надається унікальний індекс, а єдиним засобом оголошення можливого ключа,

відмінного від первинного, є явне створення унікального індексу. Це пов’язано з тим, що для перевірки збереження унікальності можливого ключа так чи інакше потрібна індексна підтримка.

Оскільки під час виконання багатьох операцій мовного рівня потрібне сортування відношень відповідно до значень деяких атрибутів, корисною властивістю індексу є забезпечення послідовного перегляду кортежів відношення у діапазоні значень ключа, в порядку зростання або зменшення цих значень.

Нарешті, одним із засобів оптимізації виконання еквіпоєднання відношень є організація так званих **мультиіндексів** для декількох відношень, що мають загальні атрибути. Будь-який з цих атрибутів (або їх набір) може бути ключем мультиіндексу. Значенню ключа зіставляється набір кортежів усіх зв’язаних мультиіндексом відношень, значення виділених атрибутів яких зберігаються із значенням ключа.

Загальною ідеєю будь-якої організації індексу, що підтримує прямий доступ за ключем і послідовний перегляд у порядку зростання або зменшення його значень, є зберігання упорядкованого списку ідентифікаторів кортежів. Одна організація індексу відрізняється від іншої, переважно, засобом пошуку ключа із заданим значенням.

Хешування

Альтернативним і популярним підходом до організації індексів є використання техніки хешування. Загальною ідеєю засобів хешування є застосування до значення ключа деякої функції перетворення (хеш-функції), що виробляє значення меншого розміру. Отримане значення ключа після цього використовують для доступу до запису.

У найпростішому, класичному випадку згортку ключа застосовують як адресу в таблиці, що містить ключі та записи. Основною вимогою до хеш-функції є рівномірний розподіл значення перетворення. У разі виникнення колізій (одного і того самого результату для декількох значень ключа) утворяться ланцюжки переповнення. Головним обмеженням для цього засобу є фіксований розмір таблиці. Якщо таблиця занадто заповнена чи переповнена, то виникає дуже багато ланцюжків переповнення, і основну перевагу хешування – доступ до запису майже завжди за одне звернення до таблиці – буде втрачено. Розширення таблиці вимагає її повного перероблення на підставі нової хеш-функції (зі значенням функції більшого розміру).

Для БД такі дії є абсолютно неможливими. Тому зазвичай уводять проміжні таблиці-довідники, що містять значення ключів і адреси

записів, а самі записи зберігають окремо. Тоді у разі переповнення довідника слід його переробити, що потребує менш накладних витрат.

Для того щоб уникнути повного перероблення довідників, під час їх організації часто використовують техніку B-дерев розподілами та поєднаннями. Хеш-функція при цьому змінюється динамічно, залежно від глибини B-дерева. Шляхом додаткових технічних рішень можна зберегти порядок записів відповідно до значень ключа. У цілому, засоби B-дерев і хешування все більше стають подібними.

Журнальна інформація

Структура журналу, як правило, є суто індивідуальною для певної реалізації. Розглянемо тільки загальні властивості.

Журнал являє собою послідовний файл із записами змінного розміру, які можна переглядати у прямому або зворотному напрямку. Обмін здійснюється стандартними порціями (сторінками) з використанням буфера оперативної пам’яті. Структура (і тим більше, зміст) журнальних записів відома тільки компонентам СКБД, що відповідають за ведення журналів і їх відновлення. Оскільки вміст журналу є критичним під час відновлення БД після відмов, до ведення файлу журналу ставлять особливі вимоги надійності, наприклад, прагнуть підтримувати дві ідентичні копії журналу на різних пристроях зовнішньої пам’яті.

Службова інформація

Для конкретної роботи підсистеми керування даними у зовнішній пам’яті необхідно підтримувати інформацію, яка використовується тільки цією підсистемою і є недоступною для підсистеми мовного рівня. Набір структур службової інформації залежить від загальної організації системи, але звичайно потрібне підтримання службових даних, наведених нижче.

Внутрішні каталоги. Описують фізичні властивості об’єктів БД, наприклад, кількість атрибутів відношення, їх розмір і , можливо, типи даних; опис індексів, визначених для даного відношення тощо.

Описувачі вільної і зайнятої пам’яті у сторінках відношення. Така інформація потрібна для знаходження вільного місця під час занесення кортежів. По-різному розв’язують задачу пошуку вільного місця для некластеризованих і кластеризованих відношень (в останньому випадку потрібно додатково використати кластеризований індекс). Нетривіальною є проблема звільнення сторінки в умовах мультидоступу.

Зв'язування сторінок одного відношення. Якщо в одному файлі зовнішньої пам'яті можуть розташовуватися сторінки декількох відношень (як правило, цього прагнуть), то потрібно зв'язати сторінки одного відношення. Тривіальний засіб використання прямих посилань між сторінками часто призводить до виникнення синхронізації транзакцій (наприклад, важко звільнювати і створювати нові сторінки відношення). Тому намагаються використати додаткове зв'язування сторінок з використанням службових індексів, наприклад, загальний механізм для опису вільної пам'яті і зв'язування сторінок B-дерев.

Контрольні запитання:

1. Назвіть властивості, які має реляційна СКБД, для впливу на організацію зовнішньої пам'яті.
2. Назвіть та опишіть усі види об'єктів у зовнішній пам'яті БД.
3. Напишіть, що вам відомо, про зберігання відношень.
4. Яке основне призначення індексів?
5. Що таке хешування?
6. Які загальні властивості журнальної інформації?
7. Дайте визначення терміну внутрішні каталоги. Та напишіть про описувачі вільної і зайнятої пам'яті у сторінках відношень та зв'язування сторінок одного відношення.

Тестові завдання:

1. Скільки є рівнів системи?

а) Два
б) Три
в) Чотири
г) Правильної відповіді немає

2. Які є види об'єктів у зовнішній пам'яті БД?

а) Рядки відношень, журнальна інформація, керуючі структури
б) Керуючі структури, журнальна інформація, службова інформація
в) Журнальна інформація, службова інформація
г) Правильної відповіді немає

3. Описують фізичні властивості об'єктів БД, наприклад, кількість атрибутів відношення, їх розмір і , можливо, типи даних; опис індексів, визначених для даного відношення тощо, це:
а) Внутрішні каталоги
б) Описувачі вільної і зайнятої пам'яті у сторінках відношення
в) Зв'язування сторінок одного відношення
г) Правильної відповіді немає

4. Індекси, створені з ініціативи користувача (адміністратора) або верхнього рівня системи з метою підвищення ефективності виконання запитів, це:
а) Рядки відношень
б) Керуючі структури
в) Журнальна інформація
г) Правильної відповіді немає

5. *Основна частина БД, більшою мірою безпосередньо доступна користувачам, це:*
а) Рядки відношень
б) Керуючі структури
в) Журнальна інформація
г) Правильної відповіді немає

Розділ 19

Індексація даних

- **Індексація**
- **Однорівнева схема індексації**
- **Дворівнева схема індексації**
- **B-дерева**

Як наголошувалося вище, визначення ключа для таблиці означає автоматичне сортування записів, контроль відсутності повторень значень в ключових полях записів і підвищення швидкості виконання операцій пошуку в таблиці. Для реалізації цих функцій в СКБД застосовують **індексацію.** Термін «індекс» тісно пов'язаний з поняттям «ключ», хоча між ними є і деяка відмінність. **Під індексом** розуміють засіб **прискорення** операції пошуку записів в таблиці, а отже, і інших операцій, що використовують пошук: витягання, модифікація, сортування і т.д. Таблицю, для якої використовується індекс, називають індексованою.

Індекс виконує роль **змісту** таблиці, перегляд якого передує зверненню до записів таблиці. В деяких системах, наприклад Paradox, індекси зберігаються в індексних файлах, збережених окремо від табличних файлів.

Варіанти рішення проблеми організації фізичного доступу до інформації залежать в основному від наступних чинників:

• виду вмісту в полі ключа записів індексного файлу;

• типу посилань (покажчиків), на запис основної таблиці;

• методу пошуку потрібних записів.

В полі **ключа індексного файлу** можна зберігати значення ключових полів таблиці, що індексується, або згортку ключа (так званий хеш-код). Перевага зберігання хеш-кода замість значення полягає в тому, що довжина згортки незалежно від довжини початкового значення ключового поля завжди має деяку постійну і достатньо малу величину (наприклад, 4 байти), що істотно знижує час пошукових операцій. Недоліком хешування є необхідність виконання операції згортки (вимагає певного часу), а також боротьба з виникненням колізій (згортка різних значень може дати однаковий хеш-код).

Для організації **посилання** на запис таблиці можуть використовуватися три типи адрес: абсолютна (дійсна), відносна і

символічна (ідентифікатор). На практиці частіше за все використовуються **два методи пошуку:** послідовний і бінарний (заснований на розподілі інтервалу пошуку навпіл).

Проілюструємо організацію індексації таблиць двома схемами: однорівневої і дворівневої. При цьому приймемо ряд припущень, які використовуються в сучасних обчислювальних системах. Нехай ОС підтримує пряму організацію даних на магнітних дисках, основні таблиці і індексні файли зберігаються в окремих файлах. Інформація файлів зберігається у вигляді сукупності блоків фіксованого розміру, наприклад, цілого числа кластерів.

При однорівневій схемі індексному файлі зберігаються короткі записи, що мають два поля: поле вмісту старшого ключа (хеш-кода ключа) блоку, що адресується, і поле адреси початку цього блоку. В кожному блоці запису розташовуються в порядку зростання значення ключа або згортки. Старшим ключем кожного блоку є ключ його останнього запису.

Якщо в індексному файлі зберігаються хеш-коди ключових полів індексованої таблиці, то алгоритм пошуку потрібного запису (з вказаним ключем) в таблиці включає наступні три етапи.

1.Побудови згортки значення ключового поля потрібного запису.

2.Пошук в індексному файлі запису про блок, значення першого поля якого більше отриманої згортки (це гарантує знаходження шуканої згортки в цьому блоці).

3.Послідовний перегляд записів блоку до збігу згорток потрібного запису і запису блоку файлу. У разі колізій згорток шукається запис, значення ключа якого співпадає із значенням ключа потрібного запису.

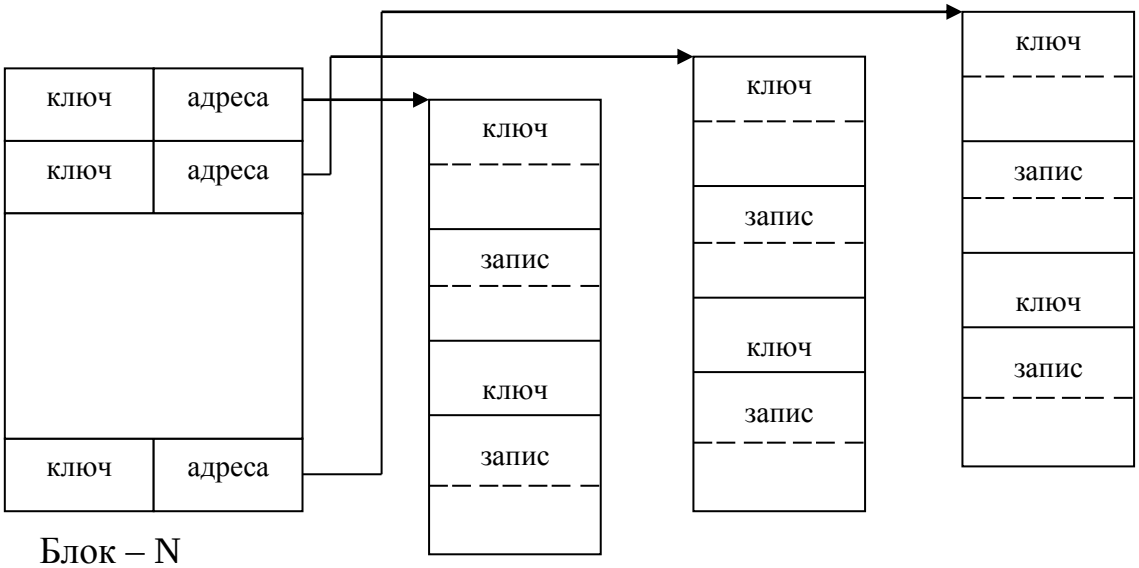


Рисунок 19.1 – Одноріvнева схема індексації

Основним **недоліком** одноріvневої схеми є те, що ключі (згортки) записів зберігаються разом із записами. Це приводить до збільшення часу пошуку записів через велику довжину перегляду (значення даних в записах доводиться пропускати).

Дворівнева схемау ряді випадків виявляється більш раціональною, в ній ключі (згортки) записів відокремлені від вмісту записів. В цій схемі індекс основної таблиці розподілений по сукупності файлів: одному файлу головного індексу і множини файлів з блоками ключів.

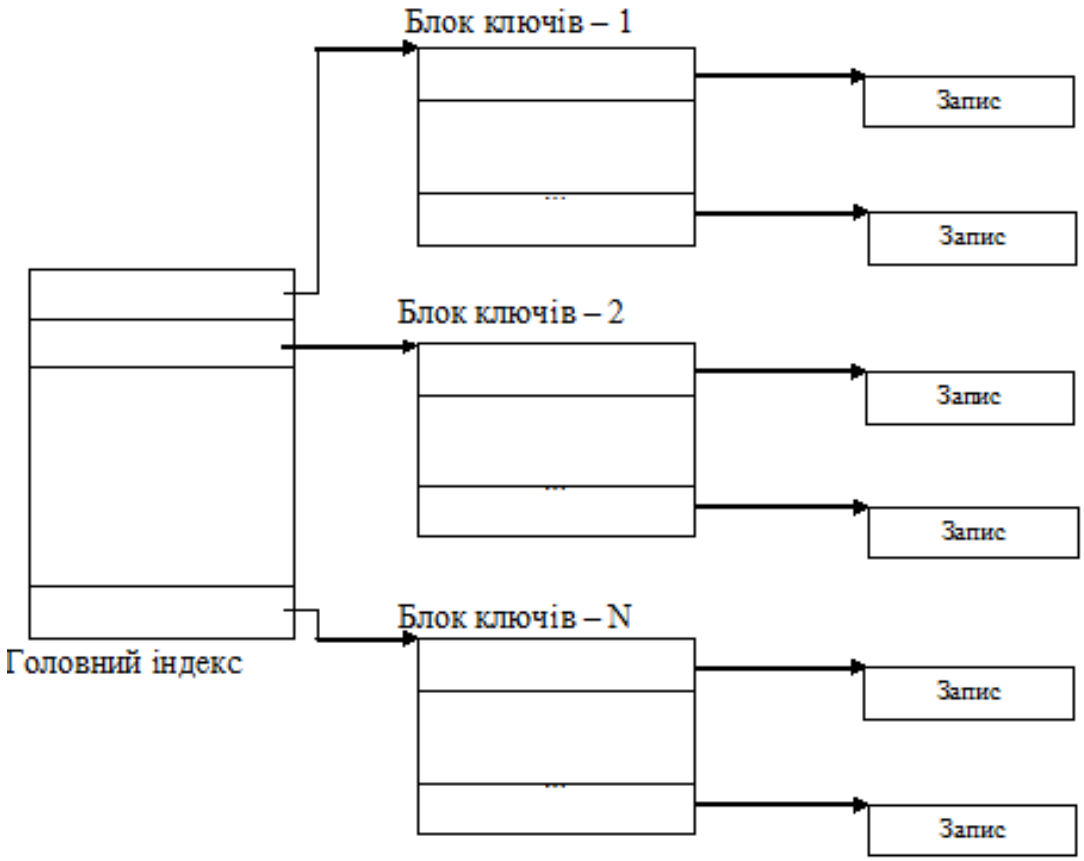


Рисунок 19.2 – Дворівнева схема індексації

На практиці для створення індексу для деякої таблиці БД користувач указує поле таблиці, яке вимагає індексації. Ключові поля таблиці в багатьох СКБД як правило індексуються автоматично. Індексні файли, що створюються за ключовими полями таблиці, часто називаються файлами **первинних індексів.** Індекси, створювані користувачем для не ключових полів, іноді називають **вторинними (призначеними для користувача) індексами.**Введення таких індексів не змінює фізичного розташування записів таблиці, проте впливає на послідовність перегляду записів. Індексні файли, що створюються для підтримки вторинних індексів таблиці, звичайно називаються **файлами вторинних індексів.**

Зв'язок **вторинного** індексу з елементами даних бази може бути встановлений різними способами. Один з них – використання **вторинного** індексу як входу для отримання первинного ключа, по якому потім з використанням первинного індексу проводиться пошук необхідних записів.

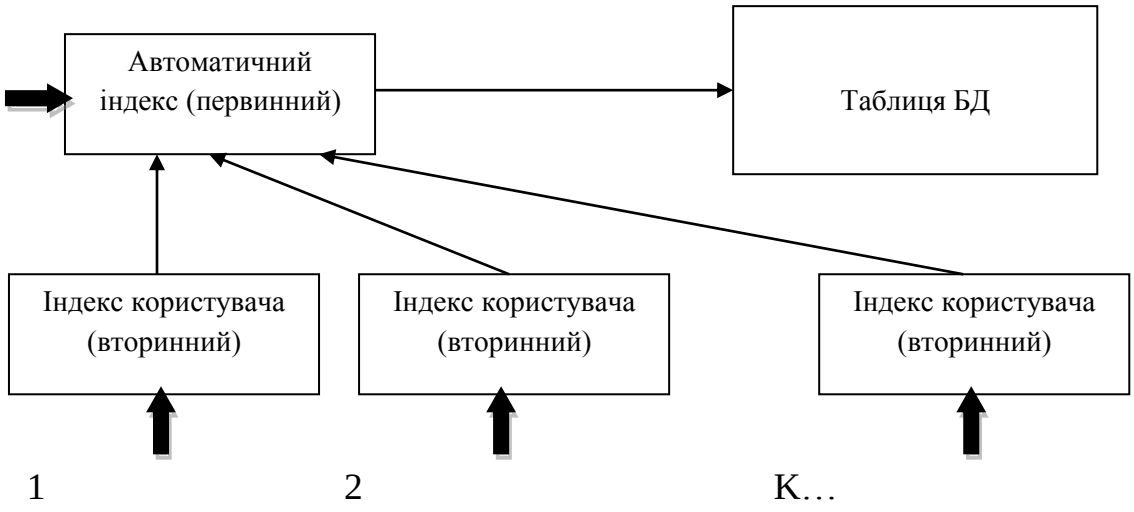


Рисунок 19.3 – Спосіб використання вторинних індексів

Деякими СКБД, наприклад Access, розподіл індексів на первинні і вторинні не проводиться. В цьому випадку використовуються автоматично створювані індекси і індекси, які визначаються користувачем по будь-якому з не ключових полів.

Головна причина підвищення швидкості виконання різних операцій в індексованих таблицях полягає в тому, що основна частина роботи проводиться з невеликими індексними файлами, а не з самими таблицями. Найбільший ефект підвищення продуктивності роботи з індексованими таблицями досягається для значних за об'ємом таблиць. Індексація вимагає невеликого додаткового місця на диску і незначних витрат процесора на зміну індексів в процесі роботи. Індекси в загальному випадку можуть змінюватися перед виконанням запитів до БД, після виконання запитів до БД, по спеціальних командах користувача або програмних викликах додатків.

B-дерева

Найбільш популярним підходом до організації індексів у БД є використання техніки B-дерев. B-дерево – це збалансоване, дуже розгалужене дерево у зовнішній пам’яті. Збалансованість означає, що довжина шляху від кореня дерева до будь-якого листка однакова. Розгалуженість дерева – це властивість кожного вузла дерева посилати на велику кількість вузлів-нащадків. З точки зору фізичної організації, B-дерево подається як мультиплексова структура сторінок зовнішньої пам’яті, тобто кожному вузлу дерева відповідає блок зовнішньої

пам’яті (сторінка). Внутрішні і листові сторінки зазвичай мають різну структуру.

Як правило, структура внутрішньої сторінки має такий вигляд:

N1 ключ (1) N2 ключ (2) N3 …… Nn ключ (n) N(n+1) ключ (n+1)

При цьому додержуються таких властивостей:

ключ (1)<= ключ (2)<= … ключ (n);

на сторінці дерева Nm знаходяться ключі k зі значеннями ключ (m) <= k<= ключ (m+1).

Листова сторінка має таку структуру:

Ключ (1) сп (1) Ключ (2) сп (2) …… ключ (t) сп (t)

Листова сторінка має такі властивості:

- ключ (1) <ключ (2)< …<ключ (t);
- Сп (r) – упорядкований список ідентифікаторів кортежів (tid), які містять значення ключа (r);
- листові сторінки зв’язані одно- або двоспрямованим списком.

Пошук у В-дереві це проходження від кореня до листка згідно із заданим значенням ключа. Зазначимо, що оскільки дерева є дуже розгалуженими і збалансованими, то для виконання пошуку за будь-яким значенням ключа потрібне одне і те саме (як правило, невелике) число обміну із зовнішньою пам’яттю. Більш точно, у збалансованому дереві, де довжини всіх шляхів від кореня до листка одні і ті самі, якщо у внутрішній сторінці зберігається n ключів, то при зберіганні m записів дерево глибиною $\log_n$ (m), де $\log_n$ обчислює логарифм на підставі n. Якщо n достатньо велике (звичайний випадок), то глибина дерева невелика, і пошук виконується швидко.

Основною властивістю В-дерев є автоматична підтримка їх збалансованості. Розглянемо процес виконання операцій занесення і вилучення записів.

Під час занесення нового запису виконуються:

Пошук листової сторінки. Фактично здійснюється звичайний пошук за ключем. Якщо у В-дереві не міститься ключ із заданим значенням, то буде отриманий номер сторінки, в якій він має міститися, та відповідні координати всередині сторінки.

Природно, що вся робота виконується у буферах оперативної пам’яті. Листова сторінка, в яку потрібно занести запис,

запам’ятовується у буфері, де і виконується операція вставлення. Розмір буфера має перевищувати розмір сторінки зовнішньої пам’яті.

Якщо після встановлення нового запису розмір частини буфера, що використовується, не перебільшує розмір сторінки, то на цьому виконання операції занесення запису закінчується. Буфер може бути негайно записаний у зовнішню пам'ять або тимчасово збережений в оперативній пам’яті, залежно від алгоритму керування буферами.

Якщо ж буфер переповнюється (тобто його розмір перевищує розмір сторінки), то виконується розподіл сторінки. Для цього організовується нова сторінка зовнішньої пам’яті, частина буфера ділиться навпіл (так, щоб друга половина також починалася з ключа), і друга половина записується у знов виділену сторінку, а у попередній сторінці модифікується значення розміру вільної пам’яті. Звичайно, модифікуються і посилання за списком листових сторінок.

Для того щоб забезпечити доступ від кореня дерева до знов створеної сторінки, необхідно відповідно модифікувати внутрішню сторінку, яка є предком листової сторінки, тобто вставити у неї відповідне значення ключа і посилання на нову сторінку. Під час виконання цієї операції може знову відбутися переповнення вже внутрішньої сторінки, і її буде поділено на дві. У результаті необхідно вставити значення ключа і посилання на нову сторінку у внутрішню сторінку-предка вище за ієрархією і т.д.

Крайнім випадком є переповнення кореневої сторінки B-дерева. У цьому разі вона теж розподіляється на дві, і створюється нова коренева сторінка дерева, тобто його глибина збільшується на одиницю.

Під час **вилучення запису** виконують такі дії:

- пошук запису за ключем. Якщо запис не знайдено, то вилучати нічого не потрібно;
- реальне вилучення запису в буфері, до якого прочитана відповідна листова сторінка. Якщо після виконання цієї операції розмір зайнятої у буфері області виявляється таким, що його сума з розміром зайнятої області у листових сторінках, розташованих ліворуч і праворуч від даної сторінки, більше, ніж розмір сторінки, операція завершується, інакше, здійснюється злиття сторінок, тобто у буфері створюється новий образ сторінки, що містить загальну інформацію з даної сторінки і її сусіда праворуч або ліворуч. Звільнена листова сторінка заноситься у список вільних сторінок. Відповідно коригується список листових сторінок.

Для того, щоб уникнути доступу від кореня до вільної сторінки, потрібно усунути відповідне значення ключа і посилання на вільну сторінку із внутрішньої сторінки – її предка. При цьому може виникнути потреба у злитті цієї сторінки з її сусідами ліворуч або праворуч.

Крайнім випадком є повне звільнення кореневої сторінки дерева, що можливо після злиття останніх двох нащадків кореня. Тоді коренева сторінка звільнюється, а глибина дерева зменшується на одиницю.

Як видно, під час виконання операцій вставки і вилучення властивість збалансованість B-дерева зберігається, а зовнішня пам'ять витрачається достатньо ощадливо.

Проблемою є те, що під час виконання операцій модифікації занадто часто можуть виникати розподілення і злиття. Щоб досягти ефективного використання зовнішньої пам’яті з мінімізацією розподілів і об’єднань, застосовують більш складні прийоми, а саме:

- попереджувальний розподіл, тобто розподіл сторінки не у разі її переповнення, а дещо раніше, коли ступінь заповнення досягає деякого рівня;
- «переливання», тобто підтримка рівномірного заповнення сусідніх сторінок;
- злиття «3 в 2», тобто проходження двох листових сторінок на основі вмісту трьох сусідніх.

Слід зазначити, що під час організації мультидоступу до B-дерев, характерного у разі використання їх СКБД, потрібно вирішувати ряд складних проблем. Звичайне рішення, наприклад, - це монопольне захоплення B-дерева протягом виконання операції модифікації. Але існують і більш гнучкі рішення, розгляд яких виходить за межі цього курсу.

Контрольні запитання:

1. Для чого існує індексація? Що таке індекс?
2. Від яких чинників залежить рішення проблеми організації фізичного доступу до інформації?
3. Напишіть етапи алгоритму пошуку потрібного запису (з вказаним ключем), якщо в індексному файлі зберігаються хеш-коди ключових полів індексованої таблиці.
4. Намалюйте однорівневу схему індексації.
5. Намалюйте дворівневу схему індексації.

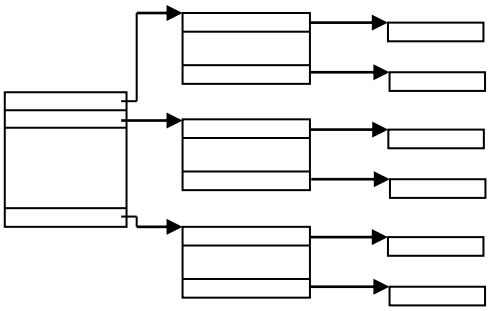
6. Що таке первинні і вторинні індекси?

7. Дайте визначення терміну В-дерево.

8. Що треба зробити, щоб досягти ефективного використання зовнішньої пам’яті з мінімізацією розподілів і об’єднань?

Тестові завдання:

1. Це схема:



а) Однорівнева схема індексації

б) Дворівнева схема індексації

в) 1 і 2 варіант правильний

г) Правильної відповіді немає

2. Листова сторінка має такі властивості:

а) Ключ (1) <ключ (2) < …<ключ (t);

б) Ключ (1) <=ключ (2) <= …<=ключ (t);

в) Ключ (1) >ключ (2) > …>ключ (t);

г) Правильної відповіді немає

3. Засіб прискорення операції пошуку записів в таблиці, а отже, і інших операцій, що використовують пошук: витягання, модифікація, сортування і т.д., це:

а) Індекс

б) Ключ

в) Хеш-код

г) Правильної відповіді немає

4. Структура внутрішньої сторінки має такі властивості:

а) Ключ (1)<= ключ (2)<= … ключ (n);

б) Ключ (1) <ключ (2) < …<ключ (n);

в) Ключ (1) >ключ (2) > …>ключ (n);

г) Правильної відповіді немає

5. На сторінці дерева Nm знаходяться ключі k зі значеннями ключ:
а) (m) <= k<= ключ (m+1)
б) (m) => k=> ключ (m+1)
в) (m) <k< ключ (m+1)
г) Правильної відповіді немає

Розділ 20

Методологія функціонального моделювання

- **Методологія функціонального моделювання**
- **Декомпозиція діаграм**
- **Об’єктно-орієнтовані моделі**
- **Загальна характеристика UML**
- **Типи діаграм UML**
- **Приклади діаграм UML**

Методологія функціонального моделювання SADT служить для побудови функціональної моделі об'єкту якої-небудь наочної області. Остання відображає функціональну структуру об'єкту - виконувані ним дії і зв'язки між ними.

Функціональна модель інформаційної системи складається з тих, що мають посилання один до одного діаграм, фрагментів текстів і глосарію. На діаграмах представляються функції ІС і взаємозв'язки (інтерфейси) між ними у вигляді блоків і дуг. Місце з'єднання дуги з блоком визначає тип інтерфейсу. Управляюча інформація вказується зверху, оброблювана інформація - з лівого боку блоку, інформація, що виводиться, - з правої сторони, виконуючий операцію механізм (людина, програма або пристрій), представляється дугою знизу блоку.

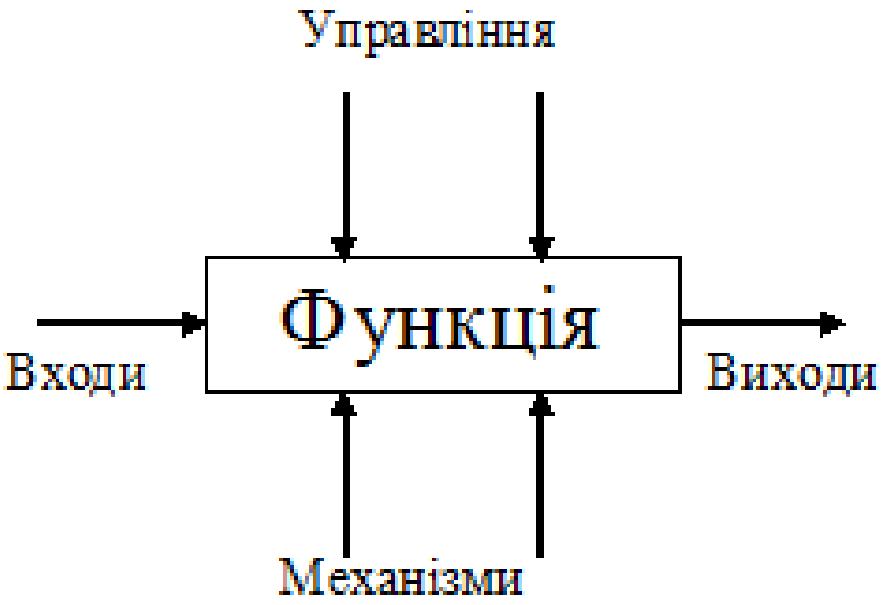


Рисунок 20.1 – Функціональний блок і дуги інтерфейсу

З використанням методології SADT виконується поступове нарощування ступеня деталізації в побудові моделі інформаційної системи. На рисунку показана декомпозиція початкового блоку системи на три становлячі компоненти. Кожний з блоків визначає підфункції початкової функції і, у свою чергу, може бути декомпонованим аналогічним чином для забезпечення більшої деталізації.

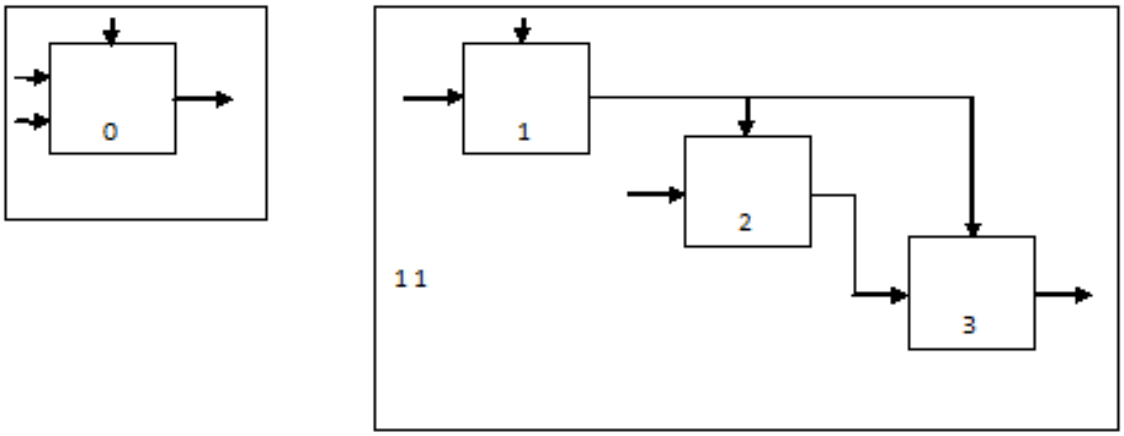


Рисунок 20.2 – Декомпозиція діаграм

В загальному випадку функціональна модель IC є серією діаграм з документацією, декомпозуючи складний об'єкт на складові компоненти у вигляді блоків. Блоки на діаграмі нумеруються. Для вказівки положення діаграми або блоку в ієрархії діаграм використовуються номери діаграм. Наприклад, позначення А32 указує на діаграму, що деталізує блок 2 на діаграмі А3. У свою чергу, діаграма АОЗ деталізує блок 3 на діаграмі АТ.

На діаграмах функціональної моделі SADT послідовність і час явно не вказуються. Зворотні зв'язки, ітерації, процеси і що перекриваються за часом функції можна відобразити за допомогою дуг.

В методології функціонального моделювання істотною властивістю є відображення можливих типів зв'язків між функціями. Можна виділити наступні сім типів зв'язків в порядку зростання значущості:

• **випадкові зв'язки, що** означають, що зв'язок між функціями малий або відсутній;

• **логічні зв'язки, що** означають, що дані і функції відносяться до одного класу або набору елементів, але функціональних відношень між ними немає;

• **тимчасові зв'язки** представляють функції, зв'язані в часі, коли дані використовуються одночасно або функції включаються паралельно, а не послідовно;

• **процедурні зв'язки** означають, що функції групуються разом, оскільки виконуються протягом однієї і тієї ж частини циклу або процесу;

• **комунікаційні зв'язки** означають, що функції групуються разом, оскільки використовують одні і ті ж вхідні дані або породжують одні і ті ж вихідні дані;

• **послідовні зв'язки** служать для позначення причинно-наслідкової залежності - вихідні дані однієї функції є вхідними даними іншої функції;

• **функціональні зв'язки** позначають випадок, коли всі елементи функції впливають на виконання тільки однієї функції.

Об'єктно-орієнтовані моделі

Більшість моделей об'єктно-орієнтованого проектування близька по можливостях, але мають відмінності в основному у формі представлення. Популярність об'єктно-орієнтованих технологій привела до зближення більшості відомих моделей. Різноманіття моделей породжує труднощі проектувальників по вибору моделі і по обміну інформацією при роботі над різними проектами. В зв'язку з цим відомі фахівці Г.Буч, Д.Рамбо і І.Джекобсон при підтримці фірми Rational Software Corporation провели роботу над уніфікованою моделлю і методом, що отримав назву UML (Unified Modeling Language - уніфікована мова моделювання).

Загальна характеристика UML

UML є єдиною мовою моделювання, призначеною для специфікації, візуалізації, конструювання і документування матеріалів програмних систем, а також для моделювання бізнесу і інших не програмних систем. В основу створення UML покладено три найпоширеніші моделі:

- Booch, що отримала назву по прізвищу автора Граді Буча (Grady Booch);
- OMT (Object Modeling Technique - метод моделювання об'єктів);
- OOSE (Object-Oriented Software Engineering - об'єктно-орієнтоване проектування програмного забезпечення).

На заключній стадії розробки, уніфікації і ухвалення UML як стандарт великий внесок вніс консорціум OMG (Object Management Group - група управління об'єктом). UML можна визначити також як промисловий об'єктно-орієнтований **стандарт моделювання**. Він включає в уніфікованому вигляді кращі методи візуального (графічного) моделювання. В даний час є цілий ряд інструментальних засобів, виробникияких заявляють про підтримкуUML, серед них

можна виділити: Rational Rose, Select Enterprise, Platinum і Visual Modeler.

Типи діаграм UML

Створюваний за допомогою UML проект інформаційної системи може включати наступні **8 видів діаграм** (diagrams):

• прецедентів використання (use case);
- класів (class)
- станів (statechart)
- активності (activity)
- проходження (sequence)
- співпраці (collaboration)
- компонентів (component)
- розміщення (deployment).

Діаграми **станів, активності, проходження** і **співпраці** утворюють набір діаграм, що служать для опису **поведінки** інформаційної системи, що розробляється. Причому, останні дві забезпечують опис **взаємодії об'єктів** інформаційної системи. Діаграми **компонентів** і **розміщення** описують фізичну реалізацію інформаційної системи.

Кожна з діаграм може містити елементи певного типу. Типи допустимих елементів і відношень між ними залежать від виду діаграми. Охарактеризуємо вказані види діаграм більш детально.

Діаграми **прецедентів використання** описують функціональність ІС, видиму користувачами системи. Кожна функціональність зображується у вигляді прецедентів використання. Прецедент - це типова взаємодія користувача з системою, яка виконує наступне:

- описує видиму користувачем функцію
- представляє різні рівні деталізації
- забезпечує досягнення конкретної мети.

Прецедент зображується як овал, пов'язаний з типовими користувачами, званими «акторами» (actors). Актором є будь-яка суттєвість, що взаємодіє з системою ззовні, наприклад, людина, обладнання, інша система. Прецедент описує, що система надає актору - визначає набір транзакцій, виконуваний актором при діалозі з інформаційною системою. На діаграмі зображується один актор, але користувачів, виступаючих в ролі актора, може бути багато. Діаграма прецедентів використання має високий рівень абстракції і дозволяє визначити функціональні вимоги до ІС.

Діаграми **класів** описують **статичну** структуру класів. До складу діаграм класів входять наступні елементи: класи, об'єкти і відношення між ними. Клас представляється прямокутником, що включає три розділи: ім'я класу, атрибути і операції. Аналогічне позначення застосовується і для об'єктів, з тією різницею, що до імені класу додається ім'я об'єкту і весь напис підкреслюється.

Допускається відображення додаткової інформації (абстрактні операції і класи, стереотипи, загальні і приватні методи, інтерфейси, і т. д.). Асоціації, або статичні зв'язки, між класами зображуються у вигляді зв'язуючої лінії, на якій може указуватися потужність асоціації, напрям, назва, і можливе обмеження. Можна відобразити властивості асоціації, наприклад, відношення агрегації, коли складовими частинами класу є інші класи. Відношення агрегації зображується у вигляді ромба, розташованого поряд з агрегуючим класом. Відношення узагальнення зображується у вигляді трикутника і зв'язуючої лінії, дозволяючи представити ієрархію спадкоємства класів.

Діаграми **станів** описують поведінку об'єкту в часі, моделюють всі можливі зміни в стані об'єкту, викликані зовнішніми діями з боку інших об'єктів або ззовні. Діаграми станів застосовуються для опису поведінки об'єктів і для опису операцій класів. Цей тип діаграм описує зміну стану одного класу або об'єкту. Кожний стан об'єкту представляється у вигляді прямокутника із закругленими кутами, що містить ім'я стану і, можливо, значення атрибутів об'єкту в даний момент часу. Перехід здійснюється при настанні деякої події (наприклад, отримання об'єктом повідомлення або прийом сигналу) і зображується у вигляді стрілки, що сполучає два сусідні стани. Ім'я події вказується на переході. На переході можуть вказуватися також дії, вироблювані об'єктом у відповідь на зовнішні події.

Діаграми **активності** представляють окремий випадок діаграм станів. Кожний стан є виконання деякої операції, і перехід в наступний стан відбувається при завершенні операції в початковому стані. Тим самим реалізується процедурне, синхронне управління, обумовлене завершенням внутрішніх дій. Описуваний стан не має внутрішніх переходів і переходів по зовнішніх подіях.

Для діаграм активності використовуються аналогічні діаграмам станів позначення, але на переходах відсутня сигнатура події і доданий символ «синхронізації» переходів для реалізації паралельних алгоритмів. Діаграми активності використовуються в основному для опису операцій класів.

Діаграма **проходження** визначає тимчасову послідовність (динаміку взаємодії) переданих повідомлень, порядок, вигляд і ім'я повідомлення. На діаграмі зображаються об'єкти, що безпосередньо беруть участь у взаємодії, і не показують статичні асоціації з іншими об'єктами.

Діаграма **проходження** має два зображення. Перше - зліва направо, у вигляді вертикальних ліній, що зображують об'єкти, що беруть участь у взаємодії. Верхня частина ліній доповнюється прямокутником, що містить ім'я класу об'єкту або ім'я екземпляра об'єкту. Друге вимірювання - вертикальна тимчасова вісь. Посильні повідомлення зображаються у вигляді стрілок з ім'ям повідомлення і впорядковані за часом виникнення.

Діаграми **співпраці** описують взаємодію об'єктів системи, виконуваної ними для отримання деякого результату. Під отриманням результату мається на увазі виконання закінченої дії, наприклад, опис в термінах взаємодіючих об'єктів змодельованого раніше **прецеденту використання** інформаційної системи або деякого сервісу, оголошеного як операція класу на відповідній діаграмі.

Діаграма співпраці зображує об'єкти, що беруть участь у взаємодії, у вигляді прямокутників, що містять ім'я об'єкту, його клас і, можливо, значення атрибутів. Асоціації між об'єктами зображуються у вигляді сполучних ліній. Можлива вказівка імені асоціації і ролей об'єктів в даній асоціації. Динамічні зв'язки (потоки повідомлень) представляються у вигляді сполучних ліній між об'єктами, зверху яких розташовується стрілка з вказівкою напряму і імені повідомлення.

Діаграма **компонентів** служить для визначення архітектури системи, що розробляється, шляхом встановлення залежності між програмними компонентами: початковим, бінарним або виконуваним кодом. В багатьох середовищах розробки модуль, або компонент, відповідає файлу. Пунктирні стрілки, сполучаючи модулі, показують відношення взаємозалежності (як при компіляції).

Діаграми **розміщення** використовуються для задання конфігурації компонентів, процесів і об'єктів, діючих в системі на етапі виконання. Крім того, вони показують фізичну залежність апаратних пристроїв, що беруть участь в реалізації системи, і з'єднань між ними - маршрутів передачі інформації.

Приклади діаграм UML

Щоб отримати більш наочне представлення, приведемо ряд діаграм UML. Розглянемо приклад, в якому описана об'єктна модель, побудована в Rational Rose 98. Як наочна область використовується опис

роботи бібліотеки, яка одержує запити від клієнтів на різні видання і реєструє інформацію про їх повернення до фондів бібліотеки.

Приклад діаграми **прецедентів використання** наведений на мал. 20.3. На діаграмі приведений ряд виділених при аналізі реалізованих інформаційною системою функцій: адміністрування користувачів (Administrative Client); облік книг (Administrative Books); складання звітів (Report) і пошук видання (Find Book).

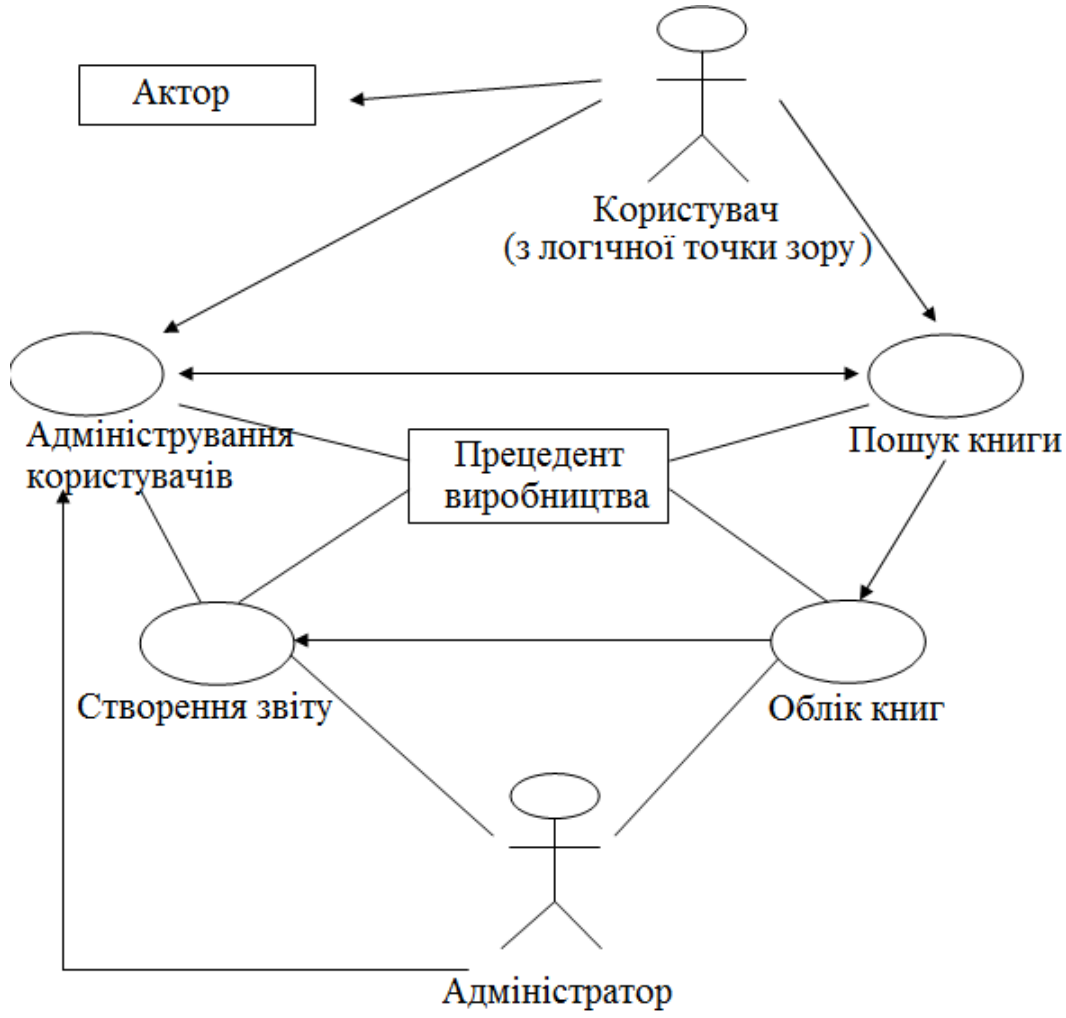


Рисунок 20.3 – Діаграма прецедентів використання

Приклад діаграми **проходження** наведений на рисунку 20.4. Приведена діаграма описує поведінку об'єктів в часі. Вона показує об'єкти і послідовність повідомлень, посланих об'єктами.

Відзначимо, що побудова моделі ІС до її програмної розробки є необхідним етапом проектування. Хороші моделі дозволяють налагодити конструктивну взаємодію між замовниками, користувачами і розробниками. Діаграми UML забезпечують ясне представлення архітектурних рішень для системи, що розробляється.

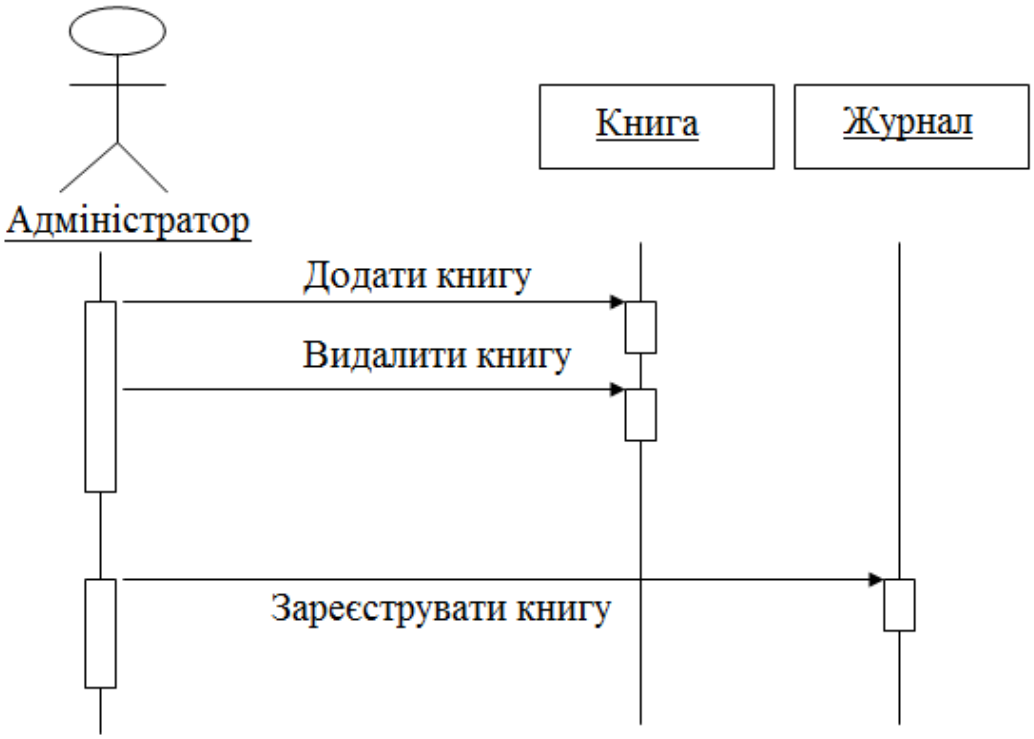


Рисунок 20.4 – Діаграма проходження

Складність інформаційних систем росте і як наслідок зростає актуальність застосування ефективних мов моделювання таких як UML.

Контрольні запитання:

1. Скільки існує типів зв'язків в порядку зростання значущості?
2. Які існують зв’язки значущості?
3. Напишіть про діаграму станів.
4. Напишіть про діаграму проходження.
5. Напишіть про діаграму активності.
6. Напишіть про діаграму співпраці.
7. Напишіть про діаграму компонентів.
8. Намалюйте приклад діаграми проходження.
9. Намалюйте приклад діаграми прецедентів використання.

Тестові завдання:

1. В методології функціонального моделювання істотною властивістю є відображення можливих типів зв'язків між функціями. Скільки таких типів зв’язків можна виділити?

а) 5

б) 6

в) 7

г) Правильної відповіді немає

2. Представляють функції, зв'язані в часі, коли дані використовуються одночасно або функції включаються паралельно, а не послідовно, це:
а) Тимчасові зв’язки
б) Процедурні зв’язки
в) Логічні зв’язки
г) Послідовні зв’язки

3. Скільки може включати видів діаграм створюваний за допомогою UML проект інформаційної системи?
а) 6 видів
б) 8 видів
в) 12 видів
г) Правильної відповіді немає

4. Які моделі покладено в основу створення UML?
а) Booch, OMT, OOSE
б) OMT, OOSE, MVC
в) Booch, OMT, OOSE, MVC
г) Правильної відповіді немає

5. Визначає тимчасову послідовність переданих повідомлень, порядок, вигляд і ім'я повідомлення, це:
а) Діаграма проходження
б) Діаграма активності
в) Діаграма станів
г) Правильної відповіді немає

Розділ 21

Інформаційні системи в мережах

- **Основні поняття мережі ЕОМ**
- **Програмне забезпечення ЛОМ**
- **Апаратні засоби обчислювальних мереж**
- **Принципи управління обчислювальних мереж**
- **Моделі архітектури клієнт-сервер**
- **Керування розподіленими даними**

Створення і застосування інформаційних систем в мережах комп'ютерів, з одного боку, дає помітні переваги, з другого боку, викликає ряд проблем. Зокрема, виникають проблеми адміністрування і захисту інформації. В розділі розглядаються основні поняття, пов'язані з мережами комп'ютерів і інформаційними системами в них, варіанти архітектури клієнт-сервер, управління даними і доступ до них, особливості інформаційних систем в локальних мережах, Internet і Intranet.

Основні поняття

Основні поняття мереж ЕОМ розкриємо, розглядаючи види і склад мереж, програмне і апаратне забезпечення, а також методи управління ресурсами, що використовуються.

Види і склад мереж

Мережею називається сукупність комп'ютерів або робочих станцій (РС), з'єднаних засобами передачі даних. Засоби передачі даних, у свою чергу, в загальному випадку можуть складатися з наступних елементів: зв'язних комп'ютерів, каналів зв'язку (супутникових, телефонних, цифрових, волоконно-оптичних, радіо- та інших), комутуючої апаратури, ретрансляторів, різного роду перетворювачів сигналів і інших елементів і пристроїв.

Сучасні мережі можна класифікувати по різних ознаках: по віддаленості комп'ютерів, топології, призначенню, переліку послуг, що надаються, принципам управління (централізовані і децентралізовані), методам комутації (без комутації, телефонна комутація, комутація ланцюгів, повідомлень, пакетів і дейтаграмм), видам середовища передачі і т.д.

Залежно від віддаленості комп'ютерів мережі умовно розділяють на ***локальні*** і ***глобальні.***

Довільна ***глобальна мережа*** може включати інші глобальні мережі, локальні мережі, а також окремі комп'ютери, що підключаються до неї (віддалені комп'ютери) або пристрої введення-виведення, що окремо підключаються. Глобальні мережі розрізняють чотирьох основних видів: міські, регіональні, національні і транснаціональні. Як пристрої введення-виведення можуть використовуватися, наприклад, друкуючі і копіюючі пристрої, касові і банківські апарати, дисплеї (термінали) і факси. Перераховані елементи мережі можуть бути віддалені один від одного на значну відстань. Прикладом глобальної мережі служить одна з перших в світі мережа ARPANET, а також мережа Інтернет.

В ***локальних обчислювальних мережах*** (ЛОМ) комп'ютери розташовані на відстані до декількох кілометрів і звичайно сполучені за допомогою швидкісних ліній зв'язку із швидкістю обміну від 1 до 10 і більш Мбіт/с (не виключається з'єднання комп'ютерів і за допомогою низькошвидкісних телефонних ліній). ЛОМ звичайно розгортаються в рамках деякої організації (корпорації, установи). Тому їх іноді називають корпоративними системами або мережами. Комп'ютери при цьому, як правило, знаходяться в межах одного приміщення, будівлі або сусідніх будівель.

В глобальній мережі основним видом взаємодії між незалежними комп'ютерами є обмін повідомленнями. Більш інтенсивний обмін інформацією відбувається в локальних мережах. В них, по суті, організовано управління апаратно-програмними ресурсами всіх комп'ютерів, що входять в мережу. Реалізує ці функції мережне ПЗ. Коротко зупинимося на програмно-апаратних ресурсах і методах управління ними в ЛОМ.

Програмне забезпечення ЛОМ

Незалежно від того, в якій мережі працює деякий комп'ютер, функції встановленого на ньому програмного забезпечення умовно можна розділити на дві групи: *управління ресурсами* самого комп'ютера (у тому числі і на користь рішення задач для інших комп'ютерів) і *управління обміном* з іншими комп'ютерами (мережні функції). Власними ресурсами комп'ютера традиційно управляє ОС. Функції мережного управління реалізує ***мережне ПЗ,*** яке може бути виконано як у вигляді окремих пакетів мережних програм, так і в вигляді мережної ОС (МОС).

При розробці мережного ПЗ використовується ієрархічний підхід, що припускає визначення сукупності порівняно незалежних рівнів і

інтерфейсів між ними. Це дозволяє легко модифікувати алгоритми програм довільного рівня без істотної зміни інших рівнів. В загальному випадку допускається спрощення функцій деякого рівня або навіть його повна ліквідація.

Для впорядкування розробки мережного ПЗ і забезпечення можливості взаємодії будь-яких обчислювальних систем, Міжнародна Організація по Стандартизації (International Organization for Standardization — ISO) розробила *Еталонну модель взаємодії відкритих систем* (Open System Interconnection Reference model — модель OSI).

Еталонна Модель визначає наступні сім функціональних рівнів:

- фізичний (physical layer);
- канальний або управління лінією (ланкою) передачі (data link);
- мережний (network layer);
- транспортний (transport layer);
- сеансовий (session layer);
- представницький (presentation layer);
- прикладний або рівень додатків (application layer) .

Відмінності мереж одна від одної викликані особливостями апаратного і програмного забезпечення, що використовується, різною інтерпретацією рекомендацій фірмами-розробниками, відмінністю вимог до системи із сторони вирішуваних задач (вимоги захищеності інформації, швидкості обміну, безпомилковості передачі даних і т.д.) і іншими причинами. В мережному ПЗ локальних мереж часто спостерігається скорочення числа реалізованих рівнів.

Апаратні засоби ЛОМ

Основними апаратними компонентами ЛОМ є: робочі станції, сервери, інтерфейсна плата і кабелі.

Робочі станції (РС) - це, як правило, персональні ЕОМ, які є робочими місцями користувачів мережі.

Іноді в РС, безпосередньо підключеної до мережного кабелю, можуть бути відсутні накопичувачі на магнітних дисках. Такі РС називають ***бездисковими робочими станціями.*** Основною *перевагою* бездискових РС є низька вартість, а також висока захищеність від несанкціонованого проникнення в систему користувачів і комп'ютерних вірусів. *Недолік* бездискової РС полягає в неможливості працювати в автономному режимі (без підключення до серверу) і мати власні архіви даних і програм.

Сервери в ЛОМ виконують функції розподілу мережних ресурсів. Звичайно його функції покладають на достатньо потужний ПК, МІНІ-ЕОМ, велику ЕОМ або спеціальну ЕОМ-сервер. В одній мережі може бути один або декілька серверів. Кожний з серверів може бути окремим або суміщеним з РС. В останньому випадку тільки частина ресурсів серверу виявляється загальнодоступною.

За наявності в ЛОМ декількох серверів, кожний з них управляє роботою підключених до нього РС. Сукупність комп'ютерів серверу, що відносяться до цього РС часто називають ***доменом.*** Іноді в одному домені знаходиться декілька серверів. Звичайно один з них є головним, а інші — виконують роль резерву (на випадок відмови головного серверу) або логічного розширення основного серверу.

Мережні адаптери, що використовуються, мають три основні характеристики: тип шини комп'ютера, до якого вони підключаються (ISA, EISA, Micro Channel і ін.), розрядність (8, 16, 32, 64) і метод доступу, що використовується, до мережного каналу даних.

Існують різні схеми об'єднання комп'ютерів в ЛОМ. Класичними вважаються топології «зірка», «кільце» і «загальна шина». Найбільш широко використовуються наступні стандартизовані методи доступу до мережного каналу:

- Ethernet (підтримує шинну топологію);
- Arcnet (підтримує зоряну топологію);
- Token-Ring (підтримує кільцеву топологію).

Конфігурація з'єднання елементів в мережу (топологія) багато в чому визначає такі найважливіші характеристики мережі, як її надійність, продуктивність, вартість, захищеність і т.д.

Одним з підходів до класифікації топології ЛОМ є виділення двох основних класів топологій: *широкомовних* і *послідовних.*

В *широкомовних* конфігураціях кожний персональний комп'ютер передає сигнали, які можуть бути сприйняті рештою комп'ютерів. До таких конфігурацій відносяться топології «загальна шина», «дерево», «зірка з пасивним центром». Мережу типу «зірка з пасивним центром» можна розглядати як різновид «дерева», що має корінь з відгалуженням до кожного підключеного пристрою.

В *послідовних* конфігураціях кожний фізичний підрівень передає інформацію одному комп'ютеру. Прикладами послідовних конфігурацій є: довільна (довільне з'єднання комп'ютерів), ієрархічна, «кільце», «ланцюжок», «зірка з інтелектуальним центром», «сніжинка».

Для з'єднання комп'ютерів в ЛОМ найчстіше всього використовують коаксиальний кабель (тонкий і товстий). Крім коаксиального кабелю може застосовуватися «вита пара» і оптоволокно. Останнім часом ведуться інтенсивні роботи по розробці і упровадженню беспровідних радіомереж. Відомі системи на їх основі, в порівнянні з кабельними системами, поки значно поступаються по швидкості передачами даних і дальності прийому (сотні метрів), але дозволяють створювати мобільні розподілені системи.

До додаткового обладнання ЛОМ відносять джерела безперебійного живлення, модеми, трансивери, повторювачі, а також різні роз'єми (коннектори, термінатори).

Принципи управління

Існує два основні методи (принципи) управління в ЛОМ: централізоване і децентралізоване. В *централізованих* мережах одна або декілька ПЕОМ є центральними, а інші — робочими станціями (РС). Центральний вузол мережі, часто званий комп'ютером-сервером (КС), може знаходитися на окремому комп'ютері або поєднуватися з РС. В першому випадку говорять про виділений КС, а в другому — про суміщений КС. Основне призначення КС — управління передачею даних в мережі і зберігання файлів, що використовуються багато якими РС. Виходячи з цього, під КС звичайно виділяють найпродуктивнішу ПЕОМ і ту що має значну пам'ять. Крім того, до КС звичайно підключається коштовне обладнання (лазерні принтери, факси, модеми, сканери і т. д.). Існує безліч мережних ОС, що реалізовують централізоване управління. Серед них Microsoft Windows NT Server, Novell NetWare (версії 3.X і 4.X), Microsoft Lan Manager, OS/2 Warp Server Advanced, VINES 6.0 і інші.

Перевагою централізованих мереж є висока захищеність мережних ресурсів від несанкціонованого доступу, зручність адміністрування мережі, можливість створення мереж з великим числом вузлів. Основний *недолік* полягає в вразливості системи при порушенні працездатності файл-серверу (це долається за наявності декількох серверів або використанні інших заходів), а також в пред'явленні досить високих вимог до ресурсів серверів.

Мережі з *децентралізованим* управлінням (однорангові мережі) не містять КС. В них функції управління мережею відповідно до деякої дисципліни по черзі передаються від однієї РС до іншої. Децентралізоване управління, як правило, застосовується в мережах із

слабкими комп'ютерами і найпростішими обмінами між ними на рівні файлів, а також без серйозного контролю прав доступу до ресурсів мережі. Основні ресурси всіх PC звичайно виявляються загальнодоступними, хоча система не завжди забезпечує коректне їх сумісне використання на прикладному рівні.

Найпоширенішими програмними продуктами, що дозволяють будувати однорангові мережі, є наступні програми і пакети: Novell NetWare Lite, Windows for Workgroups, Windows 95/98, Artisoft LANtastic, LANsmart, Invisible Software NET-30 і інші.

Розгортання однорангової мережі для невеликого числа PC часто дозволяє побудувати більш ефективне і живуче розподілене обчислювальне середовище. Мережне програмне забезпечення в них є більш простим в порівнянні з централізованими мережами. Тут не потрібна установка файл-серверу (комп'ютера і відповідних програм), що істотно здешевлює систему. Проте, такі мережі слабкі з погляду захисту інформації і адміністрування.

Кажучи про мережі, часто використовують терміни «сервер» і «клієнт». Будь-яка взаємодія в мережі припускає як мінімум два елементи: споживаючий і надаючий ресурси (сервіс або послуги). Споживач ресурсів називається клієнтом, а надаючий ресурси компонент - сервером. Основними видами ресурсів є наступні: апаратні (ціла ЕОМ, дисковий накопичувач, пристрій друку і т. д.), програмні і інформаційні.

Якщо як ресурс розглядається вся ЕОМ, то говорять про комп'ютер-клієнта і комп'ютер-сервер. Якщо мається на увазі деякий апаратний ресурс, то використовують такі терміни як: диск-сервер (файловий сервер або файл-сервер), сервер друку. Типовими клієнтами в мережах є: ЕОМ, користувач або програма.

Можливо уточнення призначення комп'ютера при обробці інформації. Тоді говорять про сервери повідомлень (обробка повідомлень, що надходять), сервери БД (обробка запитів до БД), сервери додатків (виконання додатків користувача) і т.д.

Іноді одним терміном називають різні (апаратні, програмні або апаратно-програмні) компоненти обчислювальної системи. Так, сервером друку може служити комп'ютер з підключеним до нього принтером, програма друку або комп'ютер з програмним забезпеченням управління друком.

Моделі архітектури клієнт-сервер

При побудові розподілених ІС, працюючих з БД, широко використовується архітектура клієнт-сервер. Її основу складають принципи організації взаємодії клієнта і серверу при управлінні БД.

Щоб охарактеризувати основні схеми взаємодії процесів управління БД, скористаємося Еталонною моделлю Архітектури відкритих систем OSI. Згідно цієї моделі, функція управління БД відноситься до прикладного рівня.

Зупинимося на двох самих верхніх рівнях: *прикладному* і *представницькому*, які найбільшою мірою є предметом уваги з боку розробника і користувача. Решту функцій вважатимемо зв'язними функціями, необхідними для реалізації двох перших. При цьому дотримуватимемося широкого тлумачення терміну СКБД, розуміючи під ним всі програмні системи, які використовують інформацію з БД.

Як підтримуюча інтерфейс з користувачем програма, СКБД, в загальному випадку, реалізує наступні основні функції:

- керування даними, що знаходяться в базі;
- обробка інформації за допомогою прикладних програм;
- представлення інформації в зручному для користувача вигляді.

Якщо система розміщується на одній ЕОМ, то всі функції зібрані в одній програмі і викликаються по схемі, подібній розглянутій. При розміщенні СКБД в мережі можливі різні варіанти розподілу функцій по вузлах. Залежно від числа вузлів мережі, між якими виконується розподіл функцій СКБД, можна виділити *дволанцюгові* моделі, *триланцюгові* моделі і т.д. Місце розриву функцій з'єднується комунікаційними функціями (середовищем передачі інформації в мережі).

Дволанцюгові моделі розподілу функцій

Дволанцюгові моделі відповідають розподілу функцій СКБД між *двома вузлами мережі.* Комп'ютер (вузол мережі), на якому обов'язково присутня функція управління даними, назвемо *комп'ютером-сервером.* Комп'ютер, близький до користувача і обов'язково займається питаннями представлення інформації, назвемо *комп'ютером-клієнтом.*

Найтиповішими варіантами (рисунок 21.1) розділення функцій між комп'ютером-сервером і комп'ютером-клієнтом є наступні(рисунок 21.1):

- розподілене представлення;
- віддалене представлення;

• розподілена функція;
• віддалений доступ до даних;
• розподілена БД.

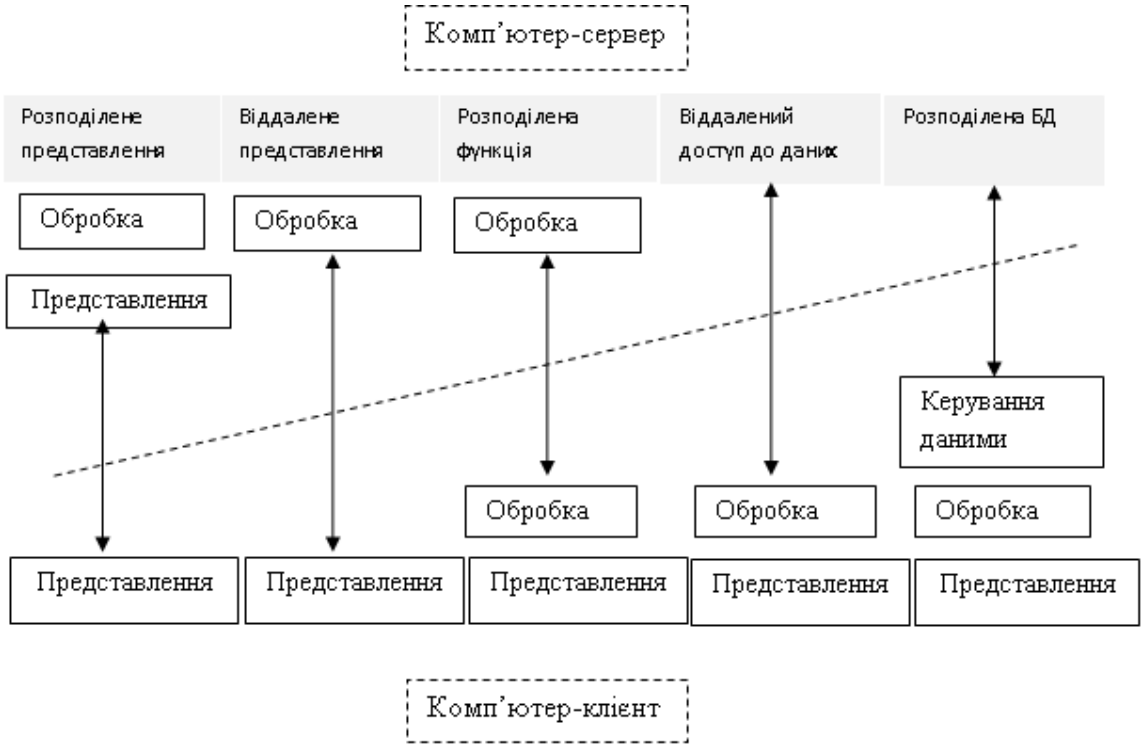


Рисунок 21.1 - Спектр моделей архітектури клієнт-сервер

Перераховані способи розподілу функцій в системах з архітектурою клієнт-сервер ілюструють різні варіанти: від могутнього серверу, коли практично вся робота проводиться на ньому, до потужного клієнта, коли велика частина функцій виконується на робочій станції, а сервер обробляє ті, що надходять до нього по мережі SQL.

В моделях віддаленого доступу до даних і віддаленого представлення проводиться жорсткий розподіл функцій між комп'ютером-клієнтом і комп'ютером-сервером. В інших моделях має місце виконання однієї з наступних функцій одночасно на двох комп'ютерах: управління даними (модель розподіленої БД), обробки інформації (модель розподіленої функції), представлення інформації (модель розподіленого представлення).

Розглянемо спочатку моделі *віддаленого доступу до даних* і *віддаленого представлення (серверу БД)* як найбільш широко поширених.

В моделі *віддаленого доступу до даних* (Remote Data Access — RDA) програми, реалізовуючи функції представлення інформації і логіку прикладної обробки, суміщені і виконуються на комп'ютері-

клієнті. Обіг за сервісом управління даними відбувається через середовище передачі за допомогою операторів мови SQL або викликом функцій спеціальної бібліотеки API (Application Programming Interface - інтерфейсу прикладного програмування).

Основна перевага RDA-моделі полягає у великій кількості готових СКБД, що мають SQL-інтерфейси, і існуючих інструментальних засобів, що забезпечують швидке створення програм клієнтської частини. Засоби розробки частіше за все підтримують графічний інтерфейс користувача в MS Windows, стандарт інтерфейсу ODBC і засоби автоматичної генерації коду. Переважна більшість засобів розробки використовує мови четвертого покоління.

Недоліками RDA-моделі є, по-перше, досить високе завантаження системи передачі даних внаслідок того, що вся логіка зосереджена в додатку, а оброблювані дані розташовані на віддаленому вузлі. Як побачимо далі, під час роботи додатків звичайно по мережі передаються цілі БД. По-друге, системи, побудовані на основі моделі RDA, незручні з погляду розробки, модифікації і супроводу. Основна причина полягає в тому, що в одержуваних додатках прикладні функції і функції представлення тісно взаємозв'язані. Тому навіть при незначній зміні функцій системи потрібна переробка всієї прикладної її частини, що ускладнює розробку і модифікацію системи.

Модель *серверу* БД (Data Base Server — DBS) відрізняється від попередньої моделі тим, що функції комп'ютера-клієнта обмежуються функціями представлення інформації, тоді як прикладні функції забезпечуються додатком, що знаходиться на комп'ютер-сервері. Ця модель є більш технологічною ніж RDA-модель і застосовується в таких СКБД, як Ingress, Sybase і Oracle. При цьому додатки реалізуються у вигляді *збережених процедур*.

Процедури звичайно зберігаються в словнику БД і розділяються декількома клієнтами. В загальному випадку збережені процедури можуть виконуватися в режимах компіляції і інтерпретації.

Перевагами моделі DBS є: можливість хорошого централізованого адміністрування додатків на етапах розробки, супроводи і модифікації а також ефективне використання обчислювальних і комунікаційних ресурсів. Останнє досягається за рахунок того, що виконання програм в режимі колективного користування вимагає значно менших витрат на пересилку даних в мережі.

Один з *недоліків* моделі DBS пов'язаний з обмеженнями засобів розробки збережених процедур. Основне обмеження — сильна прив'язка операторів збережених процедур до конкретної СКБД. Мова написання збережених процедур, по суті, є процедурним розширенням мови SQL, і не може змагатися по виразних засобах і функціональних можливостях з традиційними мовами третього покоління, такими, як З і Pascal. Крім того, в більшості СКБД немає задовільних засобів відладки і тестування збережених процедур, що робить їх механізм небезпечним інструментом — невідлагоджені програми можуть приводити до некоректностей БД, зависань серверних і клієнтських програм під час роботи системи і т.п. Іншим *недоліком* DBS-моделі є низька ефективність використання обчислювальних ресурсів ЕОМ, оскільки не вдається організувати управління вхідним потоком запитів до програм комп'ютера-серверу, а також забезпечити переміщення процедур на інші комп'ютери-сервери.

В моделі *розподіленого представлення* є потужний комп'ютер-сервер, а клієнтська частина системи практично вироджена. Функцією клієнтської частини є просто відображення інформації на екрані монітора і зв'язок з основним комп'ютером через локальну мережу. СКБД подібного роду можуть мати місце в мережах, що підтримують роботу так званих *X-терміналів*. В них основний комп'ютер (хост-машина) повинен мати достатню потужність, щоб обслуговувати декілька X-терміналів. X-термінал теж повинен володіти достатньо швидким процесором і мати достатній об'єм оперативної пам'яті (дискові накопичувачі відсутні). Часто X-термінали створюють на базі RISC-комп'ютерів (restricted [reduced] instructionset computer) - комп'ютерів з скороченимнаборомкоманд.Все програмне забезпечення знаходиться на хост-машині. Програмне забезпечення X-термінала, що виконує функції управління уявленням і мережні функції, завантажується по мережі з серверу при включенні X-термінала.

Модель розподіленого представлення мали СКБД ранніх поколінь, які працювали на малих, середніх і великих ЕОМ. В ролі X-терминалів виступали дисплейні станції і абонентні пункти (локальні і видалені). В цьому випадку основну частину функцій представлення інформації реалізовували самі СКБД, а остаточна побудова зображенні на терміналах користувача виконувалося на крайових пристроях.

По моделі розподіленого представлення побудовані системи обслуговування користувачів БД в гетерогенному (неоднорідному)

середовищі. Серверна частина таких систем звичайно забезпечує деякий уніфікований інтерфейс, а клієнтські частини реалізують функції обліку специфіки крайового обладнання або перетворення одного формату представлення інформації в іншій.

Модель розподіленого представлення реалізує централізовану схему управління обчислювальними ресурсами. Звідси слідують її основні *переваги* - простота обслуговування і управління доступом до системи і відносна дешевизна (зважаючи на невисоку вартість крайових терміналів). *Недоліками* моделі є вразливість системи при невисокій надійності центрального вузла, а також високі вимоги до серверу по продуктивності при великому числі клієнтів.

В моделі *розподіленої функції* логіка обробки даних розподілена по двом вузлам. Таку модель можуть мати ІС, в яких загальна частина прикладних функцій реалізована на комп'ютер-сервері, а специфічні функції обробки інформації виконуються на комп'ютері-клієнті. Функції загального характеру можуть включати стандартне забезпечення цілісності даних, наприклад, у вигляді збережених процедур, а прикладні функції, що залишилися, реалізують спеціальну прикладну обробку. Подібну модель мають також ІС, використовуючи інформацію з декількох неоднорідних БД.

Модель *розподіленої БД* припускає використання могутнього комп'ютера-клієнта, причому дані зберігаються на комп'ютері-клієнті і на комп'ютер-сервері. Взаємозв'язок обох баз даних може бути двох різновидів: а) в локальній і віддаленій базах зберігаються окремі частини єдиної БД; б) локальна і віддалена БД синхронізуються одина з одною копіями.

Перевагою моделі розподіленої БД є гнучкість створюваних на її основі ІС, дозволяючих комп'ютеру-клієнту обробляти локальні і віддалені БД. За наявності механізмів координації відповідності копій система в цілому, крім того, володіє високою живучістю, оскільки розривши з'єднання клієнта і серверу не призводить до краху системи, а її робота може бути відновлена з відновленням з'єднання. До *недоліку* моделі можна віднести високі витрати при виконанні великого числа однакових додатків на комп'ютерах-клієнтах.

Триланцюгові моделі розподілу функцій

Триланцюгова модель розподілу функції є типовим варіантом, при якому кожна з трьох функцій додатку реалізується на окремому комп'ютері. Варіанти розподілу функцій додатку на більше число

комп'ютерів можуть мати місце, але зважаючи на їх рідкісне застосування розглядатися не будуть. Модель що розглядається нами має назву *модель серверу додатків*, або *AS-модель* (Application Server) і показана на рисунку 21.2.

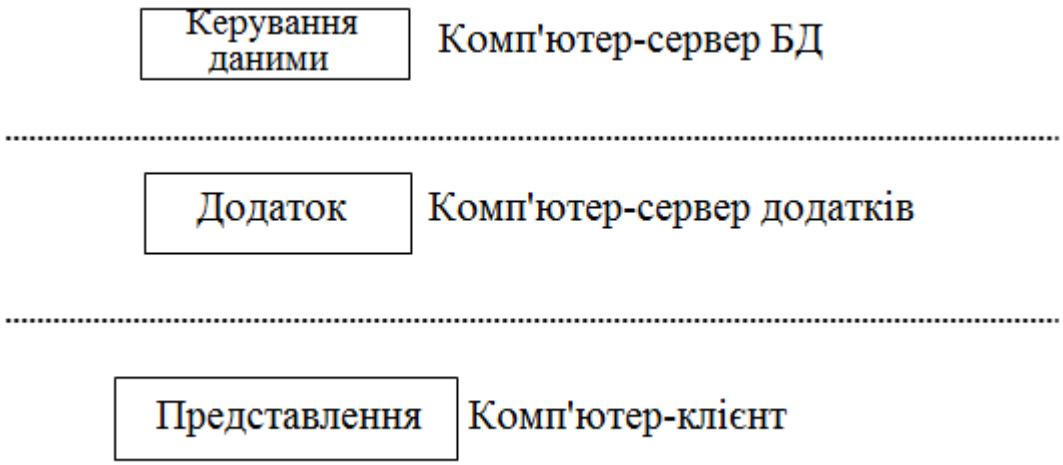


Рисунок 21.2 – Триланцюгова модель серверу додатків

Згідно триланцюгової AS-моделі, відповідаючий за організацію діалогу з кінцевим користувачем процес, як завжди, реалізує функції представлення інформації і взаємодіє з компонентом додатку так само, як в моделі DBS. Компонент додатку, мається свій в розпорядженні на окремому комп'ютері, у свою чергу, пов'язаний з компонентом управління даними подібно моделі RDA.

Центральною ланкою AS-моделі є сервер додатків. На сервері додатків реалізується декілька прикладних функцій, кожна з яких оформлена як служба надання послуг всім вимагаючим цього програмам. Серверів додатків може декілька, причому кожний з них надає свій вид сервісу. Будь-яка програма, що запрошує послугу у серверу додатків, є для нього клієнтом. Запити, що поступають від клієнтів до серверів поміщаються в чергу, з якої вибираються відповідно до деякого порядку, наприклад, по пріоритетах. Компонент, що реалізовує функції представлення і є клієнтом для серверу додатків, в цій моделі трактується більш широко, ніж звичайно. Він може служити для організації інтерфейсу з кінцевим користувачем, забезпечувати прийом даних від пристроїв, наприклад, датчиків, або бути довільною програмою.

Перевагою AS-моделі є гнучкість і універсальність внаслідок розділення функцій додатку на три незалежні складові. У багатьох

випадках ця модель виявляється більш ефективною в порівнянні з дволанцюговими. Основний недолік *моделі* — більш високі витрати ресурсів комп'ютерів на обмін інформацією між компонентами додатку в порівнянні з дволанцюговими моделями.

Складні схеми взаємодії

Можливі *складніші схеми взаємодії*, наприклад, схеми, в яких елемент, що є сервером для деякого клієнта, у свою чергу, виступає в ролі клієнта по відношенню до іншого серверу (рисунок 21.3). Приклад цього ми спостерігали в AS-моделі.

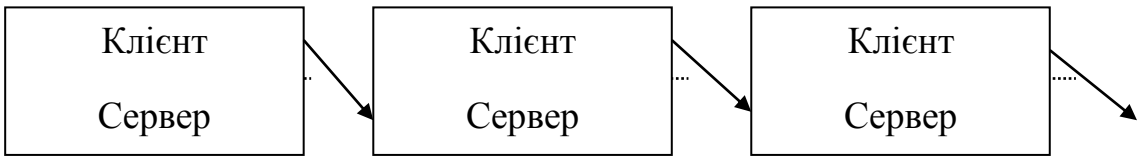


Рисунок 21.3 - Ланцюжок взаємодій типу клієнт-сервер

Можливо також, що в розподіленій обчислювальній системі при роботі з БД є *множинні* зв'язки (статичні), коли один об'єкт по відношенню до одних є клієнтом, а по відношенню до інших — сервером (рисунок 21.4).

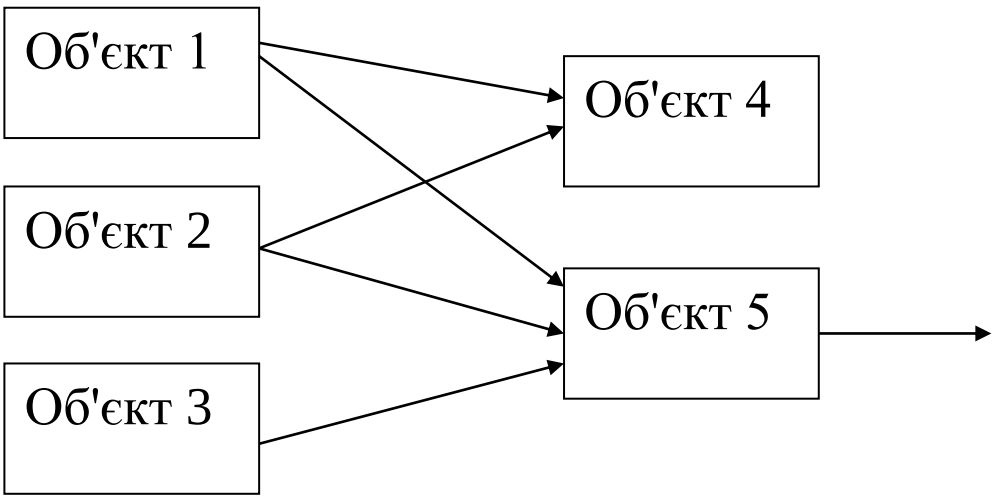


Рисунок 21.4 - Множинні зв'язки взаємодії типу клієнт-сервер

При розгляді взаємодії об'єктів в *динаміці* виходять ще складніші схеми взаємодії. Прикладом такої схеми є випадок, коли в процесі роботи ролі об'єктів міняються: об'єкт, що є в деякий момент часу клієнтом по відношенню до іншого об'єкту, в подальшому стає сервером для іншого об'єкту.

Модель монітора транзакцій

Як наголошувалося, найгнучкішою і універсальною моделлю розподілу функцій СКБД є AS-модель. Вона описує взаємодію трьох основних елементів: клієнта, серверу додатків і серверу БД, але не піднімає питання організації функціонування програмного забезпечення при обробці інформації, зокрема при виконанні транзакцій. Для подолання цього *недоліку* запропонована *модель монітора транзакцій.*

Монітори обробки транзакцій (Transaction Processing Monitor — TPM), або *монітори транзакцій*, є програмними системами категорії проміжного шару (Middleware), що забезпечують ефективне управління інформаційно-обчислювальними ресурсами в розподіленій обчислювальній системі. Основне їх призначення — організація гнучкого, відкритого середовища для розробки і управління мобільними додатками, які розподілені транзакції. Використання моніторів транзакцій найбільш ефективно в гетерогенних обчислювальних системах.

Додаток TPM дозволяє виконувати масштабування системи, підтримувати функціональну повноту і цілісність додатків а також підвищити продуктивність обробки даних при невисокій вартості невигідних витрат.

Принципи організації обробки інформації за допомогою монітора транзакцій описуються *моделлю монітора транзакцій* X/Open DTP (Distributed Transaction Processing — обробка розподілених транзакцій). Ця модель (рисунок21.5) включає три об'єкти: прикладну програму, менеджер ресурсів (ResourceManager - RM) і монітор, або менеджер, транзакцій (Transaction Manager — TM).

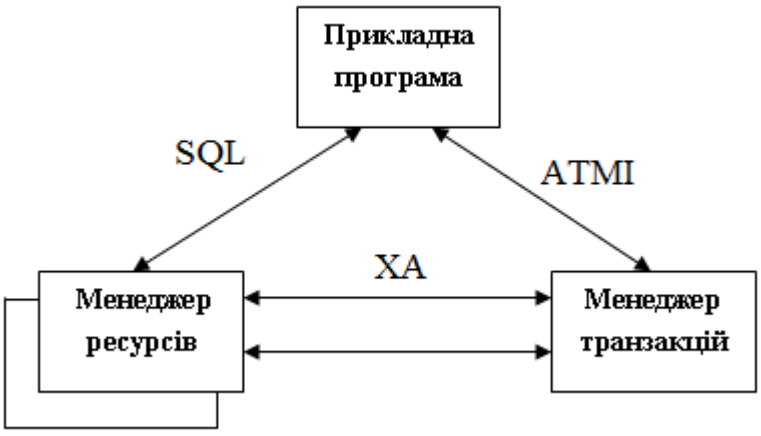


Рисунок 21.5 - Модель обробки транзакцій X/OpenDTP

Як прикладна програма може виступати довільна програма-клієнт. RM виконує функції серверу ресурсу деякого вигляду. Прикладна програма взаємодіє з RM за допомогою набору спеціальних функцій або за допомогою операторів SQL (коли сервером є сервер БД).

Інтерфейс ATMI (Application Transaction Monitor Interface — інтерфейс монітора транзакцій додатку) дозволяє викликати функції TPM на деякій мові програмування, наприклад. Функції менеджера ресурсів звичайно виконують сервери БД або СКБД. В задачах організації управління обробкою розподілених транзакцій (транзакцій, що зачіпають програмні об'єкти обчислювальної мережі) TM взаємодіє з RM, який повинен підтримувати протокол двофазної фіксації транзакцій і задовольняти стандарту X/Open XA. Прикладами СКБД, що підтримують протокол двофазної фіксаціїтранзакції, є Oracle 7.0, Open INGRES і Informix-Online 5.0.

Поняття транзакції в TPM дещо ширше, ніж в СКБД, де транзакція включає елементарні дії над даними бази. Тут транзакція може охоплювати і інші необхідні дії: передачу повідомлення, запис інформації у файл, опит датчиків і т.д. Це значить, що TPM надає більш потужні засоби управління обчислювальним процесом. Транзакції, які підтримують TPM, називають також прикладними або бізнес-транзакціями.

Модель X/Open DTP не розкриває структуру TM в деталях, а визначає склад компонентів розподіленої системи обробки інформації і як ці компоненти взаємодіють один з одним. Практичні реалізації цієї моделі, природно, можуть відрізнятися один від одного. В числі прикладів реалізацій моніторів транзакцій можна назвати ACMS, CICS, і TUXEDO System.

Для розробників додатків монітори обробки транзакцій TPM надають зручності, пов'язані з можливістю декомпозиції додатків по декількох функціональних рівнях із стандартними інтерфейсами, що дозволяє створювати такі, що легко модифікуютьсяіз струнокою архітектурою.

Прикладні програми стають практично незалежними від конкретної реалізації інтерфейсу з користувачем і від менеджера ресурсів. Перше означає, що для реалізації функцій представлення можна вибрати будь-який зручний і звичний для розробника засіб (від мови до якої-небудь CASE-системы). Незалежність від менеджера ресурсів має на увазі можливість легкої заміни одного менеджера ресурсів на іншій, лише б

вони підтримували стандарт взаємодії з прикладною програмою (для СКБД — мова SQL).

Якщо ТРМ підтримує безліч апаратно-програмних платформ, як наприклад, TUXEDO System, то додатки, що розробляються, стають, крім того, мобільними.

Зосередження всіх прикладних функцій в серверах додатків і наявність багатих можливостей управління і адміністрування істотно спрощує оновлення прикладних функцій (бізнес-функцій) і контроль за їх несуперечністю. Зміни в прикладних функціях при цьому ніяк не відображаються на програмах-клієнтах.

Для *користувачів розподілених систем* ТРМ дозволяють поліпшити показники пропускної спроможності і часу відгуку, понизити вартість обробки даних в оперативному режимі на основі ефективної організації обчислювального процесу.

Поліпшення показників функціонування досягається завдяки здійсненню статичного і динамічного балансування навантаження. Управління завантаженням полягає в запуску або зупинці AS-процесів (програмних компонентів AS-модели) залежно від наперед встановлених параметрів і поточного стану системи. При необхідності ТРМ може тиражувати копії AS-процесів на цьому або інших вузлах мережі.

Адміністратори розподілених систем, маючи ТРМ, дістають можливість легкого масштабування ІС і збільшення продуктивності обробки інформації. Тут, окрім вертикального і горизонтального масштабування, можна забезпечити так зване *матричне масштабування.* Суть його — введення додаткових ресурсів в будь-яку точку гетерогенного обчислювального середовища без зміни архітектури додатку, виконуваного в новому середовищі. Це означає, що без зупинки серверів додатків у будь-який час може бути доданий, наприклад, комп'ютер або менеджер ресурсів.

Крім того, адміністратори систем дістають можливість понизити загальну вартість програмного забезпечення систем клієнт-сервер. Зниження вартості можна досягти простим зменшенням кількості підключень до серверів БД.

Вартість серверів БД (або СКБД) в сильному ступені залежить від числа одночасних підключень до програми. Зменшення кількості підключень до серверу БД досягається шляхом мультиплексування запитів або, що те ж саме, логічного перетворення потоку запитів від

багатьох джерел до потоку від одного або декількох джерел. Виграш у вартості і продуктивності при ефективній реалізації самих TPM може виявитися вельми істотним.

Керування розподіленими даними

З керуванням даними в розподілених системах пов'язано наступні дві групи проблем: підтримка відповідності БД змінам, що вносяться, і забезпечення сумісного доступу декількох користувачів до загальних даних.

Підтримка відповідності БД змінам, що вносяться

В сучасних розподілених системах інформація може зберігатися *централізовано* або *децентралізовано*. В першому випадку проблеми ідентичності представлення інформації для всіх користувачів не існує, оскільки всі останні зміни зберігаються в одному місці. На практиці частіше інформація змінюється одночасно в декількох вузлах розподіленої обчислювальної системи. В цьому випадку виникає проблема контролю за всіма змінами інформації і надання її в достовірному вигляді всім користувачам.

Існують дві основні технології децентралізованого управління БД: *розподілених БД* (Distributed Database) і *тиражування, або реплікації, БД* (Data Replication).

Розподілена БД складається з декількох фрагментів, розміщених на різних вузлах мережі і, можливо, керованих різними СКБД. З погляду програм і користувачів, що звертаються до розподіленої БД, остання сприймається як єдина локальна БД (рисунок 21.6).

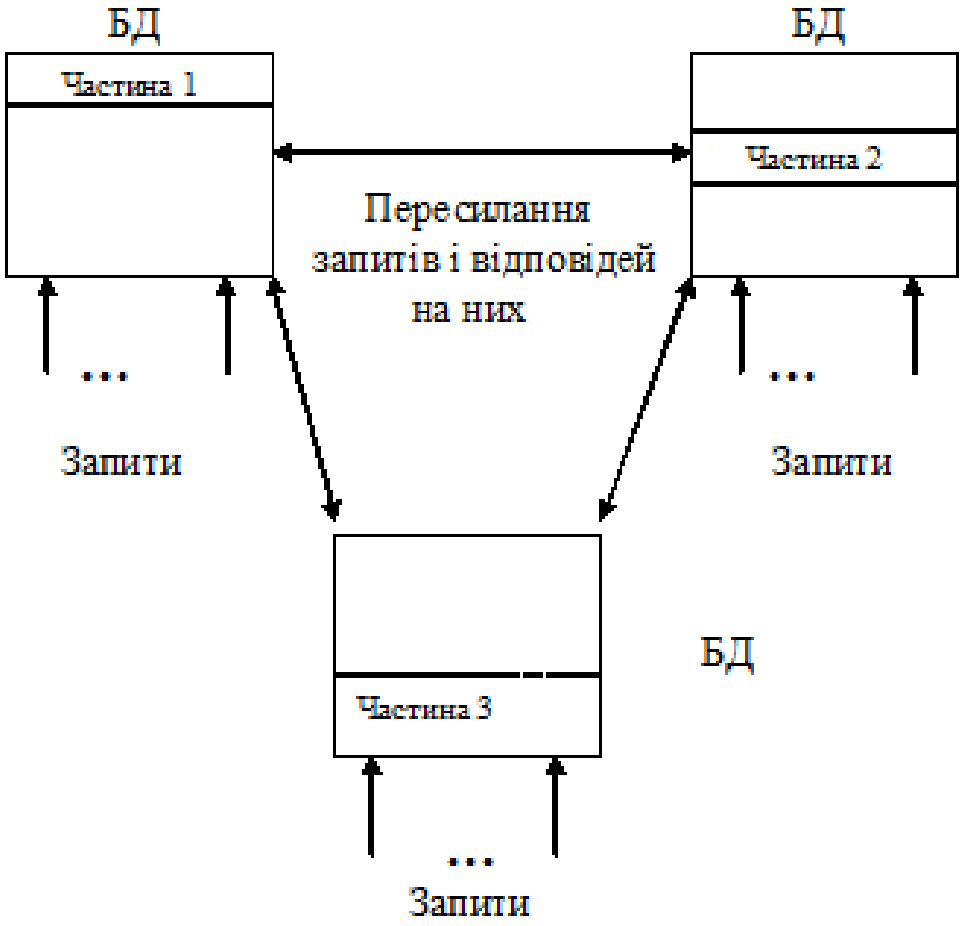


Рисунок 21.6 - Модель розподіленої БД

Інформація про місцеположення кожної з частин розподіленої БД і інша службова інформація зберігається в так званому *глобальному словнику даних.* В загальному випадку цей словник може зберігатися на одному з вузлів або теж бути розподіленим.

Для забезпечення коректного доступу до розподіленої БД в сучасних системах частіше за все застосовується протокол (метод) *двофазної фіксації транзакцій* (two-phasecommit). Сутьцього методу полягає в двох етапній синхронізації виконуваних змін на всіх задіяних вузлах. На першому етапі у вузлах мережі проводяться зміни (поки оборотні) в їх БД, про що посилаються повідомлення компоненту системи, керівнику обробкою розподілених транзакцій.

На другому етапі, отримавши від всіх вузлів повідомлення про правильність виконання операцій (що свідчить про відсутність збоїв і відмов апаратно-програмного забезпечення), управляючий компонент видає всім вузлам команду фіксації змін. Після цього транзакція вважається завершеною, а її результат необоротним.

Основною *перевагою* моделі розподіленої БД є те, що користувачі всіх вузлів (при справних комунікаційних засобах) одержують

інформацію з урахуванням всіх останніх змін. Друга перевага полягає в економному використанні зовнішньої пам'яті комп'ютерів, що дозволяє організовувати БД великих об'ємів.

До *недоліків* моделі розподіленої БД відноситься наступне: жорсткі вимоги до продуктивності і надійності каналів зв'язку, а також великі витрати комунікаційних і обчислювальних ресурсів через їх скріплення на весь час виконання транзакцій. При інтенсивних зверненнях до розподіленої БД, великому числі взаємодіючих вузлів, низькошвидкісних і ненадійних каналах зв'язку обробка запитів по цій схемі стає практично неможливою.

Модель ***тиражування даних,*** на відміну від технології розподілених БД, припускає дублювання даних (створення точних копій) у вузлах мережі (рисунок21.7).

Дані завжди обробляються як звичайні локальні. Підтримку ідентичності копій один одному в асинхронному режимі забезпечує компонент системи, званий *реплікатором* (replicator). При цьому між вузлами мережі можуть передаватися як окремі зміни, так і групи змін. Протягом деякого часу копії БД можуть відрізнятися один від одного.

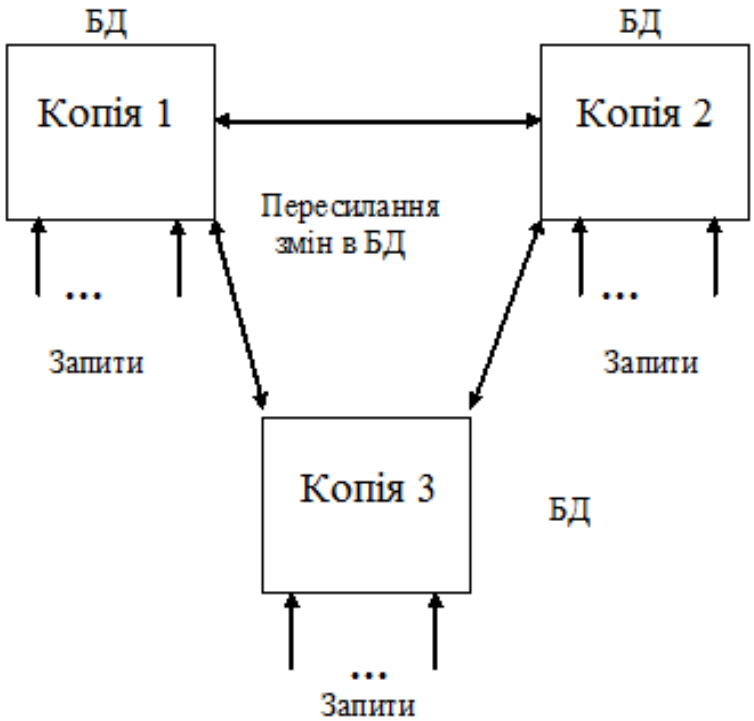


Рисунок 21.7 - Модель тиражування БД

До основних *переваг* моделі тиражування БД (порівняно з попередньою моделлю) відносяться: більш висока швидкість доступу

до даних, оскільки вони завжди є у вузлі"; істотне зменшення передаваного по каналах зв'язку потоку інформації, оскільки відбувається передача не всіх операцій доступу до даних, а тільки змін в БД; підвищення надійності механізмів доступу до розподілених даних, оскільки порушення зв'язку не приводить до втрати працездатності системи (передбачається буферизація потоку змін, що дозволяє коректно відновити роботу після відновлення зв'язку).

Основний *недолік* моделі тиражування БД полягає в тому, що на деякому інтервалі часу можлива «розбіжність» копій БД.

Контрольні запитання:

1. Що називають локальною мережею?
2. Що називають глобальною мережею?
3. Скільки функціональних рівнів мережного програмного забезпечення визначено міжнародною організацією ISO? (Перерахуйте їх).
4. Які основні методи управління обчислювальними мережами?
5. Перерахуйте дволанцюгові моделі розподілу функцій в архітектурі клієнт-сервер.
6. Перерахуйте переваги та недоліки централізованих і децентралізованих мереж.
7. Які основні функції в мережі виконує СКБД при роботі з базами даних?
8. Назвіть основні переваги і недоліки моделі віддаленого доступу до даних (RDA).
9. Охарактерезуйте модель серверу бази даних (DBS) її переваги і недоліки.
10. Що уявляє собою модель розподіленого представлення (розподіленої функції та розподіленої бази даних).
11. Охарактеризуйте трьох ланцюгову модель розподілу функцій в архітектурі клієнт-сервер (AS-модель).
12. Що є центральною ланкою AS- моделі?
13. Назвіть переваги та недоліки моделі серверу додатків (AS-моделі)
14. Взаємодію яких трьох елементів описує модель серверу додатків?
15. Для подолання якого недоліку AS – моделі запропонована модель монітора транзакцій (TPM)
16. Яке основне призначення моніторів обробки транзакцій?

17. Дайте визначення поняття транзакції в моделі монітора транзакцій.
18. Що являє собою розподілена база даних?
19. Назвіть основні переваги та недоліки моделі розподіленої бази даних.
20. Охарактеризуйте модель тиражування даних.
21. Назвіть типові варіанти розділення функцій між комп'ютером-сервером і комп'ютером-клієнтом для дволанцюгової моделі.
22. Охарактеризуйте модель віддаленого доступу до даних.
23. Назвіть переваги і недоліки моделі серверу баз даних.
24. Зобразіть структурну схему триланцюгової моделі серверу додатків.
25. Яке призначення моніторів транзакцій?
26. Зобразіть схему і охарактеризуйте модель монітора транзакцій.
27. Назвіть основні технології децентралізованого управління БД.
28. Опишіть динаміку функціонування моделі розподіленої БД.
29. Вкажіть переваги і недоліки моделі розподіленої БД.
30. Поясніть суть методу двофазної фіксації транзакцій.

Тестові завдання:

1. Що називають обчислювальною мережею?
а) Діаграма проходження
б) Сукупність комп'ютерів або робочих станцій (PC), з'єднаних засобами передачі даних
в) Тимчасова динамічна таблиця з даними
г) Правильної відповіді немає

2. Що може включати в себе глобальна мережа?
а) Форми і звіти різних баз даних
б) Сукупність таблиць або запитів, з'єднаних засобами передачі даних
в) Інші глобальні мережі, локальні мережі, а також окремі комп'ютери, що підключаються до неї або пристрої введення-виведення
г) Правильної відповіді немає

3. Які існують види глобальних мереж?

а) Міські, регіональні, національні і транснаціональні
б) Одинарні, подвійні, багатофункціональні
в) Динамічні, статичні, поєднані
г) Правильної відповіді немає

4.Скільки функціональних рівнів мережі визначає еталонна модель OSI?
а) Два
б) Сім
в) П’ять
г) Десять

5. Яку функцію в локальних мережах виконує сервер?
а) Транснаціональну
б) Управління даними бази даних
в) Розподілу мережних ресурсів
г) Правильної відповіді немає

6. Які існують моделі розподілу функцій в архітектурі клієнт-сервер?
а) Транснаціональна, тринаціональна
б) Концептуальна, нерозподіленого доступу
в) Дволанцюгові, розподіленої функції, модель розподіленого представлення, розподіленої бази даних, віддаленого доступу
г) Правильної відповіді немає

7. Які основні компоненти включає в себе триланцюгова модель розподілу функцій?
а) Сервер баз даних, сервер додатків, клієнт
б) Сукупність 3-х форм, з'єднаних засобами передачі даних
в) Три моделі монітора транзакцій
г) Правильної відповіді немає

8. Яке основне призначення комп’ютеру-серверу в системах клієнт-сервер?
а) Зберігання файлів і управління передачею даних в мережі
б) Видалення сукупностіфайлів, з'єднаних засобами передачі даних
в) Моніторинг транзакцій

г) Правильної відповіді немає

9.До якого рівня еталонної маделі (OSI) відноситься функція управління БД?
а) Рівня транзакцій
б) Транспортного рівня
в) Прикладного рівня
г) Сеансового рівня

10.Між кількома вузлами розподіляється функція СКБД в дволанцюгових моделях?
а) Між двома
б) Між трьома
в) Між чотирма
г) Правильної відповіді немає

11.Що є центральною ланкою AS-моделі?
а) Сервер бази даних
б) Сервер додатку
в) Комп’ютер клієнт
г) Правильної відповіді немає

12.Взаємодія яких трьох елементів описує модель серверу додатків?
а) Клієнта, бази даних і схеми даних
б) Серверу, монітору, клавіатури
в) Клієнта, серверу додатків і серверу бази даних
г) Правильної відповіді немає

Розділ 22

Доступ до загальних даних

- **Монопольний та колективний доступ до даних**
- **Механізм блокувань**
- **Обробка тупікових ситуацій**
- **Збережені процедури і тригери**
- **Стандартний інтерфейс відкритих систем ODBC**
- **Інформаційні системи в Intranet та Internet**

При обслуговуванні звернень до загальних даних засоби управління БД повинні забезпечувати принаймні два основні методи доступу: монопольний і колективний. Основними об'єктами доступу в різних системах можуть бути цілком БД, окремі таблиці, записи, поля записів. В СКБД, що надають можливість розробки, об'єктами доступу також можуть виступати специфікації звітів і екранних форм, запити і програми.

Монопольний доступ звичайно використовується в двох випадках:

• по-перше, коли вимагається виключити доступ до об'єктів з боку іншихкористувачів (наприклад, при роботі з конфіденційною інформацією);

• по-друге, коли проводяться *відповідальні* операції з БД, що не допускають інших дій, наприклад, зміна структури БД.

В першому випадку користувач за допомогою діалогових засобів СКБД або прикладної програми встановлює явне *блокування.* В другому випадку користувач теж може встановити явне блокування, або покластися на СКБД. Остання звичайно автоматично встановлює неявне (без відома користувача або додатку) блокування, якщо це необхідне.

В режимі *колективного доступу* повне блокування на об'єкти, що використовуються, як правило, не встановлюється. Колективний доступ можливий, наприклад, при одночасному перегляді таблиць. Спроби отримати монопольний доступ до об'єктів колективного доступу повинні бути присічені. Наприклад, в ситуації коли один або декілька користувачів проглядають таблицю, а інший користувач збирається видалити цю ж таблицю.

Для організації колективного доступу в СКБД застосовується *механізм блокувань*.Суть блокування полягає в тому, що на час

виконання якої-небудь операції в БД доступ до об'єкту, що використовується, з боку інших споживачів тимчасово забороняється або обмежується. Наприклад, при копіюванні таблиці вона блокується від зміни, хоча і дозволено проглядати її вміст.

Розглянемо деякий типовий набір блокувань. В конкретних програмах схеми блокування об'єктів можуть відрізнятися від описуваної. Виділимо чотири види блокувань, перераховані в порядку убування строгості обмежень на можливі дії:

- повне блокування;
- блокування від запису;
- оберігаюче блокування від запису;
- оберігаюче повне блокування.

Повне блокування. Означає повну заборону всяких операцій над основними об'єктами (таблицями, звітами і екранними формами). Цей вид блокування звичайно застосовується при зміні структури таблиці.

Блокування від запису. Накладається у випадках, коли можна використовувати таблицю, але без зміни її структури або вмісту. Таке блокування застосовується, наприклад, при виконанні операції злиття даних з двох таблиць.

Оберігаюче блокування від запису. Оберігає об'єкт від накладення на нього з боку інших операцій повного блокування, або блокування від запису. Цей вид блокування дозволяє тому, хто раніше «захопив» об'єкт, успішно завершити модифікацію об'єкту. Оберігаюче блокування від запису сумісне з аналогічним блокуванням (оберігаючим блокуванням від запису), а також з оберігаючим повним блокуванням (див. далі). Прикладом необхідності використання цього блокування є режим сумісного редагування таблиці декількома користувачами.

Оберігаюче повне блокування. Оберігає об'єкт від накладення на нього з боку інших операцій тільки повного блокування. Забезпечує максимальний рівень сумісного використання об'єктів. Таке блокування може використовуватися, наприклад, для забезпечення одночасного перегляду декількома користувачами однієї таблиці. В групі користувачів, що працюють з однією таблицею, це блокування не дозволить нікому змінити структуру загальної таблиці.

При незавершеній операції з деяким об'єктом в запиті на виконання нової операції з цим же об'єктом проводиться спроба ці операції

сумістити. Поєднання можливо тоді, коли сумісними виявляються блокування, що накладаються конкуруючими операціями.

Відносно перерахованих вище чотирьох блокувань діють наступні *правила поєднання:*

• за наявності повного блокування над об'єктом не можна проводити операції, що приводять хоча б до одного з видів блокувань (повне блокування несумісне ні з яким іншим блокуванням);

• блокування від запису сумісне з аналогічним блокуванням і оберігаючих повним блокуванням;

• оберігаюче блокування від запису сумісне з обома видами оберігаючих блокувань;

• оберігаюче повне блокування сумісне зі всіма блокуваннями, окрім повного.

Звичайно в СКБД кожній з виконуваних з БД операцій відповідає певний вид блокування, яке ця операція накладає на об'єкт. Користувачам сучасних СКБД, що працюють в інтерактивному режимі, не потрібно пам'ятати всю тонкість механізму блокування, оскільки система достатньо «розумно» здійснює автоматичне блокування у всіх випадках, коли це потрібне. При цьому система сама прагне надати всім користувачам найвільніший доступ до об'єктів. При необхідності користувач і програміст може скористатися командними або мовними засобами явного визначення блокувань. Наприклад, в СКБД Paradox для явного блокування окремого запису під час редагування таблиці використовується команда Record Lock.

Тупіки

Якщо не управляти доступом до об'єктів, що спільно використовуються, то між споживачами ресурсів можуть виникати тупікові ситуації (клінчі, «смертельні обійми» або блокування). Слід відрізняти поняття блокування в значенні контролю доступу до об'єктів (ми дотримуємося такого терміну) від блокування в значенні тупікової події.

Існує два основні види тупіків: *взаємні*(deadlock) і *односторонні* (livelock).

Найпростішим випадком *взаємного тупика*є ситуація, коли кожний з двох користувачів прагне захопити дані, вже захоплені іншим користувачем (рисунок22.1а). В цій ситуації користувач-1 чекає звільнення ресурсу N, тоді як користувач-2 чекає звільнення від захоплення ресурсу М. Отже, ніхто з них не може продовжити роботу.

Насправді можуть виникати і складніші ситуації, коли виконується обіг трьох і більш користувачів до декількох ресурсів. Приклад однієї з таких ситуацій наведений на рисунку 22.1б.

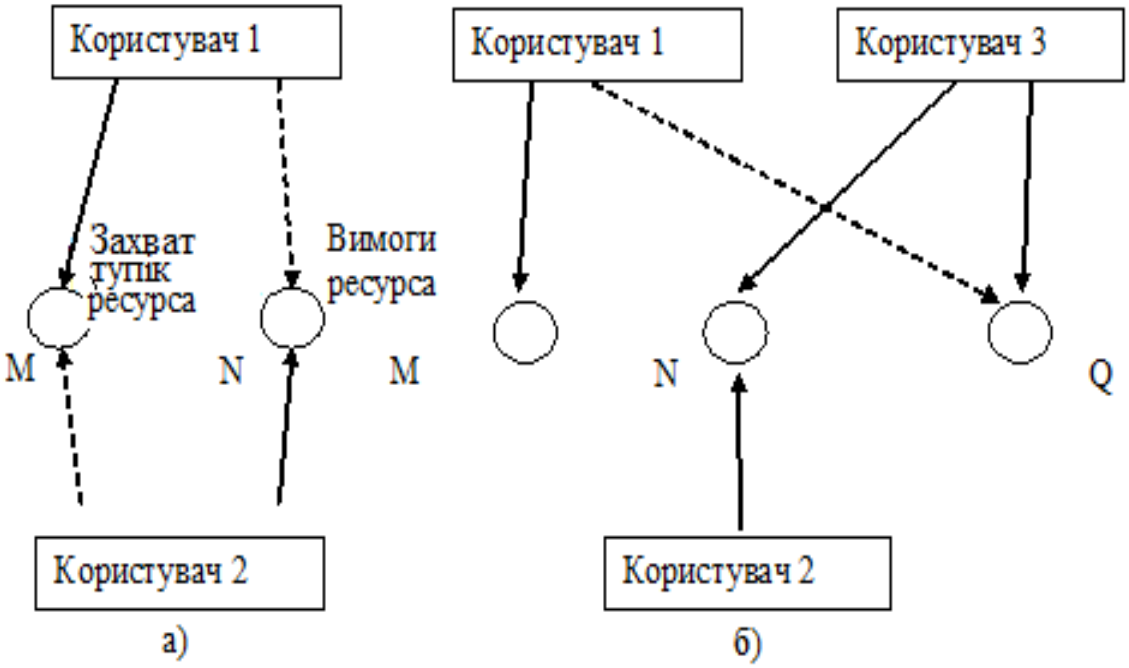


Рисунок 22.1 - Приклади взаємних тупіків в розподілених БД

Односторонній тупік виникає у разі вимоги отримати монопольний доступ до деякого ресурсу як тільки він стане доступним і неможливості задовольнити цю вимогу.

Системи управління розподіленими БД, очевидно, повинні мати відповідні засоби виявлення або запобігання конфліктів, а також дозволи виникаючих конфліктів. Однієї з найскладніших є задача усунення конфліктів в неоднорідних системах у випадку, якщо деяка програма не обробляє або обробляє некоректно сигнали (повідомлення) про наявність конфліктів. При цьому важливо не тільки зберегти цілісність і достовірність даних в розподілених БД, але і відновити обчислювальний процес, іноді це паралізує користувачів і програми очікують чогось. Користувачі і розробники додатків в розподіленому середовищі повинні передбачати обробку сигналів про тупіки.

Інформаційні системи в локальних мережах

Локальна обчислювальна мережа дає користувачу перш за все можливість більш ефективної організації групових робіт і сумісного використання апаратних ресурсів: принтерів, факсів, модемів, сканерів, дисків і т. д., а також програмно-інформаційних ресурсів, у тому числі баз даних. Розглянемо основні *схеми організації роботи* з *даними* в

ЛОМ. Спектр можливих схем побудови інформаційної системи в ЛОМ істотно залежить від можливостей МОС, що використовуються. Основним сервісом сучасних ЛОМ, незалежно від типу управління в них (централізовані або однорангові), є надання доступу одного комп'ютера (комп'ютера-клієнта) до дисків, каталогів і файлів іншого комп'ютера (комп'ютера-серверу). Так, при зверненні до зовнішнього файлу МОС спочатку передає по мережі файл (частину файла) в оперативну пам'ять комп'ютера, а після закінчення роботи з ним при необхідності повертає оновлену версію. Крім того, корисною функцією є можливість запуску на комп'ютері програм, що зберігаються на іншому комп'ютері.

Більш слабкі мережні пакети і утиліти можуть надавати доступ до інформації без автоматичного запуску програм. В цьому випадку користувач повинен спочатку скопіювати програми і файли з іншого комп'ютера на свій, після чого запустити програму. Як і на окремому комп'ютері, в ЛОМ користувач може управляти БД за допомогою засобів деякої СКБД або працюючи з додатком. В загальному випадку додаток може виконуватися під управлінням СКБД або її ядра, або бути незалежним і виконуватися як автономна програма. Для зручності надалі під СКБД розумітимемо будь-який з цих варіантів.

В локальній мережі комп’ютерів виділяють наступні три варіанти створення інформаційної системи:

- типу файл-сервер;
- типу клієнт-сервер;
- засновані на технології Intranet;

Інформаційні системи типу *файл-сервер* можна будувати двома способами:

- з використанням немережних СКБД, призначених для застосування на віддаленій машині;
- з використанням *мережних* СКБД, що розробляються для застосування в ЛОМ.

Під мережною *СКБД тут розуміється система з довільною моделлю даних (*необов'язково мережна), орієнтована на використання в мережі. Програми *немережної СКБД* і дані, що використовуються нею можуть зберігатися на комп'ютері-сервері (КС) і на комп'ютері-клієнті (КК). Запуск і функціонування немережної СКБД, що зберігається на КК і працюючої з локальними даними не відрізняється від звичайного режиму роботи на окремій ПЕВМ. Якщо дані, що використовуються,

зберігаються на КС, файлова система мережної ОС «непомітно для» СКБД виконує підвантаження потрібного файлу з віддаленого комп'ютера. Помітимо, що не кожна немережна СКБД без проблем працює в середовищі будь-якої мережної ОС. Якщо немережна СКБД використовується декількома користувачами мережі, то її програми, а також БД або її частину в цілях економії дискової пам'яті ефективно берегти на КС. Збережену на КС БД називатимемо *центральною БД* (ЦБД), а збережену на КК БД - *локальною БД* (ЛБД). При запуску СКБД в такому варіанті на кожний КК звичайно пересилається повна копія основної програми СКБД і один або декілька файлів ЦБД (рисунок22.2). Після завершення роботи файли ЦБД повинні пересилатися з КК назад на КС для узгодження даних.

Істотним *недоліком* такого варіанту застосування немережних СКБД є можливість порушення цілісності даних при одночасній роботі з однією БД декількох користувачів. Оскільки кожна копія СКБД функціонує не «знаючи» про роботу інших копій СКБД, то ніяких заходів по виключенню можливих конфліктів не приймається. При цьому елементарні операції читання-запису одних і тих же файлів, як правило, контролює мережна ОС.

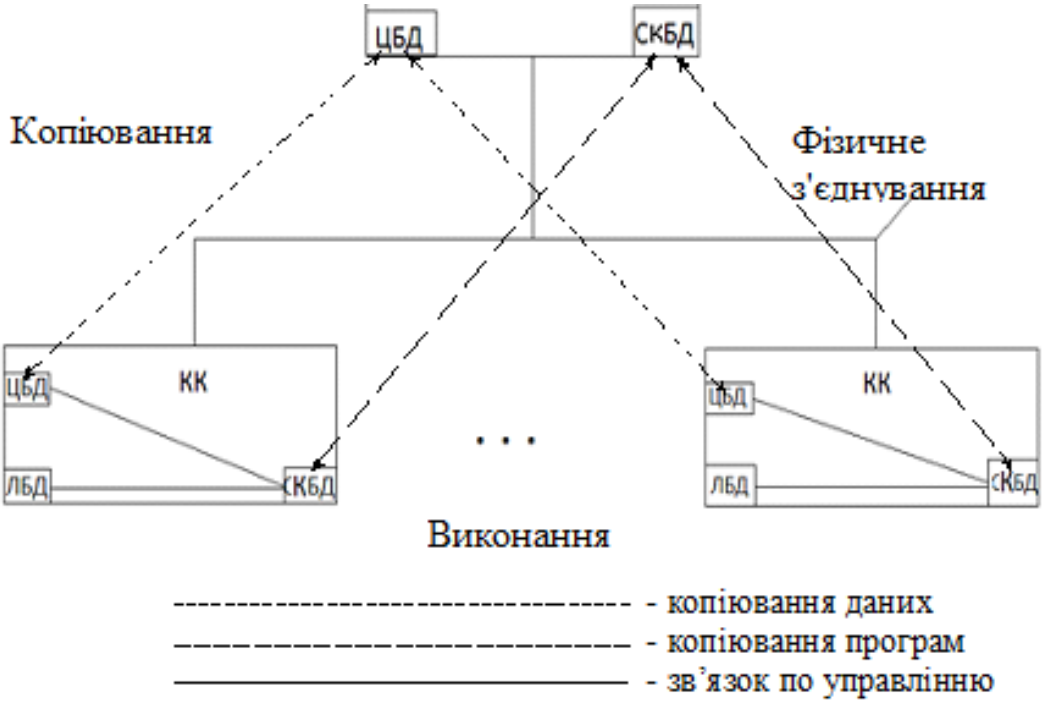


Рисунок 22.2 - Система типу файл-сервер з немережною СКБД

Як приклади немережних СКБД можна назвати перші версії системи dBase III Plus, dBase IY і FoxBase.

Мережні СКБД не мають вказаного недоліку, оскільки в них передбачається «контроль суперництва» (concurrency control). Засоби

контролю дозволяють здійснювати координацію доступу до даних, наприклад, введенням блокувань до файлів, записів і навіть окремих полів записів (рисунок22.3).

В мережних СКБД з колективним використанням файлів БД як і раніше вся обробка інформації проводиться на КК, а функції КС зводяться до надання великої дискової пам'яті. Такий підхід не можна вважати ефективним, оскільки для забезпечення прийнятної швидкості процесу обробки інформації КК повинен володіти високою швидкодією і мати велику місткість оперативної пам'яті. Крім того, пересилка копій файлів БД і команд управління блокуваннями по лініях зв'язку істотно збільшує навантаження на підсистему передачі даних, що знижує загальну продуктивність мережі.

Прикладами мережних СКБД є: FoxPro 2.5 для Windows, dBase IY, Paradox 3.5 для DOS.

Інформаційні системи типу *клієнт-сервер* відрізняються від систем типу файл-сервер перш за все тим, що програми СКБД функціонально розділені на дві частини, звані *сервером* і *клієнтом.* Між клієнтською і серверною частинами системи можливі різні варіанти розподілу функцій. *Клієнт*, або *фронтальна програма*, відповідає за інтерфейс з користувачем, для чого перетворить його запити в команди запитів до серверної частини, а при отриманні результатів виконує зворотне перетворення і відображення інформації для користувача.

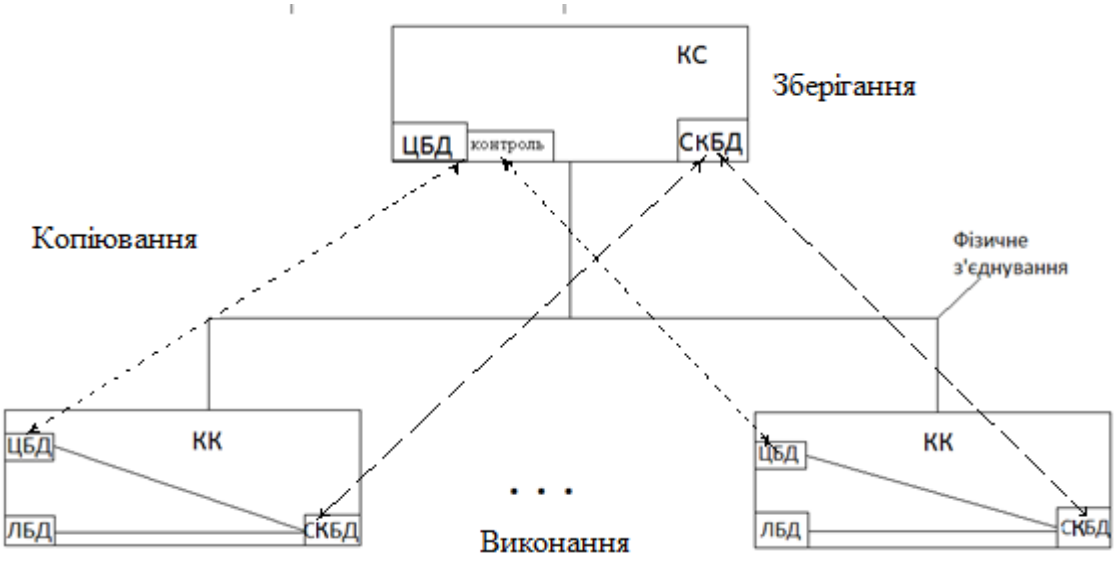


Рисунок 22.3 - Система типу файл-сервер з мережною СКБД

В ролі клієнта виступає призначена (програма що розробляється для вирішення конкретної прикладної задачі) для користувача або готова

програма, що має інтерфейс з серверною програмою. Як готові клієнтські програми можуть використовуватися текстові процесори, табличні процесори і навіть СКБД (наприклад, Access, FoxPro і Paradox).

Сервер є основною програмою, виконуючою функції управління і захисту даних в базі. У випадках, коли виклик функцій серверу виконується на мові SQL, а саме так часто і відбувається, його називають *SQL-сервером.*

Як сервер може використовуватися ядро професійної реляційної СКБД (наприклад, Informix 7.x і Sybase System 10) або деякий SQL-сервер (наприклад, Novell NetWare SQL і Microsoft SQL Server). Структуру інформаційних систем типу клієнт-сервер спрощений можна представити як показано на рисунку22.4.

Основна частина обробки інформації по формуванню запитів, складанню звітів, представленню даних в зручному для користувача вигляді і т.д. виконується на КК. Повні копії файлів БД з КС на КК і назад (як у разі систем на базі мережних СКБД) не пересилаються, оскільки для організації повноцінної взаємодії, як правило, достатньо мати на КК необхідні в даний момент часу записи БД. Все це істотно знижує трафік в мережі, ослабляє вимоги по ресурсах до КК, дозволяє створювати більш ефективні і надійні інформаційні системи.

Останнім часом на комп'ютері-сервері, окрім власне даних, зберігають програми обробки даних і запити. Це забезпечує збільшення швидкості обробки даних (програма обробки або запит знаходиться поряд з даними), а також ефективність зберігання і адміністрування програм і запитів загального користування в одному місці (на комп’ютер-сервері).

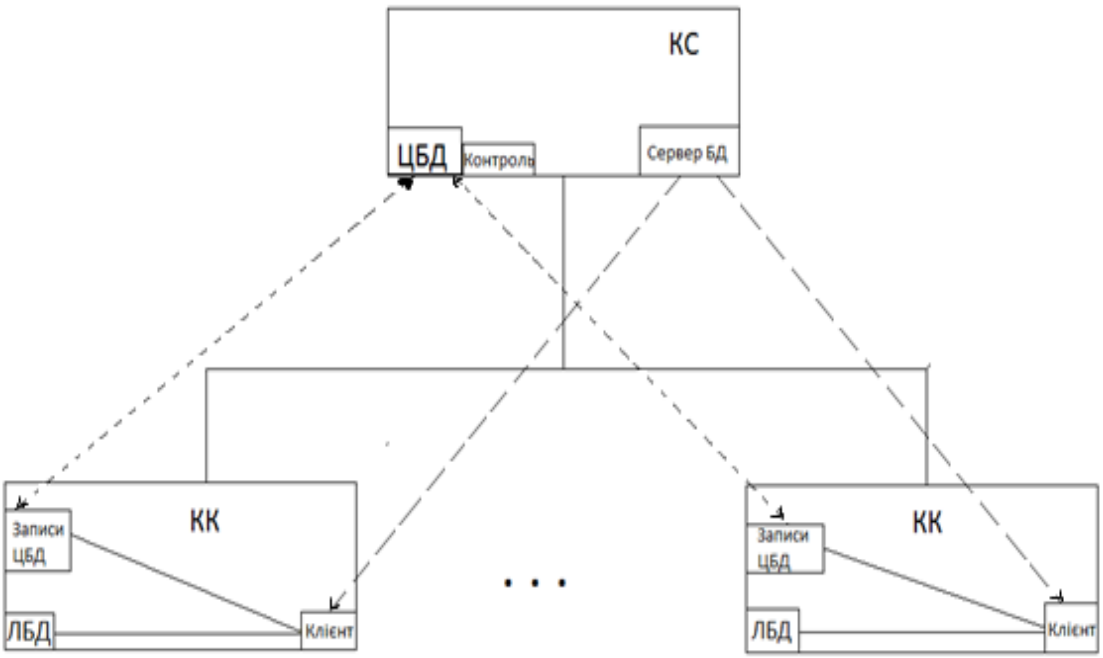


Рисунок 22.4 - Інформаційна система типу клієнт-сервер

Збережені на комп'ютері-сервері програми (процедури) обробки даних називають *збереженими процедурами*. Різновидом збереженої процедури є так званий ***тригер.*** Тригер (процедура тригера) автоматично викликається при виникненні певних подій в БД. Як події можуть бути наступні: операції вставки, оновлення і видалення окремих записів, колонок і полів записів і інші. Прикладом тригера є програма, що запускає процес посилки повідомлення по електронній пошті досягши розміру БД (кількості записів) граничного значення.

В БД серверу деяких систем можна берегти і самі запити, звані *збереженими командами*. Сукупність збережених команд — це проіменована сукупність команд, одержаних в результаті компіляції SQL-запиту. Збережені команди виконуються значно швидше, ніж відповідний SQL-запит. Основна причина прискорення полягає в тому, що при виконанні збережених команд не потрібен синтаксичний розбір запитів. Додаткове прискорення виконання запитів може бути отримано в тих випадках, коли сервер БД не просто зберігає коди команд запитів, а проводить оптимізацію коду, що зберігається.

Збереженими процедурами і командами пов'язано поняття *курсора*, відмінне від звичного поняття курсора як покажчика поточної позиції на екрані монітора. В різних СКБД — це близькі, але дещо відмінні поняття. Найбільш широко це поняття трактується в СКБД SQLBase. Тут курсор може означати наступне:

• ідентифікатор сеансу зв'язку користувача з СКБД;

• ідентифікатор збережених команд і процедур;

• ідентифікатор результуючої множини;

• покажчик поточного рядка в результуючій множині, оброблюваній клієнтським додатком.

Програми серверу (основні, збережені процедури і тригери) можуть бути виконані як звичайні програми (Windows 95/98), або як спеціально завантажувані модулі мережної Ос (NLM-модулі в мережі Novell). Програми клієнта в загальному випадку зберігаються на КС або на КК, або в обох місцях.

В даний час серед програмних продуктів існує величезна кількість універсальних (в значенні придатності роботи з різними серверами БД) засобів розробки систем типу клієнт-сервер, до числа яких відносяться: Delphi (Borland), Power Builder (Powersoft), ERwin (LogicWorks), Visual Basic (Microsoft), CA-Visual Objects (Computer Associates), SQL Windows (Gupta) та інші. Крім того, існують засоби розробки в рамках певних СКБД (наприклад, для Oracle 7 — Designer/2000). Всі подібні засоби, як правило, відносяться до CASE-систем .

При побудові інформаційних систем типу клієнт-сервер виникає проблема доступу з боку СКБД або додатків, розроблених в одному середовищі, до даних, породжених іншою СКБД. В середовищі Windows ця проблема розв'язується за допомогою стандартного інтерфейсу ODBC(Open Database Connectivity — сумісність відкритих баз даних) фірми Microsoft. Основне його призначення полягає в забезпеченні уніфікованого доступу до локальних і видалених баз даних різних виробників.

Схема доступу додатків до баз даних за допомогою ODBC показана на мал. 4.12. Доступ додатку до даних відбувається шляхом виклику на мові SQL стандартних функцій інтерфейсу ODBC. На комп'ютері-клієнті при цьому повинна функціонувати операційна система MS Windows з інтерфейсом ODBC.

Інформаційні системи в Internet і Intranet

Обробка інформації в середовищі Internet істотно відрізняється від обробки *інформації* в локальній мережі і, тим більше, на окремому комп'ютері. Перерахуємо найважливіші з них:

1.Велика протяжність комунікаційних ліній, що позначається на тимчасових характеристиках обміну. Крім того, велика віддаленість позбавляє значення завантаження програм з одного комп'ютера на

іншій і не дозволяє виконувати пересилку великих об'ємів даних в реальному масштабі часу, як в мережних СКБД локальних мереж.

2.Взаємодія розподілених елементів ІС відбувається за допомогою обміну пакетами або повідомленнями. *Окремі програмні компоненти ІС можуть бути* одного або різних виробників. В останньому випадку особливу роль відіграє рішення проблеми підтримки стандартів на мережні протоколи і на мову SQL.

3.Мережу Internet відрізняє від решти глобальних мереж те, що по масштабах вона більше всіх інших мереж об'єднує мережі і принципи її організації роблять істотний вплив на використання в мережі баз даних.

Характеристика Internet

Основними видами послуг (сервісу), що надаються користувачам при підключенні до Internet, є:

- електронна пошта (E-mail);
- телеконференції (UseNet);
- система емуляції видалених терміналів (TelNet);
- пошук і передача двійкових файлів (FTP);
- пошук і передача текстових файлів за допомогою системи меню (Gopher);
- пошук і передача документів за допомогою гіпертекстових посилань(WWW).

Створення і розвиток цих способів пов'язано з історією Internet. Кожний з них характеризується своїми можливостями і відмінністю в організації протоколів обміну інформацією. Під протоколом, в загальному випадку, розуміється набір інструкцій, що регламентують роботу взаємозв'язаних систем або об'єктів в мережі.

Електронна пошта (E-mail) — найпростіший і доступний спосіб доступу в мережі Internet. Дозволяє виконувати пересилку будь-яких типів файлів (включаючи тексти, зображення, звукові файли) за адресами електронної пошти в будь-яку точку планети *за короткий* проміжок часу у будь-який час доби. Для передачі повідомлення необхідно знати електронну адресу одержувача. Робота електронної пошти заснована на послідовній передачі інформації по мережі від одного поштового серверу до іншого, поки повідомлення не досягне адресата. До переваг електронної пошти відносяться висока оперативність і низька вартість. Недолік електронної пошти полягає в обмеженості об'єму файлів, що пересилаються.

Система телеконференцій UseNet розроблена як система обміну текстовою інформацією. Вона дозволяє всім користувачам Internet брати участь в групових дискусіях, званих телеконференціями, в яких обговорюються всілякі проблеми. Зараз в світі налічується більше 10 тисяч телеконференцій. Інформація, яка посилається в телеконференції, стає доступною будь-якому користувачу Internet, що звернувся в дану телеконференцію. В даний час телеконференції дозволяють передавати файли будь-яких типів. Для роботи з телеконференціями найбільш часто використовуються засоби програм перегляду і редагування Web-документів.

TelNet - це протокол, що дозволяє одному комп'ютеру використовувати ресурси іншого (віддаленого) комп'ютера. Іншими словами — це протокол віддаленого термінального доступу в мережі.

FTP (File Transfer Protocol) — це протокол, що дозволяє передавати файли довільного формату між двома комп'ютерами мережі. Програмне забезпечення FTP розроблено по архітектурі клієнт-сервер і розділено на дві частини: серверну (FTP-сервер) і клієнтську. FTP-клієнт, в загальному випадку, дозволяє користувачам проглядати файлову систему FTP-серверу і проводити з нею обмін файлами (вивантажувати файли свого комп'ютера, завантажувати, перейменовувати і видаляти файли віддаленого комп'ютера). Перевагою даного протоколу є можливість передачі файлів будь-якого типу, у тому числі виконуваних програм. До недоліку протоколу FTP слід віднести необхідність апріорного знання місцеположення відшукуваної інформації (FTP-адреси).

Протокол **Gopher** іреалізовуюче його програмне забезпечення надають користувачам можливість роботи з інформаційними ресурсами, не знаючи наперед їх місцезнаходження. Для початку роботи по цьому протоколу достатньо знати адресу одного Gopher-серверу. Надалі робота полягає у виборі команд, представлених у вигляді простих і зрозумілих меню. При цьому пункти меню одного серверу можуть містити посилання на меню інших серверів, що полегшує пошук необхідної інформації в мережі Internet. Під час роботи з системою Gopher програма-клієнт не підтримує постійного з'єднання з Gopher-сервером, що дозволяє економити мережні ресурси.

WWW (World Wide Web — всесвітня павутина) є найпопулярнішим і сучасним засобом організації мережних ресурсів. Вона будується на основі гіпертекстового представлення інформації.

Гіпертекстовий документ (гіпертекст) є текстом, що містить посилання на інші фрагменти текстів довільних документів, у тому числі і цього документа. Гіпертекстовий документ готується на стандартизованій мові HTML (Hypertext Markup Language — мова розмітки гіпертексту). Він складається із сторінок (web-сторінок), доступ до яких заснований на протоколі передачі гіпертексту (Hypertext Transfer Prococol, HTTP).

HTML-документ є ASCII-файлом, доступним для перегляду і редагування в будь-якому редакторі текстів. На відміну від звичайного текстового файлу, в ньому присутні спеціальні команди — *теги,* які вказують правила форматування документа. За допомогою тегів описуються різні елементи документа: заголовки, абзаци (параграфи), списки, посилання, форми і т.д. Найпростішим прикладом гіпертексту є книга, зміст якої містить посилання (внутрішні) у вигляді номерів сторінок на розділи, підрозділи, пункти книги, крім того, в книзі є зовнішні посилання на інші джерела інформації, що використовуються. Фрагмент документа може включати інформацію у вигляді звичайного тексту, графічного зображення, звуку і зображення, що рухається (анімації). Гіпертекст з нетекстовими документами часто називають **гіпермедіа.** Найважливішою властивістю гіпертексту є наявність в ньому посилань на документи, розміщувані на територіально віддалених комп'ютерах. Документи можуть створюватися і редагуватися різними людьми. Вся сукупність взаємозв'язаних документів утворює гігантську «павутину». Ця модель подібна моделі оточуючого нас нескінченного інформаційного простору, коли немає жорсткої ієрархії зв'язків, а є безліч зв'язків без початку і кінця.

Робота мережі Internet заснована на використанні протоколу TCP/IP (Transmission Control Protocol/Internet Protocol — Протокол управління передачею даних/Протокол Internet), який використовується для передачі даних в глобальній мережі і в багатьох локальних мережах. TCP/IP в основному реалізує функції транспортного і мережного рівнів моделі OS1. Він є сімейством комунікаційних протоколів, які за призначенням можна розділити на наступні групи:

- транспортні протоколи, що служать для управління передачею даних між двома комп'ютерами;
- протоколи маршрутизації, оброблювальні адресацію даних і визначаючі найкоротші доступні шляхи до адресата;

• протоколи підтримки мережної адреси, призначені для ідентифікації комп'ютера по його унікальному номеру або імені;

• прикладні протоколи, що забезпечують отримання доступу до всіляких мережних послуг;

• шлюзові протоколи, що допомагають передавати по мережі повідомлення про маршрутизацію і інформацію про стан мережі, а також обробляти дані для локальних мереж;

• інші протоколи, що не відносяться до вказаних категорій, але забезпечуючі клієнту зручність роботи в мережі.

Доступ користувачів до ресурсів Internet звичайно проводиться за допомогою програм-навігаторів, або ***браузерів*** (від англ. browser). В даний час до числа найпопулярніших програм цього класу відносяться наступні: GoogleChrome, MozillaFirefox, Opera, InternrtExplorer. Хоча ці програми засновані на використанні протоколу HTTP, вони надають простий доступ до інших сервісів Internet: електронній пошті, новинам і т.д.

Браузер, забезпечуючи доступ користувача до ресурсів мережі, по суті є програмою-клієнтом (або Web-клієнтом). Програмою, що надає інформаційні ресурси, є Web-сервер. Саме він здійснює основну роботу по збору і отриманню інформації з різних джерел, після чого в стандартному вигляді надає її Web-клієнту. Розглянемо організацію вибору інформації для користувача, якщо вона знаходиться в базах даних.

Бази даних в Internet і Intranet

Технологія Intranet по суті є технологією Internet, перенесеною в середу корпоративних ІС. Архітектура інформаційних систем в Internet і Intranet є результатом еволюційного переходу від перших розрахованих на багато користувачів централізованих обчислювальних систем (мейнфреймів) через системи типу клієнт-сервер до розподілених систем з централізованою обробкою і підготовкоюінформації до безпосереднього споживання. Розглянемо стисло вказані етапи еволюції.

1.В *мейнфреймах* (рисунок 22.5) обчислювальні ресурси, збережені дані і програми обробки інформації сконцентровані в одній ЕОМ. Основним засобом доступу був алфавітно-цифровий термінал (дисплей), керований ЕОМ. Вся обробка інформації і підготовка її до видачі виконувалися на центральній ЕОМ. З терміналів, як правило, в машину передавалися коди натиснення клавіш або вміст буфера екрану,

а назад на термінал — пересилають відображення з відповідними кодами управління відображенням, що відображаються.

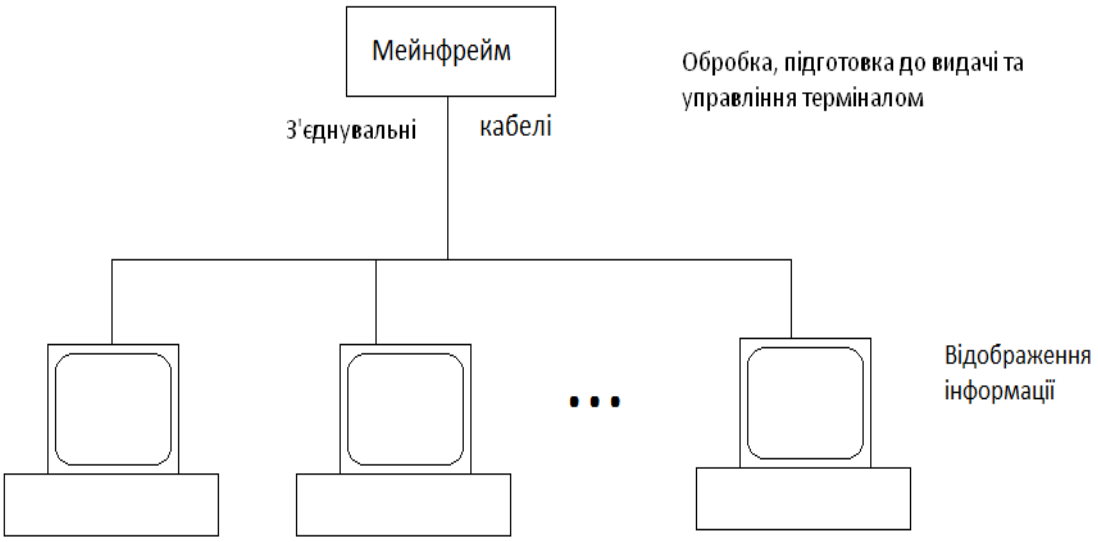


Рисунок 22.5 - Централізована розрахована на багато користувачів система

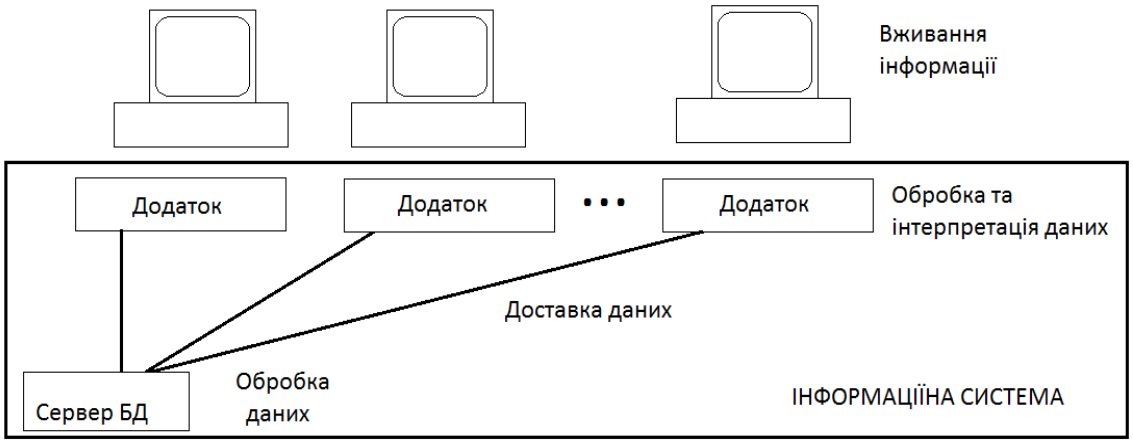


Рисунок 22.6 - Системи типу клієнт-сервер

Перевагою системи є простота адміністрування, захисту інформації і модифікації системи, до *недоліків* можна віднести високе завантаження процесорів і ліній зв'язку (як наслідок - невисоку реакцію системи при великій кількості користувачів), низьку надійність (вихід з ладу ЕОМ приводить до повної відмови всієї системи), складність масштабування системи і деякі інші.

2. Історично наступним рішенням в області інформаційних систем була архітектура *клієнт-сервер* (рисунок 22.6).

В цих системах місце терміналу зайняла ПЕВМ, а мейнфрейма - комп'ютер-сервер. Раніше ми розглядали спектр моделей подібних

систем з різним розподілом функцій між компонентами. Якщо не брати до уваги модель «розподіленого представлення» (по суті повторює модель централізованої розрахованої на багато користувачів системи), то можна сказати, що системи типу клієнт-сервер мають наступні переваги*: висока* живучість і надійність, легкість масштабування, якісний призначений для користувача інтерфейс, можливість одночасної роботи з декількома додатками, високі характеристики оперативності обробки інформації.

Основним *недоліком* клієнт-серверних систем є те, що вони орієнтовані на дані, а не на інформацію. Це вимагає від користувача знання не тільки наочної області, а і специфіки прикладної програми, що використовується.

Істотним недоліком можна вважати також складність перенесення таких систем на інші комп'ютерні платформи і інтеграцію з іншими пакетами через «закритість» протоколів взаємодії компонентів систем, що використовуються.

Ще один недолік полягає в складності адміністрування системи і її вразливості при непередбачуваних або зловмисних діях користувача або комп'ютерних вірусів.

3.*Корпоративні системи intranet,* на відміну від систем клієнт-сервер, орієнтовані не на дані, а на інформацію в її остаточному і придатному для використання некваліфікованим користувачем вигляді (рисунок 22.7).

Нові системи об'єднують в собі переваги централізованих розрахованих на багато користувачів систем і систем типу клієнт-сервер. Їм властиві наступні риси:

- на сервері породжується інформація, придатна для використання, а не дані (наприклад, у випадку СКБД - записи БД);
- при обміні між клієнтською і серверною частинами використовується протокол відкритого стандарту, а не якоїсь конкретної фірми;
- прикладна система знаходиться на сервері, і тому для роботи користувача на комп'ютері-клієнті достатньо мати програму-навігатор (можуть бути і інші рішення, коли частина обробки проводиться на комп’ютері-клиєнті).

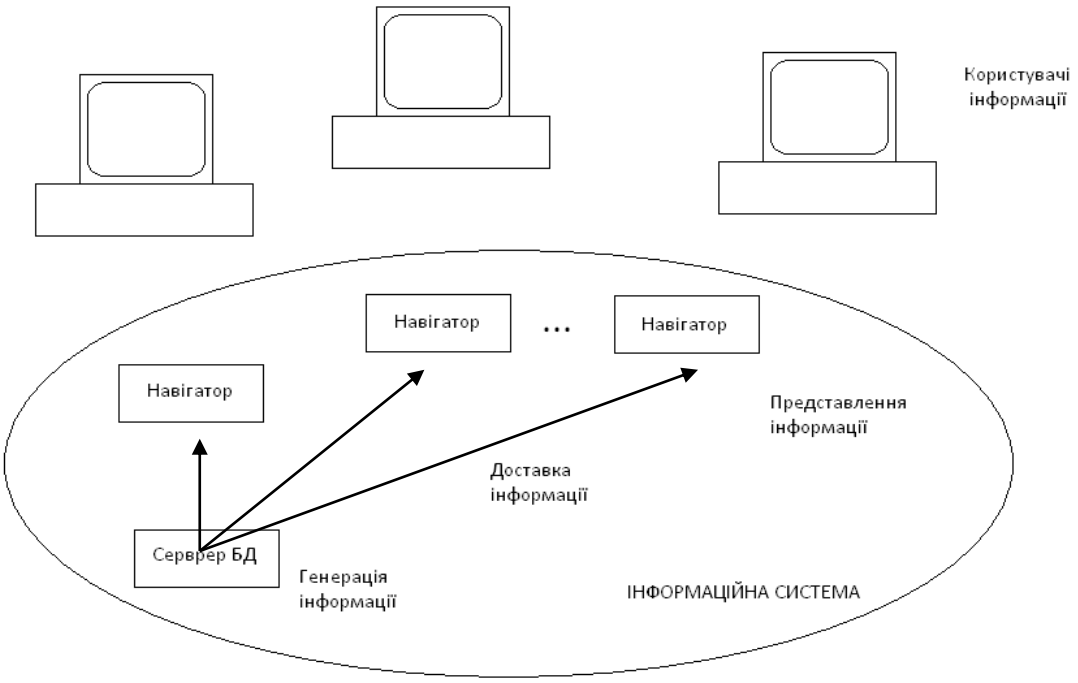


Рисунок 22.7 - Системи, що поставляють інформацію

В разі, коли джерелом інформації в Internet і Intranet є БД, має місце взаємодія компонентів WWW і традиційних СКБД. Розрізняють два наступні варіанти функціонування програмного забезпечення WWW по доступу до БД: на стороні Web-серверу (рисунок 22.8а) і на стороні Web-клієнта (рисунок 22.8б).

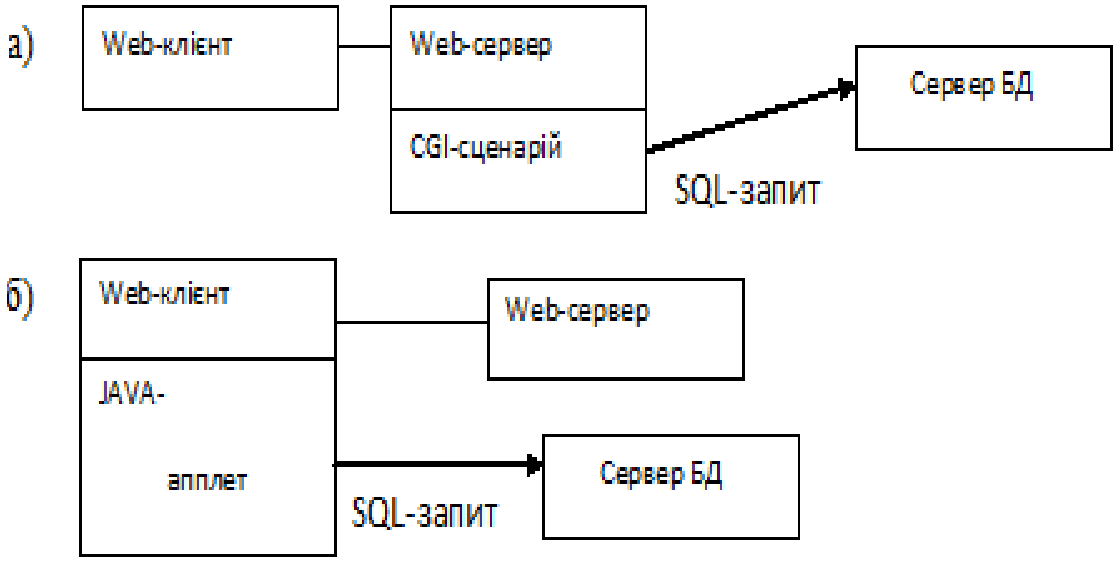


Рисунок 22.8 - Моделі доступу до бази даних в Internet

В моделі ***доступу*** *до БД на стороні серверу* звернення до серверу БД звичайно проводиться шляхом виклику програмами Web-серверу

зовнішніх по відношенню до них програм відповідно до угод одного з інтерфейсів: CGI (Common Gateway Interface — загальний шлюзовий інтерфейс), FastCGI або API (Application Program Interface — інтерфейс прикладного програмування).

Зовнішні програми взаємодіють з сервером БД на мові SQL, безпосередньо звертаючись до конкретного серверу або використовуючи драйвер ODBC.

Зовнішні програми пишуться на звичайних мовах програмування назразок Сі, Сі++ і Паскаль або спеціалізованих мовах типу Perl або PHP. Програми, розроблені відповідно до інтерфейсу CGI, називаються *CGI-сценаріями* або *CGI-скриптами.*

Для підтримки цього механізму *на стороні клієнта*в мові HTML є засіб включення в документ форм (тег <FORM> мови HTML) представлення запитів до БД.

Процедура доступу до БД з використанням інтерфейсу CGI включає наступні етапи:

1.Запит Web-клієнта до Web-серверу сторінки, що містить форму зверненнядо БД, якщо при перегляді документа користувачем Web-клієнт зустрічає посилання натаку сторінку.

2.Заповнення Web-клієнтом що міститься на отриманій сторінці форми запиту доБД і відправка її Web-серверу. Правильність заповнення форми можна контролювати за допомогою нескладної програми, що безпосередньо знаходиться в області HTML-сторінки, в якій описана форма (звичайно для цього використовують мови VBScript або JavaScript).

3.Web-сервер, отримавши цю форму, запускає відповідну зовнішню CG1-програму, передаючи їй параметри (адреса і ім'я цієї програми вказується в атрибуті ACTION тега <FORM>).

4.Зовнішня програма перетворить описаний у формі запит до БД у відповідний текст запиту на мові SQL, з якою звертається до серверу БД.

5.Після отримання результатів запиту зовнішня програма формує необхідну HTML-сторінку, передає її Web-серверу і завершує своє виконання.

6.Web-сервер передає сформовану HTML-сторінку Web-клієнту.

Основними перевагами *застосування специфікації* CGI для організації доступу до даних бази є наступні:

• мовна незалежність — сценарій може бути написаний практично на будь-якій мові програмування, що містить засоби обробки рядків;

• процесна ізольованість — сценарій виконується на сервері як окремий процес, що не має доступу до захищеної системної інформації серверу;

• широка поширеність, оскільки CGI-стандарт застосовний на кожному Web-сервері;

• архітектурна незалежність — програми, розроблені відповідно до CGI-стандарту, не залежать від архітектури серверу.

До головних *недоліків* застосування специфікації CGI відносяться наступні:

• необхідність всякий раз встановлювати і розривати з'єднання з базою даних, оскільки відсутні засоби підтримки постійного з'єднання Web-серверу і СКБД;

• обмеження на обробку початкової інформації для запитів і результатів виконання запитів;

• трудомісткість виконання програм, пов'язана із запуском програми як відладного процесу. Це уповільнює обробку запитів до даних і знижує ефективність роботи серверу.

Для усунення недоліків CGI-специфікації розроблена специфікація API. Програми, розроблені по цій специфікації, швидше і ефективно виконуються, оскільки організовані у вигляді динамічних бібліотек. Самими відомими є два інтерфейси цього вигляду: NSAPI (компанія Netscape) і ISAPI (компанія Microsoft).

Перевагою цієї технології є прискорення виконання програм, оскільки програма виконується в рамках основного серверного процесу. Самі програми мають більшу функціональність, ніж CGI-сценарії, наприклад, з'явилася можливість контролювати доступ до файлів серверу. До *недоліків* специфікації API відносяться наступні:

• мовна залежність — програма може бути написаний тільки на мові, підтримуваній даним API. Частіше всього це мови C і C++ (мова Perl, широко вживана при написанні CGI-скриптів, не застосовується);

• слабкий захист серверу від помилок прикладних програм і від можливості несанкціонованого доступу до системних ресурсів, оскільки API-програма виконується в адресному просторі основного процесу серверу;

• програми, як правило, непереносні на інші програмно-апаратні платні форми, оскільки прив'язані до інтерфейсу і архітектури серверу.

Інтерфейс FastCGI поєднує переваги попередніх технологій і більш близький до інтерфейсу CGI. Додатки, написані відповідно до нього, запускаються як окремі ізольовані процеси, але є постійно активними і не завершуються після звернення до даних як CGI-сценарії.

При доступі до БД *на стороні клієнта* основним засобом реалізації механізмів взаємодії Web-клієнта і серверу БД є мова Java. Крім того, може використовуватися JavaScript, розроблена для розширення можливостей декларативної мови HTML (немає операторів привласнення, порівняння, математичних функцій і ін.) на основі додавання процедурних засобів. Програми на мові JavaScript виконуються на комп'ютері Web-броузером в режимі інтерпретації.

Якщо в HTML-документі вимагається отримати дані з БД, то поступають таким чином.

1. На мові Java пишуться додаткові програми, звані аплетами (Java-applets), які потім компілюються. В результаті виходять мобільні (машинно-незалежні) програми. Останні можуть виконуватися в режимі інтерпретації або служити початковою інформацією для генерації програм, готових до виконання на різних апаратно-програмних платформах.

2.В тексті HTML-документа в потрібних місцях ставляться посилання на відповідні аплети. Самі програми зберігаються на сервері.

3.В процесі роботи з гіпертекстом при виявленні в тексті посилання на аплет відбувається автоматична пересилка Java-програми з серверу в середовище виконання браузера і завантаження на виконання. Остання в діалозі з користувачем уточнює параметри запиту до БД (засоби підтримки графічного призначеного для користувача інтерфейсу в мові Java є).

4.Отримавши управління, Java-аплет здійснює взаємодію з *сервером БД,* внаслідок чого отримана з бази інформація представляється користувачу.

Для звернень до серверів БД розроблений стандарт JDBC (Java DataBase Connectivity — сумісність баз даних для Java), заснований на концепції ODBC. Стандарт JDBC розроблений фірмами Sun/JavaSoft і забезпечує універсальний доступ до різних баз даних на мові Java.

З двох розглянутих схем однозначної переваги тому або іншому варіанту віддати не можна. Все залежить від цілей і умов розробки клієнт-серверних програм (найістотнішою виявляється залежність від Ос, а також від вигляду Web-сервера).

Перевагою моделі доступу на стороні серверу є порівняльна простота програм-навігаторів (Web-клієнтів) і зручність адміністрування системи, оскільки основна частина програмного забезпечення знаходиться на машині Web-серверу. Очевидним *недоліком* системи є можливе погіршення характеристик оперативності отримання інформації при великому навантаженні на Web-сервер і браку його потужності.

В другій моделі клієнтська частина системи виявляється складнішою, ніж в першій. Це ускладнює навігатор, але в той же час розвантажує Web-сервер.

В порівняно недавно випущених програмних продуктах фірми Microsoft підтримуються обидві схеми. На стороні клієнта (в середовищі Internet Explorer) існує можливість використовувати динамічний HTML, який реалізується на мові VBScript. Доступ до даних на стороні серверу здійснюється за допомогою так званих ASP-сторінок (Active Server Pages — активні серверні сторінки). ASP-сторінки — це сценарії на мові VBScript, що зберігаються і виконуються, Web-сервером (Internet Information Server).

Організація баз даних для сайтів

Робота з базами даних - це одна з найважливіших складових програмування сайтів динамічного типу. Динамічні сайти - це сайти, в яких веб-сторінки генеруються або формуються (створюються динамічно) в процесі виконання запиту користувача. База даних - це сукупність взаємопов’язаних даних, яку можна спільно використовувати та керування якою здійснюється централізовано. Програмування сайтів виконується з метою створення деякого корисного функціоналу. Чи то при формуванні сторінок «на льоту», чи при реагуванні на дії відвідувачів сайтів – вирішується задача взаємодії з базами даних.

Програмування сайтів динамічного типу виконується за допомогою різних скриптів, які розділяються зазвичай на серверні і клієнтські. За допомогою серверних скриптів дозволяється обробляти дані, введені відвідувачами сайтів у веб-форми, генерувати динамічні сторінки, відсилати й приймати cookies. Для отримання інформації, необхідної при виконанні подібних дій, серверні скрипти звертаються до БД. Звернення скрипта до БД називається запитом.

Для побудови запитів до БД широко застосовується SQL. У програмуванні сайтів керування БД здійснюється за допомогою клієнт-

серверних систем керування базами даних (СКБД), таких як Oracle, MS SQL Server, PostgreSQL, MySQL та ін. Клієнт-серверні СКБД обробляють запити централізовано, до їх переваг відносять забезпечення високої надійності БД, високої доступності і високої безпеки.

Програмування сайтів, що взаємодіють різним чином з базами даних, включає кілька основних етапів роботи з БД: побудова запитів до БД за допомогою мови SQL, програмування сценаріїв для обробки цих запитів і програмування модулів для відображення результатів обробки запитів.

Надмірне число звернень від сайтів до БД робить завантаження сайтів більш повільним, збільшує навантаження на сервер. У результаті можливі збої в роботі сайтів, аж до повного припинення доступу. Зменшення кількості запитів до БД дозволяє зменшити навантаження на сервер, а також зменшити час завантаження динамічних сторінок з сервера. Тому оптимізація взаємодії сайтів з БД - це одне із завдань професійного програмування сайтів.

Контрольні запитання:

1. Назвіть основні методи доступу до даних і вкажіть випадки найкращого їхнього використання.
2. Наведіть приклад типового набору блокувань об'єктів БД.
3. Вкажіть правила поєднання блокувань.
4. Назвіть основні різновиди тупіків.
5. Наведіть приклад взаємного тупіка в розподіленій БД.
6. Вкажіть основні варіанти створення інформаційної системи в локальній мережі.
7. Опишіть схему функціонування інформаційної системи типу файл-сервер з немережною СКБД.
8. Як організовується обробка в інформаційних системах типу файл-сервер з мережною СКБД?
9. Опишіть схему функціонування інформаційної системи типу клієнт-сервер.
10. Яке призначення збережених процедур і тригерів?
11. Дайте поняття збережених команд і курсора.
12. Як організовується доступ до даних за допомогою інтерфейсу ODBC?
13. Дайте загальну характеристику мережі Internet.

14. Вкажіть основні моделі доступу до БД в мережі Internet.
15. Які мови програмування використовуються для доступу до БД в Internet?
16. Яке призначення CGI-сценаріїв?
17. Охарактеризуйте технологію Intranet.

Тестові завдання:

1. Чи встановлюється режим повного блокування даних на об'єкти?

а) встановлюється частково
б) не встановлюється
в) встановлюється на деякий час
г) правильної відповіді немає

2. Що означає повне блокування даних?

а) повну заборону всіх операцій
б) заборона деяких визначених операцій
в) всі операції дозволені
г) правильної відповіді немає

3. Як здійснюється в мережі доступ додатку до баз даних?

а) через мережу інтернет мовою VB
б) шляхом блокування об'єктів баз даних
в) шляхом виклику на мові SQLстандартних функцій інтерфейсу ODBC
г) правильної відповіді немає

4. Що означає вислів взаємний тупік?

а) ситуація, коли кожен з двох користувачів прагне захопити вже захоплені дані
б) ситуація, коли один з користувачів захопив об'єкт, а інший чекає, коли він звільниться
в) ситуація, коли нікому з користувачів не потрібні не потрібні дані
г) правильної відповіді немає

5.Які основні функції виконує сервер баз даних мережі?

а) управління та захисту даних у базі даних
б) забезпечує виконання додатків користувача

в) здійснює копіювання даних
г) правильної відповіді немає

6.Що означає термін збережені процедури?
а) програми обробки даних, які знаходяться на сервері
б) форми і звіти різних баз даних
в) запит до бази даних
г) глобальні мережі, локальні мережі, а також окремі комп'ютери

7.Що означає термін тригер?
а) програми обробки даних, які знаходяться на сервері
б) різновид збереженої процедури, яка викликається при виникненні деяких подій в базі даних
в) запит до бази даних
г) локальні мережі

8 .Що означає FTP?
а) програми обробки даних, які знаходяться на сервері
б) різновид збереженої процедури, яка викликається при виникненні деяких подій в базі даних
в) запит до бази даних
г) протокол, який дозволяє передавати файли довільного формату між двома комп’ютерами

Розділ 23

Об’єктно-орієнтовані системи

- **Найважливіші властивості об’єктно-орієнтованого підходу**
- **Об’єктно-орієнтована система Rational Rose**
- **Рекомендації з використання CASE-систем**

Поява об'єктно-орієнтованих CASE-систем викликана рядом переваг об'єктно-орієнтованого підходу перед структурним, заснованих на трьох найважливіших властивостях: інкапсуляції, спадкоємстві і поліморфізмі.

Інкапсуляція означає об'єднання в єдине ціле даних і алгоритмів (функцій і методів) їх обробки, а також приховання даних всередині об'єктів, що дозволяє підвищити надійність програмного забезпечення, що розробляється. Властивості *спадкоємства* і *поліморфізму* дозволяють прискорити процес розробки програм, а також спростити адаптацію систем на нові умови застосування за рахунок гнучкого механізму нарощування можливостей програм в процесі їх розробки.

Областю застосування об'єктно-орієнтованих інструментальних систем є складні проекти, такі як: створення операційних систем, засобів розробки додатків і систем реального часу.

В рамках об'єктно-орієнтованого підходу існує безліч моделей опису (нотацій) і методів розробки програмних систем.

Сучасні об'єктно-орієнтовані CASE-системи можна розділити на дві основні групи: CASE-засоби, підтримуючі декілька об'єктно-орієнтованих моделей, і засоби, орієнтовані тільки на один вид моделей. В системах першого типу звичайно є можливість переходу від однієї моделі до іншої. Іноді в цих системах надається можливість створювати власні нотації.

Об'єктно-орієнтована система Rational Rose

Rational Rose є сімейством об'єктно-орієнтованих CASE-систем фірми Rational Software Corporation, що служить для автоматизації аналізу і проектування ПО, генераціями кодів на різних мовах і підготовками проектної документації. Крім того, в його складі є засоби реінженерінга програм, що забезпечують повторне використання програмних компонентів в нових проектах. В цій системі використовується синтез-методологія об'єктно-орієнтованого аналізу і проектування.

Конкретний варіант системи визначається мовою, на якій виконується генерація кодів програм (C++, Smalltalk, PowerBuilder, Ada, SqlWindows і ObjectPro). Основним варіантом системи є Rational Rose/C++, дозволяючий генерувати програмні коди на C++, готувати проектну документацію у вигляді діаграм і специфікацій.

В процесі роботи за допомогою Rational Rose виконується побудова діаграм і специфікацій, що визначають логічну і фізичну структуру моделі, її статичні і динамічні властивості. В їх склад входять наступні діаграми: класів, станів, сценаріїв, модулів і процесів.

Основними компонентами системи є наступні:

• репозиторій, що представляє об'єктно-орієнтовану БД;

• графічний інтерфейс користувача;

• засоби перегляду проекту, що забезпечують переміщення по елементах проекту, в тому числі за ієрархіями класів, перемикання між видами діаграм;

• засоби контролю проекту, що дозволяють знаходити і усувати помилки;

• засоби збору статистики;

• генератор документів, що дозволяє формувати тексти вихідних документів на основі інформації з репозиторія.

Крім того, для кожної мови програмування додається свій генератор коду і аналізатор для C++, що забезпечує відновлення моделі проекту по початкових текстах програм (реінженерінг). Засоби автоматичної генерації кодів програм на C++ на основі логічної і фізичної моделей проекту формують заголовні файли і файли описів класів і об'єктів. Отриманий таким чином скелет програми можна доповнити шляхом безпосереднього програмування на C++.

Аналізатор кодів C++ дозволяє створювати модулі проектів за інформацією, що міститься у визначених користувачем початкових текстах програм. Аналізатор здійснює контроль правильності початкових текстів і діагностику помилок. Отримана в результаті модель може використовуватися в декількох проектах.

В результаті розробки проекту за допомогою Rational Rose формуються наступні діаграми: класів, станів, сценаріїв, модулів і процесів. Крім того, створюються наступні компоненти:

• специфікації класів, об'єктів, атрибутів і операцій;

• заготівки текстів програм;

• модель програмної системи.

Модель програмної системи є текстовим файлом, що містить всю інформацію про проект. Заготівки текстів програм формуються у вигляді заголовних файлів і заготівок для методів. Система включає в програмні файли коментарі. В остаточні програми початкові тексти заготівок перетворяться програмістами.

Рекомендації з використання CASE-систем

Аналіз характеристик і можливостей більшості сучасних CASE-систем дозволяє зробити наступні висновки.

1.CASE-системи дозволяють прискорити і полегшити розробку, підвищити якість створюваних програм і інформаційних систем. Багато які CASE-системи мають засоби управління колективною роботою над проектом.

2. CASE-системи особливо корисними виявляються на початкових етапах розробки. Вони є необов'язковою частиною інструментарію розробника і поки не можуть замінити засоби проектування і розробки у складі СКБД. Однією з основних причин цього є різноманітність засобів розробки додатків програмно-апаратних платформ і методологій проектування.

3. Можливість переходу від концептуальної моделі БД до фізичної і назад, що надається багатьма CASE-системами, корисна для вирішення задач аналізу, вдосконалення і перенесення додатків з середовища однієї СКБД в іншу.

4. Більшість сучасних CASE-систем є структурною, але завдяки деяким перевагам об'єктно-орієнтованих систем, останні набувають все більшу популярність, особливо при реалізації складних проектів.

5. Сучасні CASE-системи орієнтовані на кваліфікованого користувача, оскільки для їх використання потрібне знання теорії проектування баз даних. Так, наприклад, для розробки структури БД за допомогою системи S-Designor інформацію про проектовану інформаційну систему потрібно представити у вигляді ER-моделі.

Залежно від поставлених перед користувачем задач (розробка схеми БД, реінженерінг, розробка готового додатку і т. д.), умов розробки і інших чинників якнайкращою може виявитися та або інша CASE-система. Іноді доцільно використовувати декілька CASE-систем.

Використання декількох CASE-систем часто дозволяє об'єднати переваги систем, що використовуються, і істотно скоротити терміни рішення задач дослідження або розробки. Для прикладу приведемо

схему можливого сумісного використання CASE-систем ERWin, BPWin і Rational Rose.

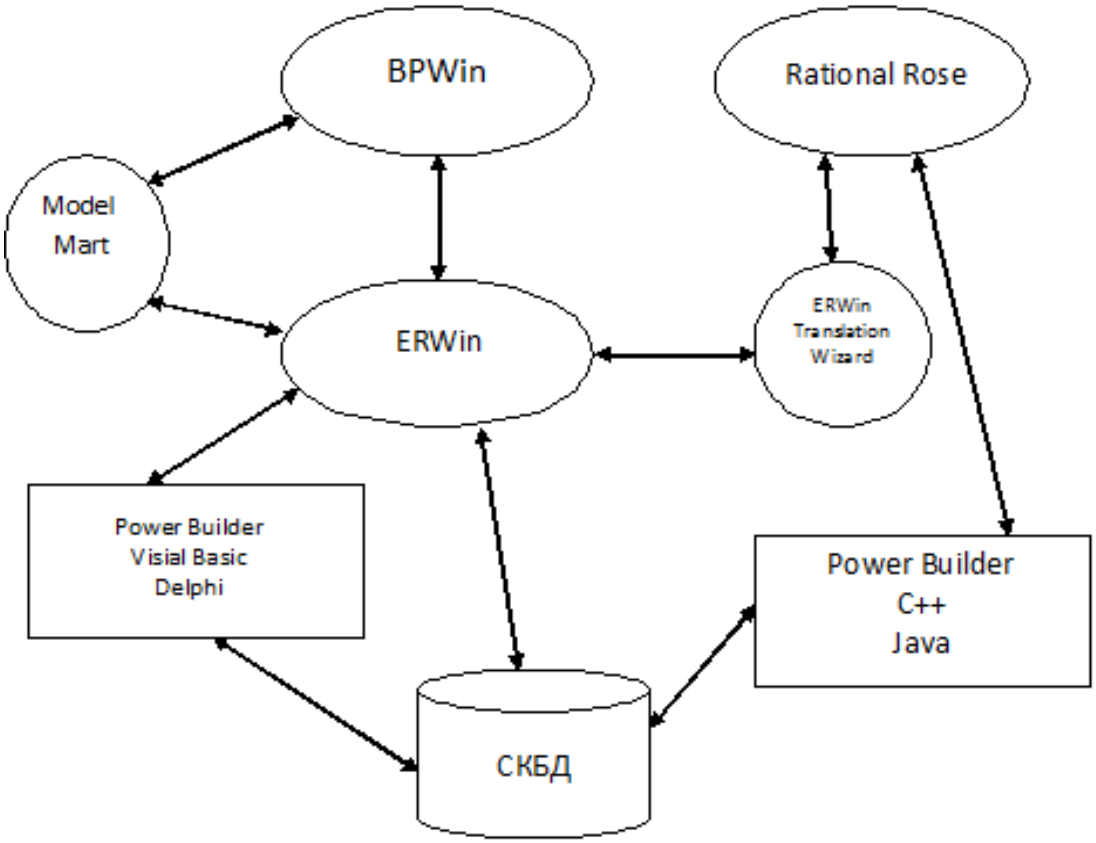


Рисунок 23.1 - Схема взаємодії ERWin, BPWin і Rational Rose

Як показано на мал. 23.1, різні CASE-системи можуть взаємодіяти одна з одною напряму (ERWin і BPWin), або за допомогою додаткових модулів (Model Mart - засобів колективної розробки, ERWin Translation Wizard - модуль імпорту в ERWin моделей, створених в Rational Rose).

Контрольні запитання:

1. Дайте визначення CASE-засобам і CASE-технології.
2. Дайте визначення поняття моделі життєвого циклу ПО і назвіть основні варіанти моделей.
3. Перерахуйте вимоги до перспективної CASE-системи.
4. Охарактеризуйте спіральну модель життєвого циклу ПО.
5. Перерахуйте поширені моделі і діаграми графічного представлення, що використовуються при структурному аналізі і проектуванні.
6. Наведіть приклад діаграми потоків даних.
7. Охарактеризуйте методологію функціонального моделювання, наведіть приклад декомпозиції діаграм.

8. Що є уніфікованою мовою моделювання UML?
9. Назвіть типи діаграм уніфікованої мови моделювання.
10. Для чого служать діаграми прецедентів використання і діаграми класів.
11. Наведіть приклад діаграми проходження.
12. Назвіть ознаки класифікації CASE-засобів.
13. На які групи діляться CASE-системи за їх функціональною орієнтацією?
14. Наведіть приклад незалежної CASE-системи структурного типу і дайте їйхарактеристику.
15. Охарактеризуйте CASE-систему Designer/2000.
16. До якого типу CASE-систем відноситься Rational Rose?
17. Назвіть основні компоненти системи Rational Rose.

Тестові завдання:

1. Що означає інкапсуляція?

а) Розділення даних від алгоритмів іх обробки , а також приховання даних в алгоритмах частково

б) Об'єднання в єдине ціле даних і алгоритмів іх обробки, а також приховання даних всередині об'єктів

в) Встановлюється на деякий час заборона деяких операцій

г) Правильної відповіді немає

2. Які можливості дають властивості спадкоємства і поліморфізму?

а) Повну заборону всіх підозрілих операцій розробки програм

б) Заборона деяких визначених операцій а також ускладнюється адаптація систем на нові умови застосування програм

в) Прискорити процес розробки програм, а також спростити адаптацію систем на нові умови застосування

г) Правильної відповіді немає

3. Сучасні об'єктно-орієнтовані CASE-системи можна розділити на дві основні групи:

а) CASE-засоби, підтримуючі декілька об'єктно-орієнтованих моделей і засоби, орієнтовані тільки на один вид моделей

б) CASE-засоби, підтримуючі декілька процедурних моделей і засоби орієнтовані шляхом блокування об’єктів баз даних
в) CASE-засоби, підтримуючі модель виклику на мові SQLi стандартних функцій інтерфейсу ODBC
г) правильної відповіді немає

4. Яка можливість є в системах підтримуючих декілька об’єктно-орієнтованих моделей?
а) переходу від однієї моделі до іншої
б) розробляти ігри
в) тупікової ситуації
г) правильної відповіді немає

5.Використання декількох CASE-систем часто дозволяє:
а) об'єднати переваги систем, що використовуються, і істотно скоротити терміни рішення задач дослідження або розробки
б) забезпечити виконання додатків користувача
в) здійснювати копіювання, видалення, поновлення даних систем
г) правильної відповіді немає

Список скорочень

АБД - адміністратор БД
АІС - автоматизована інформаційна система
БД – база даних
БнД - банк даних
ІС - інформаційна система
ЛОМ - локальна обчислювальна мережа
ПО - предметна область
РМД - реляційна модель даних
РС – робоча станція
СД - словник даних
СКБД - система керування базами даних
ANSI (American national standards institute) - Американський національний інститут стандартів
API (Application Programming Interface) - інтерфейс прикладного програмування
ASP-сторінки (Active Server Pages) - активні серверні сторінки
CASE (Computer Aided Software Engineering) - розробка програмного забезпечення за допомогою комп'ютера
CGI (Common Gateway Interface) - загальний шлюзовий інтерфейс
DCL (Data Control Language) - мова керування даними
TCL (Transaction Control Language) - мова керування транзакціями
DDL (Data Definition Language) - мова визначення даних
DML (Data Manipulation Language) - мова маніпулювання даними
DFD - діаграми потоків даних
DSD (Data Structure Diagrames) - діаграми структур даних
ERD (Entity Relations Diagram) - діаграми сутність-зв'язок
ELM (Event List Matrix) - матриця списку подій
FSD (Form Sequence Diagram) - діаграма послідовності форм
HTML (Hypertext Markup Language) - мова розмітки гіпертексту
HTTP (Hypertext Transfer Prococol) - протокол передачі гіпертексту
ISBL (Information Systems Base Language) – базова мова інформаційних систем
ISO (International Standards Organization) - міжнародна організація стандартів
OMG (Object Management Group) - група управління об'єктом
OMT (Object Modeling Technique) - метод моделювання об'єктів
OOSE (Object-Oriented Software Engineering) - об'єктно-орієнтоване проектування програмного забезпечення
ODBC (Open Database Connectivity) – сумісність відкритих баз даних.
QBE – Query By Example
SAD (System Architecture Diagram) - діаграми архітектури
SADT – (Structured Analysis and Design Technique) - технологія структурного аналіза і проектування

SCD (Structure Charts Diagram) - модель бізнес-логіки
SQL (Structured Query Language) - структурована мова запитів
WFF (Well Formed Formula) - правильно побудовані формули
WWW (World Wide Web) - всесвітня павутина

Список використаних джерел

1. Бородаєв В.А., Кущів В.Н. Банки і бази даних: Навчальний посібник. Л.: ВІКІ, 1989.
2. Основи сучасних комп'ютерних технологій: Навчальний посібник / Під редакцією проф. Хомоненко А.Д. Автори: Артамонов Б.Н., Брякалов Р.А., Гофман В.Е. та інші. СПб: КОРОНА прінт, 1998.
3. Системи управління базами даних і знань: Довід.вид. / Наумов А. М., Вендров А. М., Іванов В. К. та ін; Під. ред. Наумова А. Н. - М .: Фінанси і статистика, 1991.
4. Дейт К.Дж. Введення в системи баз даних.; Пер. з англ. 6-те вид. К.: Діалектика, 1998. – 784 с.
5. Романов Б.А., Кушніренко А.С. dBaseIV: Призначення, функції, застосування. – М.: Радіо і зв'язок, 1991. – 384 с.
6. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. - Издание второе, дополненное и переработанное - СПб.: КОРОНА принт, 2002. - 672с.
7. Гайджаржи В.І. Дацюк О.А. Основи проектування та використання баз даних: Навчальний посібник. Друге видання виправ. і доповн. - К.: ІВЦ "Видавництво Політехніка", ТОВ "Фірма Періодика" 2004. - 256 с.
8. Ульман Л. MySQL/Ларри Ульман; Пер. с англ. Слинкина А. А. – М.: ДМК Пресс; СПб.: Питер, 2004. – 352 с.: ил.
9. Кузнецов Максим, Симдянов Игорь. MySQL на примерах. — Спб.: «БХВ-Петербург», 2008. — с. 952.
10. В. Васвани. MySQL: использование и администрирование = MySQL Database Usage & Administration. — М.: «Питер», 2011. — 368 с.
11. Роберт Шелдон, Джоффрей Мойе. MySQL 5: базовый курс = Beginning MySQL. — М.: «Диалектика», 2007. — 880 с.
12. Вейд А. Стандарты об'єктних запитів // Системи Управління Базами Даниних № 4, 1996. . 89-97.
13. Елісєєв В., Ладиженській Г. Введение в Іптранет // Системи Управління базами Даних № 5-6, 1996. З. 19-43.
14. Каліїченко Л. Стандарт систем управління об'єктними базами даних ODMG: короткий огляд і оцінка стану // Системи Управління Базами Даних № 1, 1996. З. 102-109.
15. Ковалів З, Доступ до баз даних з використанням технології WWW //Системи Управління Базами Даних № 5-6, 1996. З. 4-9.
16. Ладиженській Г. Системы управління базами даних — коротко про головне // Системи управління базами даних № 4, 1995. З. 123-141.
17. Олейников А, Я. Открытые системи: концепція і реальність // Відкриті системи № 4, 1993. - З. 53-58.
18. Орлик З. В. Borland Delphi як засіб розробки додатків, що масштабуються // Системи Управління Базами Даних № 4, 1995. З. 50-56.
19. Роберт Сигнор, Міхаель О. Стегман. Використання ODBC для доступу до баз даних: Пер. з англ. М.: БІНОМ; НАУКОВА КНИГА.

Додаток А
Відповіді на тестові завдання:

Розділ 1
1б ; 2б ; 3г ; 4г ; 5а
Розділ 2
1а ; 2б ; 3г ; 4б ; 5г
Розділ 3
1в ; 2в ; 3в ; 4б ; 5а ; 6а ; 7д; 8в
Розділ 4
1 г; 2а ; 3г ; 4г ; 5в ; 6г
Розділ 5
1 д; 2г ; 3в ; 4б ; 5б
Розділ 6
1а ; 2б ; 3а ; 4б ; 5г
Розділ 7
1в ; 2г ; 3б ; 4а ; 5б ; 6а ; 7а ; 8в; 9в; 10г
Розділ 8
1б ; 2а ; 3а ; 4а ; 5б
Розділ 9
1а ; 2а ; 3б ; 4в ; 5а
Розділ 10
1а ; 2а ; 3б ; 4а ; 5в
Розділ 11
1б ; 2в ; 3б ; 4а ; 5в
Розділ 12
1а ; 2а ; 3в ; 4б ; 5а
Розділ 13
1а ; 2в ; 3а ; 4б ; 5а; 6а
Розділ 14
1б ; 2а ; 3а ; 4в ; 5б
Розділ 15
1в ; 2в ; 3г ; 4а ; 5а
Розділ 16
1б ; 2в ; 3в ; 4в ; 5б
Розділ 17
1б ; 2а ; 3в ; 4а ; 5а
Розділ 18
1а ; 2а ; 3а ; 4б ; 5а
Розділ 19
1б ; 2а ; 3а ; 4а ; 5а
Розділ 20
1в ; 2а ; 3б ; 4а ; 5а
Розділ 21
1б ; 2в ; 3а ; 4б ; 5в ; 6в ; 7а ; 8а, 9в, 10а, 11б, 12в

Розділ 22
1 б; 2 а; 3 в; 4 а; 5 а; 6 а; 7 б; 8 г
Розділ 23
1б ; 2в ; 3а ; 4а ; 5а

Навчальний посібник

Сидоренко Валентина Володимирівна

Константинова Лілія Володимирівна

Смірнов Сергій Анатолійович

ОРГАНІЗАЦІЯ БАЗ ДАНИХ

Українською мовою

Редактор: Константинова Л. В.

Формат 60х84 1/16. Ум. друк. арк 15.93. Облік. видав арк. 10.59. Тираж 100. Зам 168.

Видавець і виготовлювач СПД ФО Лисенко В. Ф.

25028, м. Кропивницький, вул. Пацаєва, 14, корп. 1, кв. 101. Тел.: (0522) 322-326

Свідоцтво суб'єкта видавничої справи: серія ДК № 3904 від 22.10.2010