МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ

Харківський національний університет внутрішніх справ

Факультет №4

Кафедра інформаційних технологій

ТЕКСТ ЛЕКЦІЇ

з дисципліни «ОРГАНІЗАЦІЯ БАЗ ДАНИХ ТА ЗНАНЬ»

за темою: Поняття системи управління базами даних. Моделі баз даних

Галузь знань: 1701 «Інформаційна безпека»

Напрям підготовки: 6.170102 «Системи технічного захисту інформації»

Освітньо-кваліфікаційний рівень: бакалавр

м. Харків
2017 рік

Текст лекції з дисципліни «Організація баз даних та знань» за темою: «Поняття системи управління базами даних. Моделі баз даних» для курсантів за напрямом підготовки «Системи технічного захисту інформації» (6.0170102).

СХВАЛЕНО
Науково-методичною радою
Харківського національного
університету внутрішніх справ
_____ Протокол № _____
(дата, місяць, рік)

ЗАТВЕРДЖЕНО
Вченою радою факультету №4
_____ Протокол № _____
(дата, місяць, рік)

_____ **Марков В.В.**
(підпис)

ПОГОДЖЕНО
Секцією науково-методичної ради
ХНУВС з технічних дисциплін
_____ Протокол № _____
(дата, місяць, рік)

_____ Струков В.М.
(підпис)

ЗАТВЕРДЖЕНО
На засіданні кафедри інформаційних
технологій
_____ Протокол № _____
(дата, місяць, рік)

_____ Струков В.М.
(підпис)

Рецензенти:

Єрохін А.Л., декан факультету комп'ютерних наук ХНУРЕ, д.т.н., професор

Гнусов Ю.В., завідувач кафедри кібербезпеки ХНУВС, к.т.н., доцент

Розробники: Колісник Т.П.– Харків: Харківський національний університет внутрішніх справ, 2017.

План лекції

1. Вступ.
2. Файлові системи баз даних.
3. Історія розвитку БД та СУБД.
4. Системи з базами даних.
5. Компоненти середовища БД.
6. Розподіл обов'язків в СУБД.
7. Переваги і недоліки СУБД.

Література:

Основна:

1. Дейт К.Дж. **Введение в системы баз данных**. 6-е издание. - Киев – Москва: Діалектика, 1998. - 784 с.; Стор. 14-34.
2. Хансен Г., Хансен Дж. **Базы данных: разработка и управление**. – Москва: Бином, 1999. - 700 с. Пер. с англ.; Стор. 20-34, 45-52.
3. Коннолли Т., Бегг К., Страчан А. **Базы данных: проектирование, реализация и сопровождение**. Теория и практика. 2-е издание. Москва – Санкт – Петербург - Киев: Вильямс, 2000. - 1111 с.; Стор. 36-69, 73-92.
4. Д. Кренке. **Теория и практика построения баз данных**. 8-е изд. – СПб.: Питер, 2003. – 800 с.: ил. – (Серия «Классика Conputer Science»). ; Стор. 24-50, 52-58.
5. Гарсиа-Молина, Гектор, Ульман, Джефри, Д., Уидом, Дженифер. **Системы баз данных**. Полный курс.: Пер. с англ. – М.: Издательский дом «Вильямс», 2003. – 1088 с.: ил.; Стор. 32-38, 40-48.

Додаткова:

1. Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие. Санкт-Петербургский Государственный институт точной механики и оптики (технический университет). Кафедра вычислительной техники. - http://www.cs.ifmo.ru
2. Кузнецов С.Д. Основы современных баз данных. Информационно-аналитические материалы. - http://www.citmgu.ru/

Текст лекції

1.1. Вступ

Бази даних (БД) стали невід'ємною частиною нашого повсякденного життя. У подальших міркуваннях будемо розглядати БД як ***деякий набір зв'язаних даних***, а систему управління ними, чи **СУБД** (Database Management System — **DBMS**), як ***програмне забезпечення***, що керує доступом до цієї бази даних. Більш детальні і точні визначення СУБД будуть викладені далі. Обговорення баз даних у цьому розділі почнемо з вивчення деяких прикладів необхідності використання систем баз даних.

Приклад 1. Доступ до бази даних необхідний при покупці продуктів у найближчому супермаркеті**,** коли касир зчитує з покупок штрих-код. При цьому ручний сканер, зв'язаний з програмою роботи з базою даних, використовує штрих-код для пошуку ціни даного предмета в базі даних усіх товарів. Потім програма відніме кількість усіх тільки що проданих товарів з товарних запасів і відобразить на касовому апараті їхню вартість. Якщо запаси складу опустяться нижче деякого заздалегідь визначеного рівня, то в такому випадку система зможе автоматично направити замовлення на постачання додаткової кількості даного товару. Коли клієнт робить покупки по телефону, касир може

перевірити наявність того чи іншого товару на складі, також запустивши деяку програму баз даних.

Приклад 2. При використанні кредитних карток при покупках**,** касир повинний перевірити наявність кредитних ресурсів. Це можна зробити по телефону або автоматично за допомогою спеціального пристрою, що зчитує дані з кредитної картки і зв'язаний з комп'ютером. У будь-якому випадку при цьому використовується база даних, що містить зведення про покупки, здійснювані за допомогою кредитної картки. На підставі номера кредитної картки спеціальна програма звіряє ціну товарів, що купуються в даний момент і куплених протягом цього місяця товарів із кредитним лімітом. Після підтвердження допустимості такої покупки всі зведення про придбані товари вводяться в базу даних. Обов'язково ще до одержання підтвердження допустимості операції покупки програма бази даних повинна переконатися в тому, що дана картка не знаходиться в списку украдених чи загублених. Крім того, повинна існувати ще одна самостійна програма баз даних, що буде оплачувати рахунок після одержання суми платежу, а також щомісяця посилати кожному власнику кредитної картки повний звіт про баланс на його рахунку.

Приклад 3. При плануванні відпустки звертаєтеся в туристичне агентство. Працівник цього агентства на підставі вашого запиту переглядає бази даних зі зведеннями про наявні путівки і розклад польотів. При бронюванні якої-небудь путівки система баз даних повинна виконати все необхідне для цієї дії та переконатися в тому, що два різних співробітники агентства не бронюють ту саму путівку для даного рейса або забронюванні місця в літаках понад гранично припустиму кількість. Наприклад, якщо в літаку деякого рейса залишилося тільки одне вільне місце, і два співробітники туристичного агентства спробують його забронювати, то система повинна коректно обробити цю ситуацію і дозволити забронювати останнє місце тільки одному співробітнику, пославши іншому повідомлення про відсутність вільних місць. Крім того, кожний з них може мати ще й окрему систему баз даних для виписки рахунків.

Приклад 4. При відвідуванні бібліотеки читачу, можливо необхідно буде звернутися до бази даних, що містить зведення про всі книги, що маються в цій бібліотеці, її клієнтів, заявки на бронювання книг і т.д. У цій БД міститься комп'ютеризований індекс, що дозволяє читачам знаходити потрібну їм книгу за назвою, прізвищами авторів чи тематикою. Як правило, подібна система баз даних здатна обробляти інформацію про бронювання книг, що дозволить вам зарезервувати узяту іншим читачем книгу. Коли цю книгу повернуть, поштою вам буде послане повідомлення, що книга вже на місці, і ви можете її взяти. Крім того, така система може посилати нагадування тим читачам, що не повернули узяту книгу в зазначений термін. Для введення інформації про книгу в такій системі звичайно використовується пристрій сканування штрих-коду, аналогічний тому, що застосовується в супермаркетах. З його допомогою організується облік повертання і видачі книг з бібліотеки.

Приклад 5. При оформленні будь якого страхового поліса (наприклад, для страхування життя, здоров'я, будівлі чи автомобіля) страховий агент може звертатися до декількох баз даних, що містять зведення про різні страхові компанії. Після вказівки персональних зведень, наприклад імені, адреси, віку, а також пристрасті до паління чи спиртним напоям, додаток системи баз даних використає їх для визначення вартості страхового поліса. Страховий агент може переглянути кілька баз даних з метою пошуку страхової компанії, що запропонує вам найкращі умови страховки.

Приклад 6. В університеті може існувати база даних з інформацією про студентів, відвідуваних ними курсах, виплачуваних стипендіях, вже пройдених і досліджуваних у даний час предметах, а також про результати здачі різних іспитів та заліків. Крім того, може також підтримуватися база даних з інформацією про прийом студентів у наступному році, а також база даних з особистими даними про співробітників і зведеннями про їхню зарплату для бухгалтерії.

1.2. Файлові системи баз даних

Незважаючи на те, що **файлові системи баз даних** давно застаріли, все-таки є кілька причин для знайомства з ними.

Розуміння проблем, які властиві файловим системам, може запобігти їх повторення в СУБД. Інакше кажучи, варто врахувати досвід минулих помилок. Насправді, слово "помилки" у даному випадку звучить трохи зневажливо і не дає ніякого представлення про ту корисну роботу, що виконувалася протягом багатьох років. Проте воно дає зрозуміти, що існують і інші, більш ефективні способи керування даними.

Знати принципи роботи файлових систем не тільки дуже корисно, але і необхідно при виконанні переходу від файлової системи до системи баз даних.

ФАЙЛОВІ СИСТЕМИ БД - НАБІР ПРОГРАМ У СУКУПНОСТІ З ДАНИМИ У ФАЙЛАХ, ЩО ВИКОНУЮТЬ ДЛЯ КОРИСТУВАЧІВ ДЕЯКІ ОПЕРАЦІЇ ОБРОБКИ ІНФОРМАЦІЇ, НАПРИКЛАД ВНЕСЕННЯ ЗМІН ТА СТВОРЕННЯ ЗВІТІВ. КОЖНА ПРОГРАМА ВИЗНАЧАЄ СВОЇ ВЛАСНІ ДАНІ І КЕРУЄ ТІЛЬКИ НИМИ.

Файлові системи були **першою спробою комп'ютеризувати** відомі усім ручні картотеки. Подібна картотека (чи підшивка документів) у деякій організації могла містити всю зовнішню і внутрішню документацію, зв'язану з яким-небудь проектом, продуктом, задачею чи клієнтом. Звичайно таких папок буває дуже багато, вони позначаються і зберігаються в одному чи декількох шафах. У кожного з нас є деяка подібність такої картотеки, що містить підшивки документів, які представляють собою рахунки, гарантійні талони, рецепти, страхові і банківські документи і т. д. Якщо нам необхідно знайти якусь інформацію, то виконується перегляд всієї картотеку від початку до кінця для її пошуку. Більш витончений підхід передбачає використання в такій системі алгоритму індексування, що дозволяє прискорити пошук потрібних зведень. Наприклад, можна використовувати спеціальні роздільники чи окремі папки для різних ***логічно, зв'язаних*** типів об'єктів.

Ручні картотеки дозволяють успішно справлятися з поставленими задачами, якщо кількість збережених об'єктів невелика. Так картотеки також підходять для роботи з великою кількістю об'єктів, які потрібно тільки зберігати і витягати, але зовсім не підходять, коли потрібно встановити перехресні зв'язки чи виконати обробку зведень.

Наступним прикладом використання ручних картотек є робота роботу типового агентства з обліку нерухомості у місті, яке виконує операції по здачі в оренду та продажу об'єктів нерухомості. У цьому випадку для збереження даних можуть бути використані окремі файли для кожного об'єкта, що здається в оренду чи продається, для кожного потенційного покупця чи орендаря, а також для кожного співробітника компанії.

Файлові системи були розроблені у відповідь на потребу в одержанні більш ефективних способів доступу до даних. Однак, замість організації централізованого сховища всіх даних підприємства, був використаний децентралізований підхід, при якому співробітники кожного відділу працюють зі своїми власними даними і зберігають їх у своєму відділі.

Наприклад, співробітники ***відділу реалізації,*** які відповідають за продаж і оренду нерухомості, коли клієнт звертається до них з пропозицією здати в оренду об'єкт нерухомості, який йому належить, повинні заповнити відповідну форму про цей об’єкт. У формі указуються такі зведення про об'єкт нерухомості: адреса, кількість кімнат та інформація про самого власника. Крім того, співробітники відділу реалізації обробляють запити від потенційних орендарів, кожний з яких повиннен заповнити іншу форму з даними про об’єкт, який він хоче орендувати або купити та дані про себе.

Такими чином ця система повинна складатися з трьох наступних файлів:

1. Файл даних про нерухомість (**Об’єкти нерухомості**).
2. Файл даних про власників **(Власники).**
3. Файл даних про орендарів **(Орендарі)**.

Співробітники ***відділу контрактів*** відповідають за оформлення договорів про оренду нерухомості. Якщо клієнт дає згоду на оренду деякого об'єкта, то одним із співробітників відділу реалізації заповнюється відповідна форма. У ній указуються всі необхідні зведення про орендаря й про цей об'єкт нерухомості. Ця форма передається у відділ контрактів, співробітники якого привласнюють договору номер і вносять додаткові зведення про оплату і тривалість оренди. Ця система вже буде складатися з таких файлів:

Файл зведення про контракти **(Контракти)**,

Файл даних про нерухомість (**Об'єкти нерухомості**)

Файл даних про орендарів **(Орендарі)**.

Як бачимо, багато даних відділу контрактів подібні даним в інформаційній системі відділу реалізації. Кожен відділ звертається до своїх власних даних за допомогою спеціалізованих програм. Набір програм кожного відділу дозволяє вводити дані, працювати з файлами і генерувати деякий фіксований набір спеціалізованих звітів. Фізична структура і методи збереження записів файлів з даними жорстко визначені в коді програм, що є найважливішим.

Аналогічні приклади можна знайти й в інших відділах. Очевидно, що дуже велика кількість даних у відділах дублюється і це характерно для будь-яких файлових систем. Перш ніж почати обговорення недоліків цього підходу, було б корисно познайомитися з термінологією, яка використовується у файлових системах.

Файл у нашому випадку є простим набором **записів (record)**, що містять логічно зв'язані дані. Наприклад, файл **Об'єкти нерухомості** містить записи, по одному для кожного об'єкту нерухомості, що здається в оренду. Кожен запис містить логічно зв'язаний набір з одного чи декількох **полів** (**field**), кожне з яких представляє деяку характеристику об'єкту що описується. Наприклад, запис файлу **Об'єкти нерухомості** буде складатися з наступних полів:

[Номер об'єкту], [Вулиця], [Будинок], [Місто], [Поштовий індекс], [Тип власності], [Кількість кімнат], [Орендна плата], [Власник].

Тепер поговоримо про обмеження, які властиві файловим системам. Серед найбільш серйозних слід звернути увагу на наступні:

— поділ і ізоляція даних;
— дублювання даних;
— залежність від даних;
- несумісність файлів;

Розглянемо детально про кожне з обмежень.

1.2.1. Поділ і ізоляція даних

Коли дані ізольовані в окремих файлах, доступ до них дуже ускладнений. Наприклад, для створення списку всіх будинків, що відповідають вимогам потенційних орендарів, попередньо потрібно створити тимчасовий файл зі списком орендарів (на підставі даних з файлу **Орендарі)**, які бажають орендувати нерухомість типу "будинок". Потім у файлі **Об'єкти нерухомості** варто здійснити пошук об'єктів нерухомості типу "будинок" з орендною платою нижче встановленого орендарем максимуму. Виконувати подібну обробку даних у файлових системах досить складно. Для витягу відповідної поставленим умовам інформації програміст повинен організувати синхронну обробку двох файлів. Труднощі істотно зростають, коли необхідно витягти дані з більш ніж двох файлів.

1.2.2. Дублювання даних

У файловій системі фактично заохочується неминуче безконтрольне дублювання даних із за децентралізованої роботи з даними, яка виконується в кожнім відділі незалежно. Безконтрольне дублювання даних небажано з наступних двох причин:

1. Дублювання даних супроводжується неощадливою витратою ресурсів, оскільки на введення надлишкових даних потрібно витрачати додаткові час і гроші.
2. Дублювання даних може привести до порушення їхньої **цілісності**. Інакше кажучи, дані, у різних відділах можуть стати суперечливими. При виявленні подібної помилки для її виправлення буде потрібно витратити додатковий час і засоби. Оскільки не існує ніякого автоматичного способу відновлення даних одночасно у файлах різних відділів, неважко передбачити, що подібні протиріччя час від часу обов'язково будуть виникати.

1.2.3. Залежність від даних

Як уже згадувалося вище, фізична структура і спосіб збереження записів файлів даних жорстко зафіксовані в коді програм. Це значить, що змінити існуючу структуру даних досить складно. Наприклад, збільшення у файлі **Об'єкт нерухомості** довжини поля адреси з 50 до 52 символів здається зовсім незначною зміною його структури, але для втіленні цієї зміни буде потрібно, як мінімум, створити одноразову програму спеціального призначення (тобто програму, що виконується тільки один раз), що перетворить вже існуючий файл у новий формат.

Крім цього, всі інші програми, які звертаються до файлу **Об'єкт нерухомості** повинні бути змінені з метою відповідності новій структурі файлу. Причому таких програм може бути дуже багато. Отже, програміст повинен виявити всі такі програми, а потім перевірити ще раз і змінити їх**.** Такаё особливість файлових систем називається **залежністю від програм і даних** (program-data dependence).

1.2.4. Несумісність форматів файлів

Структура файлів визначається кодом програми, і залежить від мови програмування цієї програми. Наприклад, структура файлу, створеного програмною мовою **Pascal**, може відрізнятися від структури файлу, створеного програмною мовою **C++**. Пряма несумісність таких файлів ускладнює процес їхньої спільної обробки.

1.3. Історія розвитку СУБД

Як вже згадувалося вище, попередницями СУБД були файлові системи. Однак поява СУБД не привела до повного зникнення файлових систем. Для виконання деяких спеціалізованих задач подібні файлові системи використовуються дотепер. Вважається, що розвиток СУБД почався ще в 60-і роки, коли розроблявся проект польоту корабля **Apollo** на місяць. Цей проект був початий з ініціативи президента США Дж. Ф. Кеннеді, що поставив задачу висадити людину на місяць до кінця десятиліття. У той час не існувало ніяких систем, здатних обробляти або керувати такою величезною кількістю даних, які необхідні для реалізації цього проекту.

У результаті фахівці основного підрядчика - фірми **North American Aviation** (**NAA**) (тепер ця фірма має назву **Rockwell International**) - розробили програмне забезпечення за назвою **GUAM** (Generalized Update Access Method). Основна ідея GUAM була побудована на тому, що малі компоненти поєднуються разом як частини більш великих компонентів доти, поки не буде зібраний воєдино весь проект. Ця відповідна інвертованому дереву структура часто називається **ієрархічною структурою** (hierarchical structure). У середині 60-х років корпорація **IBM** приєдналася до фірми **NAA** для спільної роботи над GUAM, у результаті чого була створена система **IMS** (Information Management

System). Причина, з якої корпорація IBM обмежила функціональні можливості **IMS** тільки керуванням ієрархіями записів. Вона полягала в тому, що необхідно було забезпечити роботу з пристроями збереження з послідовним доступом - з магнітними стрічками, що були основним типом носія в той час. Через деякий час це обмеження вдалося зняти. Незважаючи на те, що **IMS** є найпершою з усіх комерційних **СУБД**, вона дотепер залишається основною ієрархічною СУБД, яка використовується на більшості великих мейнфреймів.

Іншим помітним досягненням середини 60-х років була поява системи **IDS** (Integrated Data Store) фірми **General Electric**. Роботу над нею очолював один з піонерів досліджень в області систем управління базами даних - **Чарльз Бачман** (Charles Bachmann). Розвиток цієї системи привів до створення нового типу систем управління базами даних - **мережних** (network) СУБД, - що вплинуло на інформаційні системи того покоління. Мережева СУБД створювалася для представлення більш складних взаємозв'язків між даними, ніж ті, котрі можна було моделювати за допомогою ієрархічних структур, а також для формування стандарту баз даних. Для створення таких стандартів у 1965 році на конференції організації **CODASYL** (Conference on Data Systems Languages), що проходила при участі представників уряду США і бізнесменів, була сформована робоча група List Processing Task Force, перейменована в 1967 році в групу **Data Base Task Group** (DBTG). У компетенцію групи DBTG входило визначення специфікацій середовища, що допускала би розробку баз даних і керування даними. Попередній варіант звіту цієї групи був опублікований у 1969 році, а перший повний варіант — у 1971 році. Пропозиції групи DBTG містили три компоненти:

— Мережна схема - це логічна організація всієї бази даних у цілі (з погляду АДБ), що включає визначення імені бази даних, типу кожного запису і компонентів записів кожного типу.

— Підсхема — це частина бази даних, як її бачать користувачі чи програми.

— Мова керування даними - інструмент для визначення характеристик і структури даних, і управління ними.

Група DBTG також запропонувала стандартизувати три різні мови:

1. Мова визначення даних (Data Definition Language — **DDL)** для схеми, що дозволить **АБД** описати її.
2. Мова визначення даних (також **DDL)** для підсхеми, що дозволить визначати в програмах ті частини бази даних, доступ до яких буде необхідний.
3. Мова маніпулювання даними (Data Manipulation Language — **DML),** яка призначена для управління даними.

Незважаючи на те, що цей звіт офіційно не був схвалений Національним Інститутом Стандартизації США (American National Standards Institute — **ANSI**), велика кількість систем було розроблено в повній відповідності з цими пропозиціями групи DBTG. Тепер вони називаються CODASYL-системами, чи DBTG-системами. CODASYL-системи і системи на основі ієрархічних підходів являють собою СУБД **першого покоління.**

Однак ці дві моделі мають наведені нижче **недоліки**:

— навіть для виконання простих запитів з використанням переходів і доступом до визначених записів необхідно створювати досить складні програми;

— незалежність від даних існує лише в мінімальній мірі;

— відсутність загальновизнаних теоретичних основ.

У 1970 році **Э.Ф.Кодд** (E. P. Codd), що працював у дослідницькій лабораторії корпорації IBM, опублікував дуже важливу і дуже своєчасну статтю про **реляційну модель** даних, яка дозволяла усунути недоліки колишніх моделей. Потім з'явилася безліч експериментальних реляційних СУБД. Перші комерційні продукти з'явилися наприкінці 70-х - початку 80-х років, серед яких можна відмітити проект **System R**, розроблений у дослідницькій лабораторії корпорації **IBM**, (розташованої в місті Сан Хосе, штат

Каліфорнія) створений наприкінці 70-х років (Astrahan et al., 1976). Цей проект був задуманий з метою довести практичність реляційної моделі, що досягалося за допомогою реалізації передбачених нею структур даних і необхідних функціональних можливостей. На основі цього проекту були отримані найважливіші результати такі як:

Була розроблена структурована мова запитів **SQL**, яка стала стандартною мовою будь-яких реляційних СУБД.

У 80-х роках були створені різні комерційні реляційні СУБД - наприклад, DB2 чи SQL/DS корпорації IBM чи **Oracle** корпорації **Oracle Corporation**.

В даний час існує кілька сотень різних реляційних СУБД для мейнфреймів і мікрокомп'ютерів, хоча для багатьох з них визначення реляційної моделі носить трохи перебільшений характер. Приклад СУБД для багатьох користувачів може служити система CA-OpenIngres фірми **Computer Associates** і систему **Informix** фірми Informix Software, Inc., а для персональних комп'ютерів є **Access** і **FoxPro** фірми Microsoft, **Paradox** і **Visual dBase** фірми Borland, а також **R:Base** фірми Microrim. Реляційні СУБД відносяться до СУБД **другого покоління.** Більш детально реляційну модель даних розглядемо далі.

Однак реляційна модель також має деякі недоліки - зокрема, обмеженими можливостями моделювання. Для рішення цієї проблеми був виконаний великий обсяг дослідницької роботи. У 1976 році Чен запропонував модель "**сутність-зв'язок**" (Entity-Relationship model - ER-модель), що у даний час стала самою розповсюдженою технологією проектування баз даних і є основою методології **концептуального** проектування баз даних і **логічного** проектування реляційних баз даних. У 1979 році Кодд зробив спробу усунути недоліки власної основної роботи й опублікував розширену версію реляційної моделі - RM/T (1979), потім ще одну версію — RM/V2 (1990). Спроби створення моделі даних, що дозволяють більш точно описувати реальний світ, називають **семантичним моделюванням даних** (semantic data modeling).

У відповідь на все зростаючу складність програм баз даних з'явилися дві нові системи: **об'єктно-орієнтовані СУБД,** чи ОО СУБД (Object-Oriented DBMS - OODBMS), **і об'єктно-реляційні СУБД, чи** ОР СУБД (Object-Relational DBMS -ORDBMS). Однак, на відміну від попередніх моделей, дійсна структура цих моделей не зовсім ясна. Спроби реалізації подібних моделей являють собою СУБД **третього покоління,** і більш детально будуть розглянуті далі.

1.4. Системи з базами даних

Обмеження файлових систем, які перераховані вище, є наслідком двох факторів.

1. Визначення даних міститься усередині програмного коду, а не зберігається окремо і незалежно від них**.**
2. Не передбачено ніяких інших інструментів доступу до даних і їх обробки, крім програм.

Для підвищення ефективності роботи необхідно використовувати новий підхід, а **саме базу даних** (DB або database) **і систему керування базами даних,** чи **СУБД** (Database Management System — DBMS). У цьому розділі представлене більш формальне визначення цих термінів, а також розглянуті компоненти середовища СУБД.

1.4.1. База даних

БАЗА ДАНИХ - НАБІР ЛОГІЧНО ЗВ'ЯЗАНИХ ДАНИХ, ЯКІ СПІЛЬНО ВИКОРИСТОВУЮТЬСЯ ТА ПРИЗНАЧЕНІ ДЛЯ ЗАДОВОЛЕННЯ ІНФОРМАЦІЙНИХ ПОТРЕБ ОРГАНІЗАЦІЇ І ЇХ ОПИС.

Щоб глибше вникнути в суть цього поняття, розглянемо його визначення більш уважно. База даних - це єдине, велике сховище даних, що одноразово визначається, а потім може використовуватися одночасно багатьма користувачами з різних підрозділів. Замість розрізнених файлів з надлишковими даними, тут усі дані зібрані разом з мінімальною часткою надмірності. База даних вже не належить будь-якому єдиному

відділу, а є загальним корпоративним ресурсом. Причому база даних зберігає не тільки робочі дані цієї організації, але і їх опис. З цієї причини базу даних ще називають *набором інтегрованих записів із само описом.* У сукупності, опис даних називається **системним каталогом** (system catalog), чи **словником даних** (data dictionary), а самі елементи опису прийняті називати **мета-даними** (meta-data), тобто "даними про дані". Наявність самоопису даних забезпечує незалежність між ними і програмами (program-data independence).

Застосування баз даних, де визначення даних відділене від програм, подібно методу, який використовується при розробці сучасного програмного забезпечення, коли поряд із внутрішнім визначенням об'єкта існує його зовнішнє визначення. Користувачі об'єкта бачать тільки його зовнішнє визначення і не піклуються про те, як він визначається і функціонує. Одне з переваг такого підходу, а саме **абстрагування даних** (data abstraction), полягає в тому, що можна змінити внутрішнє визначення об'єкта без яких-небудь наслідків для його користувачів, за умови, що зовнішнє визначення об'єкта залишається незмінним. Аналогічним чином, у підході з використанням баз даних, структура даних відділена від програм і зберігається в базі даних. Додавання нових структур даних чи зміна існуючих ніяк не впливає на програми, за умови, що вони не залежать безпосередньо від змінюваних компонентів. Наприклад, додавання нового поля в запис чи створення нового файлу ніяк не вплине на роботу наявних програм. Однак видалення поля з використовуваного програмою файлу вплине на цю програму, а тому його також необхідно відповідним чином модифікувати.

І, нарешті, пояснемо останній термін з визначення бази даних - "логічно зв'язаний". При аналізі інформаційних потреб організації варто виділити сутності, атрибути і зв'язки.

Сутністю (**entity**) називається окремий тип об'єкта організації (людина, місце або річ, поняття або подія), який потрібно представити в базі даних.

Атрибутом (**attribute**) називається властивість, що описує деяку характеристику описуваного об'єкта.

Зв'язок (**relationship**) — це те, що поєднує кілька сутностей.

Подібна база даних представляє сутності, атрибути і логічні зв'язки між об'єктами. Інакше кажучи, база даних містить логічно зв'язані дані.

1.4.2. Система управління базами даних - СУБД

СУБД - ЦЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ЗА ДОПОМОГОЮ ЯКОГО КОРИСТУВАЧІ МОЖУТЬ ВИЗНАЧАТИ, СТВОРЮВАТИ І ПІДТРИМУВАТИ БАЗУ ДАНИХ, А ТАКОЖ ЗДІЙСНЮВАТИ ДО НЕЇ КОНТРОЛЬОВАНИЙ ДОСТУП.

СУБД - це програмне забезпечення, що взаємодіє з прикладними програмами користувача і базою даних і має наведеними нижче можливості:

1. Дозволяє визначати базу даних, що здійснюється за допомогою **мови визначення даних** (DDL - Data Definition Language). Мова DDL надає користувачам засоби призначати типи даних і їх структури, а також засоби завдання обмежень для інформації, яка зберігається в базі даних.
2. Дозволяє вставляти, обновляти, видаляти і витягати інформацію з бази даних, що звичайно здійснюється за допомогою **мови керування даними** (DML - Data Manipulation Language). Наявність централізованого сховища всіх даних і їх описів дозволяє використовувати мову DML як загальний інструмент організації запитів, який іноді називають **мовою запитів** (query language). Наявність мови запитів дозволяє усунути властивим файловим системам обмеження, при яких користувачі вимушені мати справу тільки з фіксованим набором запитів чи постійно зростаючою кількістю програм, що породжує інші, більш складні проблеми управління програмним забезпеченням.

Існує два різновиди мов DML - **процедурні** (procedural) і **непроцедурні** (non-procedural) мови, - які відрізняються між собою способом витягу даних. Основна відмінність між ними полягає в тому, що **процедурні мови** звичайно обробляють інформацію в базі даних послідовно, запис за записом, а **непроцедурні** оперують відразу

цілими наборами записів. Тому за допомогою процедурних мов DML звичайно вказується, ***як*** можна одержати бажаний результат, тоді як непроцедурні мови DML використовуються для опису того, ***що*** варто одержати. Найбільш розповсюдженим типом непроцедурної мови є **мова структурованих запитів** (**Structured Query Language — SQL**), що у даний час визначається спеціальним стандартом і *фактично* є обов'язковою мовою для будь-яких реляційних СУБД. (SQL вимовляється або по буквах "S-Q-L", або як мнемонічне ім'я See-Quel").

3. Надає контрольований доступ до бази даних за допомогою перерахованих нижче засобів:

— системи забезпечення безпеки, що запобігають несанкціонований доступ до бази даних з боку користувачів;

— системи підтримки цілісності даних, що забезпечують несуперечливий стан збережених даних;

— системи керування паралельною роботою програм, що контролюють процеси їх спільного доступу до бази даних;

— системи відновлення, що дозволяють відновити базу даних до попереднього несуперечливого стану, порушеного в результаті збою апаратного чи програмного забезпечення;

— доступного користувачам каталогу, що містить опис збереженої в базі дані інформації.

У новому варіанті відділ реалізації і відділ контрактів використовують власні програмні засоби для доступу до загальної бази даних, організованої за допомогою СУБД. Набір програм кожного відділу забезпечує введення і коректування даних, а також генерацію необхідних звітів. Однак, на відміну від варіанта з файловою системою, фізична структура і спосіб збереження даних у цьому випадку контролюються за допомогою СУБД.

Володіння зазначеними вище функціональними можливостями перетворює СУБД у надзвичайно корисний інструмент. Однак, оскільки для кінцевих користувачів неважливо, наскільки проста чи складна внутрішня організація системи, можна почути заперечення, що СУБД ускладнює роботу, надаючи користувачам набагато більшу кількість даних, ніж їм дійсно потрібно.

Для рішення цієї проблеми в СУБД пропонується інший **механізм** - створення **відображень** (view) - які дозволяють будь-якому користувачеві **мати** свій власний погляд на базу даних. Мова DDL включає засоби визначення відображень, кожне з який є деякою підмножиною бази даних. Наприклад, можна організувати відображення, у якому співробітникам відділу контрактів будуть доступні тільки ті дані, що необхідні для оформлення договорів оренди.

Крім спрощення роботи за рахунок надання користувачам тільки дійсно потрібних їм даних, **відображення мають інші переваги**:

— Забезпечують додатковий рівень безпеки. Відображення можуть створюватися з метою виключення тих даних, що не повинні бачити деякі користувачі. Наприклад, можна створити деяке відображення, що дозволить менеджерам відділень і співробітникам розрахункового сектора бухгалтерії переглядати всі дані про персонал, включаючи зведення про їх зарплату. У той же час для організації доступу до даних інших користувачів можна створити ще одне відображення, з якого всі зведення про зарплату будуть виключені.

— Надають механізм настроювання зовнішнього інтерфейсу бази даних. Наприклад, співробітники відділу контрактів можуть працювати з полем [Щомісячна орендна плата], використовуючи для нього більш коротке і просте ім'я - Рента.

— Дозволяють зберігати зовнішній інтерфейс бази даних несуперечливим і незмінним навіть при внесенні змін у її структуру - наприклад, при додаванні чи видаленні полів, зміні зв'язків, розділенні файлів, їх реорганізації чи перейменуванні.

Якщо у файл додаються чи з нього віддаляються поля, які не використовуються в деякім відображенні, то всі ці зміни на даному відображенні ніяк не відіб'ються. Таким чином, відображення забезпечує повну незалежність програм від реальної структури даних, що дозволяє усунути найважливіший недолік файлових систем.

Приведені вище міркування мали загальний характер. Насправді реальний обсяг функціональних можливостей, які пропонуються у деякій конкретній СУБД, відрізняється від продукту до продукту. Сучасні системи являють собою надзвичайно складне програмне забезпечення, що складається з мільйонів рядків коду і багатьох томів документації. Такий результат прагнення одержати програмне забезпечення, що могло б задовольняти вимогам усе більш загального характеру. Програмне забезпечення СУБД постійно удосконалюється і повинне усе більше і більше розширюватися, щоб задовольняти все новим вимогам користувачів. Більш докладно основні функції СУБД розглядаються далі.

1.5. Компоненти середовища СУБД

У середовищі СУБД можна виділити наступні п'ять основних компонентів:

- апаратне забезпечення;
- програмне забезпечення;
- дані;
- процедури;
- користувачі.

1.5.1. Апаратне забезпечення

Для роботи СУБД і програм необхідно деяке апаратне забезпечення. Воно може варіювати в дуже широких межах - від єдиного персонального комп'ютера чи одного мейнфрейма до мережі з багатьох комп'ютерів. Апаратне забезпечення залежить від вимог даної організації і СУБД, яка використовується. Одні СУБД призначені для роботи тільки з конкретними типами операційних систем чи устаткування, інші можуть працювати із широким колом апаратного забезпечення і різних операційних систем. Найбільш поширеною системою вважається така, що складається з мережі міні-комп'ютерів з одним центральним комп'ютером. На центральному комп'ютері працює **серверна частина** СУБД (backend), що обслуговує і контролює доступ до бази даних. До нього мають доступ інші комп'ютери, які можуть бути розташовані і в інших регіонах. На них працюють **клієнтські частини** СУБД (frontend), що здійснюють взаємодію з користувачами. Подібна архітектура зветься **клієнт/сервер** (client-server), де сервером є комп'ютер із серверною частиною СУБД, а клієнтами — комп'ютери з клієнтськими частинами СУБД.

1.5.2. Програмне забезпечення

Цей компонент охоплює програмне забезпечення самої СУБД і прикладних програм, разом з операційною системою, включаючи і мережне програмне забезпечення, якщо СУБД використовується в мережі. Як правило програми створюються мовами третього покоління, такими як C, Pascal чи VB, а також мовами четвертого покоління, таких як SQL, оператори яких впроваджують у програми на мовах третього покоління. Утім, СУБД може мати з власні інструменти четвертого покоління, призначені для швидкої розробки програм з використанням убудованих непроцедурних мов запитів, генераторів звітів, форм, графічних зображень і навіть повномасштабних програм. Використання інструментів четвертого покоління може істотно підвищити продуктивність системи і сприяти створенню більш зручних для обслуговування програм.

1.5.3. Дані

Імовірно, найважливішим компонентом середовища СУБД (з погляду кінцевих користувачів) є дані. База даних містить як робочі дані, так і мета-дані, тобто "дані про дані". Структура бази Даних називається **схемою** (schema). У нашому простому прикладі схема бази даних складається з чотирьох файлів, чи **таблиць** (table):

- об’єкти нерухомості;
- власники;

— орендарі;
— контракти.

Кожна з наведених таблиць має відповідні поля або **атрибути.** Крім своєї основної функції характеристики конкретних даних деякі атрибути виконують роль зв'язку з іншими таблицями. Наприклад атрибут **Власник** у таблиці **Об'єкти нерухомості** моделює зв'язок між цією таблицею і таблицею **Власник**, тобто деякий власник *володіє* деякою нерухомістю, що здається в оренду.

У системному каталозі містяться наступні зведення:
— імена, типи і розміри елементів даних;
— імена зв'язків;
— обмеження цілісності даних;
— імена зареєстрованих користувачів, яким надані деякі права доступу до даних;
— індекси і структури збереження - наприклад, інвертовані файли чи дерева B+.

1.5.4. Процедури

До процедур відносяться інструкції і правила, що повинні враховуватися при проектуванні і використанні бази даних. Користувачам і обслуговуючому персоналу бази даних необхідно надати документацію, що містить докладний опис процедур використання і супроводи даної системи, включаючи інструкції про правила виконання приведених нижче дій:
— реєстрація в СУБД;
— використання окремого інструмента СУБД чи програми;
— запуск та зупинка СУБД;
— створення резервних копій СУБД;
— обробка збоїв апаратного і програмного забезпечення, включаючи процедури ідентифікації компонента, що вийшов з ладу, виправлення компонента, що відмовив, (наприклад, за допомогою виклику фахівця з ремонту апаратного забезпечення), а також відновлення бази даних після усунення несправності;
— зміна структури таблиці, реорганізація бази даних, розміщеної на декількох дисках, способи поліпшення продуктивності і методи архівації даних на вторинних пристроях збереження.

1.5.5. Користувачі

Останнім, ще не розглянутим нами компонентом середовища СУБД, є користувачі системи. Цей компонент докладно описується далі.

1.6. Розподіл обов'язків у системах з базами даних

У цьому розділі розглянемо згаданий вище п'ятий компонент СУБД - її користувачів. Серед них можна виділити чотири різні групи:
— адміністратори даних і баз даних;
— розроблювачі баз даних;
— прикладні програмісти;
— кінцеві користувачі.

1.6.1. Адміністратори даних і адміністратори баз даних

База даних і СУБД є корпоративними ресурсами, якими варто управляти так само, як і будь-якими іншими ресурсами. Звичайне керування даними і базою даних передбачає управління і контроль за СУБД і розміщеними в неї даними. **Адміністратор даних,** чи **АД** (Data Administrator - DA), відповідає за керування даними, включаючи планування бази даних, розробку і супровід стандартів, бізнес правил і ділових процедур, а також за концептуальне і логічне проектування бази даних. АД консультує і дає свої рекомендації керівництву вищої ланки, контролюючи відповідності загального напрямку розвитку бази даних установленим корпоративним цілям.

Адміністратор бази даних, чи **АБД** (Database Administrator - DBA), відповідає за фізичну реалізацію бази даних, включаючи фізичне проектування і втілення проекту, за

забезпечення безпеки і цілісності даних, за супровід операційної системи, а також за забезпечення максимальної продуктивності програм і користувачів. У порівнянні з АД, обов'язки АБД носять більш технічний характер, для чого необхідне знання конкретної СУБД. і системного оточення. В одних організаціях між цими ролями не робиться розходжень, а в інших важливість корпоративних ресурсів відбита саме у виділенні окремих груп персоналу з зазначеним колом обов'язків.

1.6.2. Розроблювачі баз даних

У проектуванні великих баз даних беруть участь два різних типи розроблювачів: розроблювачі логічної бази даних і розроблювачі фізичної бази даних. **Розроблювач логічної бази даних** займається ідентифікацією даних (тобто сутностей і їх атрибутів), зв'язків між даними і встановлює обмеження, що накладаються на збережені дані. Розроблювач логічної бази даних повинний мати всебічне і повне розуміння структури даних організації і їх бізнес правил. Бізнес правила описують основні характеристики даних з *погляду організації.* Нижче приводяться приклади типових бізнес правил:

4. Будь який співробітник не може відповідати одночасно більш ніж за десять об'єктів нерухомості що здаються в оренду чи продаються.
5. Будь який співробітник не має права продавати чи здавати в оренду свою власну нерухомість.
6. Довірена особа не може виступати одночасно і як покупець, і як продавець нерухомості.

Для ефективної роботи розроблювач логічної бази даних повинен якомога раніше включити всіх передбачуваних користувачів бази даних у процес створення моделі даних і його робота поділяється на два етапи.

1. Концептуальне проектування бази даних, що зовсім не залежить від таких деталей її втілення, як конкретна цільова СУБД, програми, мови програмування чи будь-якої іншої фізичної характеристики.
2. Логічне проектування бази даних, що проводиться з урахуванням особливостей обраної моделі даних: реляційної, мережної, ієрархічної чи об'єктно-орієнтованій.

Розроблювач фізичної бази даних одержує готову логічну модель даних, займається її фізичною реалізацією, у тому числі:

- перетворенням логічної моделі даних у набір таблиць і обмежень цілісності даних;
- вибором конкретних структур збереження і методів доступу до даних, що забезпечують необхідний рівень продуктивності, при роботі з базою даних;
- проектуванням будь-яких необхідних мір захисту даних.

Багато етапів фізичного проектування бази даних у значній мірі залежать від обраної цільової СУБД, а тому може існувати кілька різних способів реалізації необхідної схеми. Якщо **концептуальне і логічне** проектування бази даних відповідає на запитання "що?", то **фізичне** проектування відповідає на запитання "як?". Для рішення цих задач вимагаються різні навички роботи, якими найчастіше володіють різні люди.

1.6.3. Прикладні програмісти

Відразу після створення бази даних варто приступити до розробки програм, що надають користувачам необхідні їм функціональні можливості. Саме цю роботу і виконують **прикладні програмісти.** Звичайно прикладні програмісти працюють на основі специфікацій, створених системними аналітиками. Як правило, кожна програма містить деякі оператори, що вимагають від СУБД виконання визначених дій з базою даних - наприклад, такі як витяг, вставка, чи відновлення видалення даних. Як уже згадувалося в попередньому розділі, ці програми можуть створюватися на різних мовах програмування третього чи четвертого покоління.

1.6.4. Користувачі

Користувачі є клієнтами бази даних - вона проектується, створюється і підтримується для того, щоб обслуговувати їх інформаційні потреби. Користувачів можна класифікувати за способами використання ними системи.

Наївні користувачі звичайно навіть і не підозрюють про наявність СУБД. Вони звертаються до бази даних за допомогою спеціальних програм які дозволяють у максимальній мірі спростити операції, які вони виконують. Такі користувачі ініціюють виконання операцій у бази даних, способом введення найпростіших команд чи вибираючи команди меню. Це значить, що таким користувачам не потрібно нічого знати про базу даних СУБД.

Досвідчені користувачі. Досвідчені кінцеві користувачі, що знайомі зі структурою бази даних і можливостями СУБД. Для виконання необхідних операцій вони можуть використовувати таку мову запитів високого рівня, як SQL. А деякі досвідчені користувачі можуть навіть створювати власні прикладні програми.

1.7. Переваги і недоліки СУБД

СУБД володіють як багатообіцяючими потенційними перевагами, так і недоліками, які ми коротко розглянемо далі.

1.7.1. Переваги:

Контроль за надмірністю даних

Як уже говорилося раніше, традиційні файлові системи неощадливо витрачають зовнішню пам'ять, зберігаючи ті самі дані в декількох файлах. При використанні бази даних, навпаки, є спроба виключити надмірність даних за рахунок інтеграції файлів, щоб уникнути збереження декількох копій того самого елемента інформації. Однак цілком надмірність інформації в базах даних не виключається, а лише контролюється її ступінь.

Несуперечність даних

Усунення надмірності даних чи контроль над ними дозволяє скоротити ризик виникнення суперечливих станів. Якщо елемент даних зберігається в базі тільки в одному екземплярі, то для зміни його значення буде потрібно виконати тільки одну операцію відновлення, причому нове значення стане доступним відразу всім користувачам бази даних.

Більше корисної інформації при одному обсязі збережених даних

Завдяки інтеграції робітників даної організації на основі тих же даних можна одержувати додаткову інформацію. Наприклад, у показаній файловій системі співробітникам відділу контрактів невідомі власники зданих в оренду об'єктів. Аналогічно, співробітники відділу реалізації не мають повних зведень про договори про оренду. При інтеграції цих файлів у загальній базі співробітники відділу контрактів одержують доступ до зведень про власників, а співробітники відділу реалізації - до зведень про договори оренди.

Спільне використання даних

Файли звичайно належать окремим особам чи цілим відділам, що використовують їх у своїй роботі. У той же час, база даних належить всій організації в цілому і може спільно використовуватися всіма зареєстрованими користувачами. При такій організації роботи більша кількість користувачів може працювати з великим обсягом даних. Більш того, при цьому можна створювати нові програми на основі вже існуючої в базі даних інформації і додавати в неї тільки ті дані, що у даний момент ще не зберігаються в ній, а не визначати заново вимоги до всіх даних, необхідним новій програмі.

Підтримка цілісності даних

Цілісність бази даних означає коректність і несуперечність збережених у ній даних. Цілісність звичайно описується за допомогою **обмежень,** тобто правил підтримки несуперечності, що не повинні порушуватися в базі даних. Обмеження можна застосовувати до елементів даних усередині одного запису чи до зв'язків між записами. Наприклад, обмеження цілісності може говорити, що зарплата співробітника повинна перевищувати певне значення чи, що в записі з даними про співробітника номер відділення, у якому він працює, повинний відповідати реально існуючому відділенню

компанії. Таким чином, інтеграція даних дозволяє АБД задавати вимоги по підтримці цілісності даних, а СУБД застосовувати їх.

Підвищена безпека

Безпека бази даних полягає в захисті бази даних від несанкціонованого доступу з боку користувачів. Без залучення відповідних засобів безпеки інтегровані дані стають більш уразливими, чим дані у файловій системі. Однак інтеграція дозволяє АБД визначити необхідну систему безпеки бази даних, а СУБД втілити її в дію. Система забезпечення безпеки може бути виконана у формі облікових імен і паролів для ідентифікації користувачів, що зареєстровані в цій базі даних. Доступ до даних з боку зареєстрованого користувача може бути обмежений тільки деякими операціями (витягом, вставкою, відновленням і видаленням). Наприклад, АБД може бути надане право доступу до всіх даних у базі даних, менеджеру відділення компанії - до всіх даних, що відносяться до його відділення, а інспектору відділу реалізації - лише до всіх даних про нерухомість, у результаті чого він не буде мати доступу до конфіденційних даних, наприклад, про зарплату співробітників.

Застосування стандартів

Інтеграція дозволяє АБД визначати і застосовувати необхідні стандарти. Наприклад, стандарти відділу і організації, державні і міжнародні стандарти можуть регламентувати формат даних при обміні ними між системами, угоди про імена, форму представлення документації, процедури відновлення і правила доступу.

Підвищення ефективності з ростом масштабів системи

Комбінуючи всі робочі дані організації в одній базі даних і створюючи набір програм, що працюють з одним джерелом даних, можна домогтися істотної економії засобів. У цьому випадку бюджет, що звичайно виділявся кожному відділу для розробки і підтримки їх власних файлових систем, можна об'єднати з бюджетами інших відділів (з більш низькою загальною вартістю), що дозволить домогтися підвищення ефективності приросту масштабів виробництва.

Можливість знаходження компромісу для суперечливих вимог

Потреби одних користувачів чи відділів можуть суперечити потребам інших користувачів. Оскільки база даних контролюється АБД, він може приймати рішення про проектування і спосіб використання бази даних, при яких наявні ресурси всієї організації в цілому будуть використовуватися щонайкраще. Ці рішення забезпечують оптимальну продуктивність для найважливіших програм, причому найчастіше за рахунок менш критичних.

Підвищення приступності даних і їхньої готовності до роботи

Дані, що перетинають кордони відділів, у результаті інтеграції стають безпосередньо доступними кінцевим користувачам. Потенційно це підвищує функціональність системи, що, наприклад, може бути використане для більш якісного обслуговування кінцевих користувачів чи клієнтів організації. У багатьох СУБД передбачені мови запитів чи інструменти для створення звітів, що дозволяють користувачам задавати непередбачені заздалегідь питання і майже негайно одержувати необхідну інформацію на своїх терміналах, не прибігаючи до допомоги програміста, що для витягу цієї інформації з бази даних необхідно було б створити спеціальне програмне забезпечення.

Поліпшення показників продуктивності

Як уже згадувалося вище, СУБД передбачає багато стандартних функцій, які програміст повинен самостійно реалізувати в програмах для файлових систем. На базовому рівні СУБД забезпечує всі низько рівневі процедури роботи з файлами, які, як правило, виконують програми. Наявність цих процедур дозволяє програмісту сконцентруватися на розробці більш спеціальних, необхідних користувачам функцій, не піклуючись про подробиці їх втілення на більш низькому рівні. У багатьох СУБД також передбачене середовище розробки четвертого покоління з інструментами, що спрощують

створення програм баз даних. Результатом є підвищення продуктивності роботи програмістів і скорочення часу розробки нових програм з відповідною економією засобів.

Спрощення супроводу системи за рахунок незалежності від даних

У файлових системах опис даних і логіка доступу до даних вбудовані і кожна програма стає залежною від даних. У СУБД підхід інший: описи даних відділені від програм, а тому програми захищені від змін в описах даних. Ця особливість називається **незалежністю від даних.** Наявність незалежності програм від даних значно спрощує обслуговування і супровід програм, що працюють з базою даних.

Поліпшене керування паралельністю

У деяких файлових системах при одночасному доступу до одного і того ж файлу двох користувачів може виникнути конфлікт двох запитів, в результаті якого буде втрата інформації чи втрата її цілісності. У свою чергу, у багатьох СУБД передбачена можливість паралельного доступу до бази даних, що гарантує відсутність подібних проблем.

Розвинені служби резервного копіювання і відновлення

Відповідальність за забезпечення захисту даних від збоїв апаратного і програмного забезпечення у файлових системах покладається на користувача. Так, може знадобитися щодобово виконувати резервне копіювання даних. При цьому у випадку збою може бути відновлена резервна копія, але результати роботи, виконаної після резервного копіювання, будуть втрачені, і дану роботу буде потрібно виконати заново. У сучасних СУБД передбачені засоби скорочення обсягу втрат інформації від виникнення різних збоїв.

1.7.2. Недоліки:

Складність

Забезпечення функціональності, який повинна володіти кожна СУБД, супроводжується значним ускладненням програмного забезпечення СУБД. Щоб скористатися всіма перевагами СУБД, проектувальники і розроблювачі даних, адміністратори даних і адміністратори баз даних, а також кінцеві користувачі повинні добре розуміти функціональні можливості СУБД. Нерозуміння принципів роботи системи може привести до невдалих результатів проектування, що буде мати самі серйозні наслідки для всієї організації або установи.

Розмір

Складність і широта функціональних можливостей приводить до того, що СУБД стає надзвичайно складним програмним продуктом, що може займати багато місця на диску і вимагати великого обсягу оперативної пам'яті для ефективної роботи.

Вартість СУБД

У залежності від наявного обчислювального середовища і необхідних функціональних можливостей, вартість СУБД може варіювати в дуже широких межах. Наприклад, СУБД для одного користувача для персонального комп'ютера може коштувати не багато. Однак велика СУБД для багатьох користувачів для мейнфреймів, що обслуговує сотні користувачів може бути надзвичайно дорогою: від 100.000 до 500.000 доларів. Крім того, варто врахувати щорічні витрати на супровід системи, що складають деякий відсоток від її загальної вартості.

Додаткові витрати на апаратне забезпечення

Для задоволення вимог, пропонованих до дискових накопичувачів з боку СУБД і бази даних, може знадобитися придбати додаткові пристрої збереження інформації. Крім того, для досягнення необхідної продуктивності може знадобитися більш могутній комп'ютер, який, можливо, буде працювати тільки із СУБД. Придбання іншого додаткового апаратного забезпечення приведе до подальшого росту витрат.

Витрати на перетворення

У деяких ситуаціях вартість СУБД і додаткового апаратного забезпечення може виявитися несуттєвою в порівнянні з вартістю перетворення існуючих програм для роботи з новою СУБД і новим апаратним забезпеченням. Ці витрати також включають вартість підготовки персоналу для роботи з новою системою, а також оплату послуг фахівців, що

будуть робити допомогу в перетворенні і запуску нової системи. Усе це є однією з основних причин, по якій деякі організації залишаються прихильниками колишніх систем і не хочуть переходити до більш сучасних технологій керування базами даних.

Продуктивність

Звичайно файлова система створюється для деяких спеціалізованих програм, наприклад для оформлення рахунків, а тому її продуктивність може бути дуже висока. Однак СУБД призначені для рішення більш загальних задач і обслуговування відразу декількох програм, а не якоїсь однієї з них. У результаті багато програм у новому середовищі будуть працювати не так швидко, як колись.

Більш серйозні наслідки при виході системи з ладу

Централізація ресурсів підвищує уразливість системи. Оскільки робота всіх користувачів і програм залежить від готовності до роботи СУБД, вихід з ладу одного з її компонентів може привести до повного припинення всієї роботи організації.

Питання для контролю

1. Дайте визначення поняття „**База даних (БД)”**.
2. Дайте визначення поняття „Система управління базами даних (СУБД)”.
3. З якою метою використовується системний каталог?
4. Як Ви розумієте термін **«абстрагування даних»**?
5. Дайте визначення поняттям «сутність», «атрибут», «зв'язок».
6. З якою метою в СУБД використовується мова визначення даних (DDL)?
7. Які операції у СУБД виконуються за допомогою мови керування даними (DML)?
8. У чому різниця **процедурних** і **непроцедурних** мов DML?
9. Розшифруйте абревіатуру SQL та поясніть її значення.
10. З якою метою у СУБД використовуються відображення?
11. Перелічите та дайте стислий опис основних компонентів середовища СУБД.
12. Назвіть основні операції, які у СУБД покладаються на процедури.
13. Перелічите та коротко опишіть перелік обов’язків кожної групи користувачів СУБД.
14. У чому основна різниця між Адміністратором даних (АД) та Адміністратором баз даних (АБД)?
15. Чим відрізняються обов’язки розроблювача логічної бази даних та розроблювача фізичної бази даних?
16. Опишіть основні переваги використання СУБД.
17. Які недоліки або проблеми виникають при використання СУБД?