МІНІСТЕРСТВО ВНУТРІШНІХ СПРАВ УКРАЇНИ

**Харківський національний
університет внутрішніх справ**

Кафедра інформаційних технологій факультету № 4

ТЕКСТ ЛЕКЦІЇ

**з навчальної дисципліни
«Теорія розподілених інформаційних ресурсів, захист баз даних»
вибіркових компонент освітньої програми
другого (магістерського) рівня вищої освіти**

125 «Кібербезпека» (Безпека інформаційних та комунікаційних систем)

за темою – «Вступ до систем баз даних. Основні відомості про бази даних »

Харків 2018

ЗАТВЕРДЖЕНО
Науково-методичною радою
Харківського національного
університету внутрішніх справ
Протокол від _____ № ___

СХВАЛЕНО
Вченою радою
факультету № 4
Протокол від _____ № __

ПОГОДЖЕНО
Секцією Науково-методичної ради
ХНУВС з технічних дисциплін
Протокол від _____ № ___

Розглянуто на засіданні кафедри інформаційних технологій факультету № 4 *(протокол від _____ № ___).*

Розробники:

1. Доцент кафедри інформаційних технологій факультету № 4, кандидат технічних наук, доцент Тулупов В.В.

Рецензенти:

1. Професор кафедри кібербезпеки факультету № 4, кандидат технічних наук, доцент Носов В.В.
2. Завідувач кафедри систем інформації Національного технічного університету «Харківський політехнічний інститут», д.т.н., професор Серков О.А.

План лекції

1. Файлові системи баз даних
2. Історія розвитку БД та СУБД

Рекомендована література:

Основна:

1. Дейт К.Дж. Введение в системы баз данных. 6-е издание. - Киев – Москва: Діалектика, 1998. - 784 с.

2. Хансен Г., Хансен Дж. Базы данных: разработка и управление. – Москва: Бином, 1999. - 700 с. Пер. с англ.

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.

Додаткова:

4. Д. Кренке. Теория и практика построения баз данных. 8-е изд. – СПб.: Питер, 2003. – 800 с.: ил. – (Серия «Классика Conputer Science»).

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.

6. 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

7. Кузнецов С.Д. Основы современных баз данных. Информационно-аналитические материалы. - http://www.citmgu.ru/

Текст лекції

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 забронювані 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.

Таким 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:

1. поділ і ізоляція даних;
2. дублювання даних;
3. залежність від даних;
4. 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.

Питання для контролю:

1. Доповніть список прикладів використання баз даних, який наведено на початку лекції, своїми прикладами та обґрунтуйте необхідність використання СУБД для них.
2. Який підхід використовувався для перших систем обробки баз даних?
3. Поясніть, у чому особливість використання файлових систем баз даних? Звідки пішла їх назва?
4. З яких компонентів, як правило, складається кожний файл бази даних у файлових системах? Наведіть приклад структури файлу бази даних **Власники**.
5. Перерахуйте основні недоліки, які властиві файловим системам баз даних.
6. Поясніть як Ви розумієте вираз **«поділ і ізоляція даних»** у файлових системах?
7. У чому сутність **«дублювання даних»** у файлових системах?
8. Коли ми використовуємо термін **«залежність від даних»**, то ЩО залежить від даних з Вашої точки зору?
9. Які операції (покрокові) необхідно виконати для зміни структури файлу **Об'єкт нерухомості** у спеціальній програмі (дивись приклад з тексту лекції)?

10. Файловим системам властива **«несумісність файлів»**. З ЧИМ або ЯК несумісні файли баз даних?
11. Який проект вперше використав систему управління базами даних?
12. Назвіть першу комерційну СУБД.
13. Ким у перше було розроблено стандарти мов роботи з даними?
14. Які системи вважають СУБД першого покоління?
15. Ким та коли було запропоновано основні положення реляційної моделі даних?
16. Назвіть декілька прикладів сучасних реляційних СУБД та фірм, які їх розробили.
17. Які СУБД можна віднести до систем другого покоління?
18. Які СУБД можна віднести до систем третього покоління?