МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Ю.С.Рамський, Г.Ю.Цибко

ПРОЕКТУВАННЯ І ОПРАЦЮВАННЯ БАЗ ДАНИХ

Посібник для вчителів

ЗМІСТ

ВСТУП

Інформатизація суспільства і пов'язане з нею широке розповсюдження обчислювальної техніки, засобів комунікації, методів опрацювання інформації вимагають удосконалення змісту підготовки спеціалістів практично всіх галузей, оновлення складу навчальних дисциплін, спрямованого на гуманізацію навчального процесу і гуманітаризацію освіти. Підготовка висококваліфікованих педагогічних кадрів, які мають достатній рівень інформаційної культури, є найважливішим напрямком розв'язання задач інформатизації освіти.

В умовах швидких змін і постійного удосконалення засобів інформаційних технологій особливої уваги заслуговує проблема вивчення теоретичних основ інформатики, з якого має починатися науково і методично обґрунтоване впровадження нових інформаційних технологій у навчальний процес загальноосвітньої школи та вищого навчального закладу.

Знання теоретичних положень, які покладені в основу функціонування того чи іншого програмного засобу, дозволить ефективніше використовувати його в професійній діяльності, полегшить адаптацію до його нових версій або інших засобів подібного призначення.

Проблема відриву теорії від практики, яка нерідко виникає при вивченні питань теоретичної інформатики, зокрема баз даних у школі і у вищому педагогічному навчальному закладі, пов'язана з використанням традиційної методики вивчення СУБД, що переважно ставить за мету навчити студентів лише вводити дані у базу, виконувати її нескладні модифікації і формулювати запити до розробленої і заповненої БД у середовищі конкретної СУБД. Проте і у школі, і у вищому педагогічному навчальному закладі слід акцентувати увагу на теоретичних основах побудови і опрацювання БД. Даний посібник призначений допомогти вчителям і студентам вирішити цю проблему.

В навчально-методичному посібнику "Проектування і опрацювання баз даних" розгляд теоретичних принципів, на основі яких здійснюється робота з базами даних, поєднується з ілюстрацією

практичної реалізації зазначених положень на прикладі застосування популярних СУБД, орієнтованих на роботу під управлінням операційних систем MS-DOS і Windows: Borland Paradox 3.5(4.5) і Microsoft Access 97 (2000).

В роботі розглянуті три групи питань, що стосуються проектування і опрацювання баз даних: 1) основні відомості про побудову і функціонування банків даних (в рамках цієї групи розглядаються питання загальної структури банків даних, словник даних, адміністрування бази даних, захист даних, цілісність і несуперечливість даних, рівні подання даних); 2) моделювання даних (на основі об'єктно-зв'язної моделі), проектування баз даних на основі реляційної моделі даних (розглянуті питання ключів відношення, посилальної цілісності, нормалізації відношень); опрацювання відношень засобами реляційної алгебри, мов SQL і QBE; 3) системи управління базами даних Paradox і Access як засіб практичної реалізації запропонованих теоретичних положень. Кожна група питань супроводжується значною кількістю прикладів, контрольними запитаннями і завданнями.

Значна увага приділена процесу проектування бази даних, який включає окреслення предметної галузі, визначення її об'єктів, дані про які мають бути відображені в базі; їх властивостей - атрибутів; зв'язків між виділеними об'єктами. Для ознайомлення з питаннями розробки і опрацювання баз даних використовується реляційна модель даних. Ця модель підтримується більшістю сучасних СУБД, є нескладною для розуміння і вивчення, має для своєї підтримки розвинутий математичний апарат і характерна тим, що вся інформація про об'єкти предметної галузі, їх атрибути і зв'язки подається в однорідній формі - у вигляді відношень бази даних (таблиць). Реляційна модель є досить потужним засобом формалізованого опису складних предметних галузей, тому її вивчення є потрібним для розуміння процесів опрацювання даних сучасними СУБД.

Ґрунтовне вивчення реляційної моделі даних, засвоєння прийомів вдалого проектування БД, ознайомлення з мовами опису даних і маніпулювання ними на прикладі конкретної СУБД, яка використовується не як мета, а як засіб, дозволить глибоко усвідомлювати процеси опрацювання даних і не відчувати труднощів при переході до інших програмних продуктів аналогічного типу.

Матеріал посібника може бути використаний при викладанні курсу “Бази даних та інформаційні системи” у вищому педагогічному навчальному закладі, а також відповідних розділів курсу інформатики вищого педагогічного навчального закладу і середньої школи.

Для поглибленого вивчення матеріалу доцільно розглянути всі розділи посібника. Якщо основна увага приділяється практичному застосуванню програмних продуктів (зокрема для гуманітарних спеціальностей), можна обмежитись розглядом розділу 5 “Основи роботи з сучасними системами управління базами даних”, матеріал якого є методичною розробкою лекційно-практичного курсу “Системи управління базами даних Paradox і Access”. Серед об’єктів бази даних в розглянутих СУБД акцент робиться на вивченні таблиць і запитів як основних засобів для зберігання структурованих даних і їх опрацювання. В залежності від конкретних умов навчального процесу дані програми можуть розглядатися разом або окремо.

Розділ 1

БАНКИ І БАЗИ ДАНИХ

1.1. ОСНОВНІ ПОНЯТТЯ

Життя і успішна практична діяльність сучасної людини значною мірою залежить від ефективної організації обміну інформацією. Інформаційні процеси реалізуються в усіх сферах людської діяльності: науці, техніці, економіці, технології, медицині тощо. Інформація і дані все частіше розглядаються як життєво важливі ресурси, які повинні бути організовані так, щоб ними можна було легко користуватися.

Великі обсяги інформації практично неможливо опрацьовувати без спеціальних засобів машинної обробки. Останнім часом широкого розповсюдження набули автоматизовані інформаційні системи: інформаційно-довідкові, інформаційно-пошукові, інформаційно-логічні і т.п. Всі вони призначені для реєстрації, зберігання і опрацювання інформації з метою пошуку і видачі відповідей на запити користувачів. В більшості випадків автоматизовані інформаційні системи розробляються як банки даних. Банки даних є одним з основних компонентів автоматизованих систем різних типів і рівнів. Вони створюються для багатьох галузей і сфер суспільного життя: планування, обліку, управління, статистики, охорони здоров'я тощо.

Перш ніж перейти до більш детального розгляду питань, пов'язаних з організацією і функціонуванням банків даних, введемо деякі основні поняття.

Під інформацією будемо розуміти будь-які відомості про певну подію, сутність, процес і т.п., які є об'єктом деяких операцій: сприйняття, передавання, перетворення, зберігання, використання.

Під даними будемо розуміти інформацію, фіксовану (закодовану) в певній формі, придатній для наступної обробки, зберігання і використання.

При розробці автоматизованих інформаційних систем, у відповідності з двома поняттями – "інформація" і "дані", в автоматизованих інформаційних системах розрізняють два аспекти: інфологічний і датологічний.

Інфологічний аспект стосується питань, пов'язаних зі смисловим змістом даних незалежно від способів їх подання в пам'яті системи;

датологічний – питань подання даних в пам'яті інформаційної системи.

На етапі інфологічного проектування системи розв'язуються задачі:

- визначаються об'єкти і явища реального світу, інформація про які має бути накопичувана і опрацьовувана в системі;

- визначаються суттєві характеристики і взаємозв'язки цих об'єктів і явищ;

- уточнюється вищеназвана інформація для введення в інформаційну систему.

На етапі інфологічного проектування виділяється предметна область–частина реального світу, яка визначає інформаційні потреби системи.

Предметною областю називають сукупність об'єктів, предметів реального світу, які розглядаються в межах даного контексту (теорії, моделі, банку інформації і ін.).

На етапі датологічного проектування системи розв'язуються наступні задачі:

- розробляються форми подання інформації, які відповідають наявним засобам сприйняття і обробки;

- наводяться моделі і методи подання і перетворення даних;

- формулюються правила смислової інтерпретації даних (семантичні правила).

Банк даних є сукупністю спеціальних методів і засобів (математичних, інформаційних, програмних, мовних, організаційних і технічних) для підтримки динамічної інформаційної моделі предметної області з метою забезпечення інформаційних запитів користувачів.

Банки даних є одними з найважливіших складових різноманітних автоматизованих систем: управління, довідкових систем різного профілю, систем автоматизованого проектування тощо.

Організація банку даних дозволяє не тільки використовувати інформацію, що зберігається в системі, а і змінювати при необхідності її структуру.

Банк даних включає до свого складу дві основні компоненти: базу даних (БД) і систему управління базами даних (СУБД).

Базою даних називають сукупність взаємопов'язаних даних деякої предметної області, які зберігаються в пам'яті ЕОМ і організовані таким чином, що можуть використовуватись для розв'язання багатьох задач багатьма користувачами.

БД є датологічним поданням інформаційної моделі предметної області.

БД розробляються таким чином, щоб існувала можливість формулювати запит і отримувати потрібну інформацію без трудомісткого написання програм.

Системою управління базами даних називають сукупність програм і мовних засобів, за допомогою яких реалізується централізоване управління даними в базі, доступ до них і забезпечується взаємодія бази з прикладними програмами.

В кожній СУБД перш за все є транслятори або інтерпретатори з мови опису даних (МОД) і мови маніпулювання даними (ММД).

Мова опису даних є непроцедурною мовою високого рівня, призначеною для опису вмісту і структури бази даних або її частини. За допомогою МОД виконується опис типів даних, які підлягають збереженню в базі або виборці з бази, їх структур і взаємозв'язків.

Мова маніпулювання даними (або мова запитів до БД) є засобом, який застосовується користувачами або прикладними програмістами для виконання операцій над даними. ММД звичайно подається системою команд маніпулювання даними, серед яких можуть, наприклад, міститися команди:

- вибрати з БД конкретне дане за його найменуванням;
- вибрати з БД всі дані певного типу, значення яких задовольняють заданим умовам;
- записати нові дані до бази;
- вилучити певні дані з бази;
- знайти певні дані в БД і змінити їх (оновити дані) тощо.

В сучасних системах обробки даних властивості МОД і ММД можуть поєднуватись в одній мові (наприклад, мова SQL Structured Query Language–структурована мова запитів).

1.2. СЛОВНИК ДАНИХ

Концепція баз даних передбачає інтеграцію даних, зростання сумісного використання даних різними прикладними програмами і скорочення надмірного зберігання одних і тих самих даних. Для успішного здійснення таких процесів потрібне централізоване управління вмістом баз даних.

Важливим засобом централізації управління даними є словник даних.

Словник даних є спеціальною системою у складі банку даних, яка призначена для зберігання і опрацювання єдиним чином організованої і централізованої інформації про всі ресурси даних конкретного банку.

В словнику містяться відомості про об'єкти певної предметної області, їх властивості і відношення між ними, відомості про дані, які зберігаються в базі (найменування даних, їх структура, зв'язки з іншими даними), про їх можливі значення і формати подання, про джерела їх виникнення, про коди захисту, обмеження доступу до даних з боку користувачів тощо.

Словник даних покликаний сприяти зменшенню надмірності і суперечливості даних, зберігати їх централізований опис, який дозволяє централізовано вводити в систему нові типи даних або змінювати існуючі чи вилучати застарілі. Крім того, словник дозволяє користувачам системи і адміністратору бази даних користуватись єдиною термінологією при розв'язуванні задач, пов'язаних з обслуговуванням запитів в банку даних. Будь-які зміни, які відбуваються в БД, легко визначаються з допомогою словника.

Словник даних використовується кінцевими користувачами при роботі з системою, прикладними програмістами–при написанні програм, системними програмістами – в процесі розвитку системи.

Словник даних переважно описує базу даних у вигляді кількох фізичних таблиць з логічними зв'язками між ними.

1.3. АДМІНІСТРАЦІЯ БАЗИ ДАНИХ

Колектив спеціалістів, які забезпечують функціонування автоматизованого банку даних, називають адміністрацією бази даних. В залежності від складності інформаційної системи група адміністрації може складатися як з одного, так і з кількох осіб.

Звичайно до складу адміністрації бази даних входить адміністратор, аналітики і програмісти.

Адміністратор – це спеціаліст, який обізнаний з інформаційними потребами кінцевих користувачів, працює в тісному контакті з користувачами і відповідає за визначення, завантаження, захист і ефективність експлуатації бази даних. Адміністратор повинен мати повне уявлення про зберігання і використання даних. Зазначимо, що для адміністратора словник даних є основним джерелом інформації для прийняття певних рішень.

Програмісти розробляють сервісні програми і інші програмні засоби, які забезпечують обробку інформації і розв'язування задач на ЕОМ в рамках ОС і СУБД.

Аналітик будує інформаційну, зокрема математичну модель певного проблемного середовища, використовуючи для цього необхідні математичні методи і методи моделювання. Функцією аналітика є переведення задачі кінцевого користувача в деяку вихідну формальну модель.

1.4. ЗАХИСТ ДАНИХ

З появою централізованих баз даних виникла необхідність в захисті даних. Термін захист даних означає запобігання несанкціонованого або випадкового доступу до даних, їх зміни або руйнування з боку користувачів, запобігання зміни або руйнування даних при збоях апаратних і програмних засобів і помилках в роботі співробітників групи експлуатації.

Захист даних може виконувати дві функції: забезпечення безпеки даних і забезпечення секретності даних.

Під функцією безпеки розуміють захист даних від ненавмисного доступу до даних і можливості їх псування з боку користувачів або осіб, що виконують експлуатацію, а також від збоїв в апаратурі або програмних засобах. Тому забезпечення функції безпеки вважається внутрішньою задачею банку даних, оскільки пов'язано з забезпеченням його нормального функціонування.

Під функцією секретності розуміється захист даних від доступу до них користувачів або осіб, які не мають на це прав. Забезпечення секретності вимагає перш за все поділити дані в базі на загальнодоступні і такі, які повинні використовуватись конфіденційно, тобто вони або самі безпосередньо містять секретну інформацію, або її можна отримати з цих даних за допомогою

спеціальної алгоритмічної обробки. Вирішення цього питання виходить за рамки банку даних і знаходиться в компетенції юридичних органів або адміністрації підприємства, для якого створюється банк даних.

1.5. ЦІЛІСНІСТЬ ДАНИХ

Проблема цілісності полягає в забезпеченні правильності даних в базі в будь-який момент часу. Цей аспект стосується захисту даних від ненавмисних помилок і запобігання останнім. Цим проблема цілісності відрізняється від проблеми безпеки, в решті аспектів ці два питання тісно пов'язані. Існують два основні види обмежень цілісності, які мають підтримуватись СУБД:

– структурні обмеження. Ці обмеження в багатьох випадках задаються функціональними залежностями і відповідним чином контролюються;

– обмеження реальних значень даних, які зберігаються в базі. Такі обмеження переважно вимагають, щоб значення полів належали певним діапазонам значень, або задають деяке арифметичне співвідношення, якому повинні задовольняти значення полів.

Існують і інші обмеження цілісності. Наприклад, обмеження на умови виконання паралельних операцій над даними в базі, обмеження типу "старий-новий", коли база переходить в новий стан тощо.

В загальному випадку цілісність може бути порушена при збої обладнання, програмній помилці, помилці людини-оператора, помилках у вхідних даних і т.п.

Основна ідея забезпечення обмежень цілісності даних полягає в тому, щоб використовувати мову маніпулювання даними для задання обмежень. ММД реальних систем дозволяють тією чи іншою мірою підтримувати обмеження цілісності.

Розділ 2

ПРОЕКТУВАННЯ БАЗ ДАНИХ

2.1. ТРИ РІВНІ ПОДАННЯ ДАНИХ

При описі предметної області дані прийнято подавати у вигляді трьохрівневої схеми: концептуальне подання (з точки зору адміністратора), зовнішнє (з точки зору кінцевого користувача і прикладного програміста) і внутрішнє (з позицій системного програміста).

Зовнішнє подання даних є сукупністю вимог до даних деякої конкретної задачі (або програми). Користувацьке представлення деякої задачі (проблеми) відображає конкретні інформаційні потреби. Зовнішнє подання з точки зору прикладного програміста відображає елементи даних і їх взаємозв'язки, наявність яких необхідне в базі даних для розв'язування його задач.

Концептуальне подання даних пов'язане з відображенням знань про предметну область. Структура даних на концептуальному рівні називається концептуальною схемою і пов'язана з виявленням семантики даних. Елементарними одиницями концептуального представлення даних є три поняття: елементи (об'єкти, предмети, процеси) предметної області; властивості елементів предметної області; зв'язки між цими елементами і їх властивостями.

Внутрішнє (фізичне) подання виражає представлення даних системними програмістами і пов'язане з організацією зберігання даних на фізичних носіях інформації.

2.2. МОДЕЛЮВАННЯ ДАНИХ

Питання моделювання даних і предметної області є ключовими в теорії автоматизованих інформаційних систем. Основними видами моделей є 1) модель предметної області, 2) модель даних, 3) модель бази даних.

Існують два основні підходи до створення моделі предметної області (МПО). В рамках першого підходу МПО будується на основі аналізу і інтеграції інформаційних потреб користувачів банку даних. Другий підхід базується на аналізі самої предметної області і побудові МПО з урахуванням інформаційних потреб користувачів. На основі аналізу цих потреб розробник банку даних створює

модель предметної області, формалізації якої приводять до двох видів моделей:

1) інфологічної (яку називають також концептуальною), орієнтованої на користувача;

2) датологічної, орієнтованої на реалізацію в обчислювальному середовищі.

В банку даних при відображенні інфологічної МПО в середовище СУБД породжується одна з можливих датологічних моделей.

Моделлю даних називають сукупність правил породження структур даних в базі, допустимих операцій над ними, а також обмежень цілісності, яка визначає допустимі зв'язки, значення даних і послідовності їх зміни.

В банку даних переважно використовується одна з наступних трьох моделей даних: ієрархічна, мережева, реляційна. Ці моделі являють собою зразки типових структур даних, які відрізняються одна від одної певними особливостями. Ієрархічна модель даних – це модель деревовидних структур, мережева модель даних – модель орієнтованих графів. Реляційна модель базується на теоретико-множинному понятті відношення, тобто множині кортежів фіксованої довжини.

Практично не існує таких СУБД, які підтримують у чистому вигляді якусь одну модель даних. Це пов'язане з тим, що на практиці при датологічному проектуванні предметної області не вдається уникнути серйозних труднощів, пов'язаних, наприклад, з надмірністю (дублюванням) даних. Тому мови конкретних СУБД орієнтуються в основному на певну модель даних – ієрархічну, мережеву або реляційну, але при цьому в мовах реалізуються деякі можливості роботи з іншими типами моделей даних.

Модель бази даних (МБД) визначається як множина конкретних допустимих типів даних і відношень, конкретних обмежень, які визначають допустимі операції над об'єктами бази даних. МБД описується схемою БД, яка визначає її структуру, обмеження цілісності і управління доступом.

Розроблена адміністратором банку даних схема бази даних використовується в подальшому ним самим при виконанні його функцій по супроводженню бази даних, а також механізмами доступу до даних СУБД, в середовищі якої зберігається база даних.

2.3. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

Проектування бази даних є складний процес здійснення відображення:

“опис предметної області” – “схема внутрішньої моделі бази даних”.

Цей процес представляють послідовністю простіших процесів відображень між проміжними моделями даних.

Основні етапи проектування БД – інфологічне проектування і датологічне проектування. В останньому виділяють логічне і фізичне проектування.

Завданням інфологічного етапу проектування БД є отримання семантичних (смислових) моделей даних, які відображують інформаційний зміст конкретної предметної області. На цьому етапі здійснюється сприйняття реальної дійсності, абстрагування, вивчення і опис предметної області. Спочатку визначаються межі предметної області, відбувається абстрагування від несуттєвого для даного конкретного застосування банку даних. Визначаються об’єкти, їх властивості і зв’язки, суттєві для користувачів системи. Після цього вивчається предметна область, накопичуються знання про неї, які подаються в певній мовній системі. Переважно це неформалізований опис з використанням природної мови, математичних формул, діаграм зв’язків тощо. Виконується структуризація знань про предметну область: виділяються і класифікуються множини складових предметної області, стандартизується термінологія.

Потім на основі потреб користувачів створюється концептуальна інфологічна модель. На цьому етапі описуються запити до БД. Кожний запит співвідноситься з певним фрагментом предметної області. Формуються описи зовнішніх представлень предметної області, взаємно ув’язуються з концептуальною моделлю. Ці описи не повинні залежати від методів подання даних в конкретній СУБД.

При логічному проектуванні дані, виділені на попередньому етапі, перетворюються в таку форму, яка прийнята в СУБД. Отже, схема концептуальної моделі розробляється з використанням тільки тих типів моделей даних, які підтримуються обраною СУБД.

На фізичному етапі проектування вибирається раціональна структура зберігання даних і методів доступу до них, виходячи з методів і засобів, які надаються СУБД.

2.4. МОДЕЛЬ “ОБ’ЄКТ-АТРИБУТ-ЗВ’ЯЗОК”

На етапі інфологічного проектування БД широко використовується інформаційна модель предметної області, яка отримала назву ”об’єкт-атрибут-зв’язок”. Вона дозволяє моделювати об’єкти предметної області, яка відображується в банку даних, їх властивості і відношення між ними. Завдяки простоті і використанню природної мови модель “об’єкт-атрибут-зв’язок” використовується при проектуванні бази даних інформаційної системи також як засіб для отримання у майбутніх користувачів інформації про предметну область.

Основним призначенням неформальної моделі “об’єкт-атрибут-зв’язок” є моделювання і структурування інформації для подальшого використання інформаційною системою.

При побудові моделей типу “об’єкт-атрибут-зв’язок” для подання складових предметної області використовуються три основні структурні елементи: об’єкт, атрибут і зв’язок.

Об’єкт – це збірне поняття, деяка абстракція реально існуючого предмета, процесу або явища, про яке необхідно зберігати інформацію в системі. Об’єктами в моделях предметної області можуть виступати як матеріальні об’єкти (співробітники установи, вироби підприємства тощо), так і нематеріальні (опис деякого явища, опис структур даних, що використовуються системою тощо). В моделі використовуються поняття “тип об’єкта” (клас) і “екземпляр об’єкта” (власне об’єкт). Тип об’єкта визначає набір однорідних об’єктів, а поняття “екземпляр об’єкта” відноситься до конкретного об’єкта в наборі. Наприклад, типом об’єкта може бути СТУДЕНТ, а екземплярами ІВАНЕНКО І.І., ПЕТРЕНКО П.П. тощо.

Атрибутом називається пойменована характеристика об’єкта. Атрибут набуває значень з деякої визначеної множини. В моделі він є засобом, за допомогою якого визначаються властивості об’єкта. Наприклад, для опису властивостей об’єкта СТУДЕНТ можуть бути використані атрибути ПРІЗВИЩЕ, ІМ’Я, ПО_БАТЬКОВІ, РІК_НАРОДЖЕННЯ, ГРУПА. Для задання атрибута в моделі йому необхідно присвоїти ім’я, навести смисловий опис атрибута, визначити множину його допустимих значень і вказати його роль, тобто вказати, для чого він використовується.

Зв’язок – це засіб, за допомогою якого подаються відношення між об’єктами, які мають місце в предметній області. Тип зв’язку

розглядається між типами об'єктів, а конкретний екземпляр зв'язку певного типу існує між конкретними екземплярами типів об'єктів.

Між об'єктами можуть існувати бінарні зв'язки (зв'язки між двома об'єктами), тернарні зв'язки (між трьома об'єктами) і в загальному випадку – n-арні зв'язки. Найчастіше зустрічаються бінарні зв'язки. Розглянемо види бінарних зв'язків між екземплярами об'єктів різного типу. Розрізняють три види зв'язків між об'єктами, які позначаються: один-до-одного, один-до-багатьох і багато-до-багатьох.

Один-до-одного. Так називається зв'язок між об'єктами, при якому кожний об'єкт пов'язаний з одним і тільки одним іншим об'єктом. Розглянемо як приклад два об'єкти: СТУДЕНТ і СТУДКВИТОК. Між цими об'єктами існує найпростіший зв'язок. Відомо, що номер студентського квитка вибирають таким чином, щоб кожний студент мав свій унікальний номер квитка. І навпаки, кожному номеру студквитка відповідає тільки один студент. Отже, між об'єктами СТУДЕНТ і СТУДКВИТОК існує зв'язок типу один-до-одного, який зображений графічно на рис.1.

СТУДЕНТ ⟷ СТУДКВИТОК
Рис.1. Зв'язок 1:1

Один-до-багатьох (або багато-до-одного). Так називається зв'язок між об'єктами (атрибутами), при якому один об'єкт може співставлятися з одним із об'єктів з певної групи.

Наприклад, в одну кімнату студентського гуртожитку можна поселити одного або кількох студентів, але в той же час одного студента можна поселити в гуртожитку лише в одну кімнату. Таким чином, між двома атрибутами ІМ'Я_СТУДЕНТА і НОМЕР_КІМНАТИ існує зв'язок, названий багато-до-одного, а між атрибутами НОМЕР_КІМНАТИ і ІМ'Я_СТУДЕНТА існує зв'язок, названий один-до-багатьох (в одній кімнаті живе багато студентів). Для визначеності тут і далі будемо вважати, що атрибут ІМ'Я_СТУДЕНТА однозначно ідентифікує студента. Графічна інтерпретація розглянутих видів звязків подана на рис.2.

ІМ'Я_СТУДЕНТА ←⟷ НОМЕР_КІМНАТИ
НОМЕР_КІМНАТИ ⟷→ ІМ'Я_СТУДЕНТА
Рис.2. Зв'язки ∞:1, 1: ∞

Багато-до-багатьох . Найбільш складний вид зв’язку між об’єктами. Такий вид зв’язку може мати місце у співвідношенні об’єктів СТУДЕНТ-ВИКЛАДАЧ. Тут кожний студент вчиться у багатьох викладачів, і кожний викладач читає лекції багатьом студентам, що на схемі позначимо двома подвійними стрілками рис.3).

СТУДЕНТ $\leftarrow\!\leftarrow\!\longrightarrow\!\rightarrow$ ВИКЛАДАЧ

Рис.3. Зв’язок ∞:∞

Розділ 3

РЕЛЯЦІЙНА МОДЕЛЬ ДАНИХ

3.1. ОСНОВНІ ПОНЯТТЯ

Реляційна модель даних була запропонована в 1970 р. Коддом, який показав, що набір таблиць (або відношень) може бути використаний для моделювання взаємозв'язків між об'єктами реального світу і для зберігання даних про ці об'єкти.

В основі реляційної моделі використано поняття відношення, яке являє собою підмножину декартового добутку доменів. Доменом називають сукупність однотипних значень даних (наприклад, домен цілих чисел, домен імен) або множину допустимих значень, які можуть набувати атрибути об'єкта. Зазначимо, що декілька атрибутів можуть набувати значення з одного домену.

Декартовим добутком доменів $\mathbf{D_1, D_2, ..., D_k}$

$\mathbf{D = D_1 \times D_2 \times ... \times D_k}$,

де

$\mathbf{D_1 = \{ d_{1.1}, d_{1.2},..., d_{1.i1},..., d_{1.n1} \}}$;

$\mathbf{D_2 = \{ d_{2.1}, d_{2.2},..., d_{2.i2},..., d_{2.n2} \}}$;

.....

$\mathbf{D_k = \{ d_{k.1}, d_{k.2},..., d_{k.ik},..., d_{k.nk} \}}$,

називається множина всіх впорядкованих наборів (кортежів) довжини **k** (тобто тих, що складаються з **k** елементів – по одному з кожного домену $\mathbf{D_i}$):

$\mathbf{< d_{1.i1}, d_{2.i2},..., d_{k.ik} >}$

Наприклад, якщо $D_1 = \{A,B,C\}$, $D_2 = \{1,2\}$, $D_3 = \{x,y\}$

то k = 3, і відповідно $D = D_1 \times D_2 \times D_3$ = {<A,1,x>, <A,1,y>, <A,2,x>,<A,2,y>, <B,1,x>, <B,1,y>, <C,1,x>, <C,1,y>, <C,2,x>, <C,2,y>}.

Відношенням **R** на множинах $\mathbf{D_1, D_2, ..., D_k}$ називається підмножина декартового добутку $\mathbf{D_1 \times D_2 \times ... \times D_k}$.

Інакше кажучи, відношення **R**, визначене на множинах $\mathbf{D_1, D_2, ..., D_k}$ (причому не обов'язково, щоб ці множини були різними), є деяка підмножина кортежів довжини k: $\mathbf{< d_{1.i1}, d_{2.i2},..., d_{k.ik} >}$

таких, що $d_{1.i1} \in D_1$, $d_{2.i2} \in D_2$,..., $d_{k.ik} \in D_k$:

$R \in D_1 \times D_2 \times ... \times D_k$

Вкажемо в даному прикладі кілька відношень:
R_1 = {<A,1,x>,<A,2,y>};
R_2 = {<B,2,x>,<C,1,y>,<C,2,y>};
R_3 = {<A,1,x>,<A,1,y>,...,<C,2,x>,<C,2,y>};
R_4 = ∅.

Отже, відношення є множиною кортежів, і ця множина є підмножиною декартового добутку фіксованої кількості доменів. Компонентами кортежів є значення атрибутів. Для зручності опрацювання відношень атрибутам надаються імена. Значення кожного атрибута вибираються з певного домену.

Довжина кортежу називається його арністю і визначає арність відношення. Відношення арності 1 називають унарним, арності 2 – бінарним, арності 3 – тернарним, арності n – n-арним. Оскільки відношення є множиною, воно не повинно містити однакових кортежів.

В ряді випадків відношення зручно подавати таблицею, де заголовками колонок (стовпців) є імена атрибутів, а кожний рядок відповідає одному кортежу. Всі значення в певній колонці належать одному й тому самому домену. Подання розглянутих відношень таблицями наведено на рис.4.

R_1

A	1	x
A	2	y

R_2

B	2	x
C	1	y
C	2	y

R_3

A	1	x
A	1	y
A	2	x
A	2	y
B	1	x
B	1	y
B	2	x
B	2	y
C	1	x
C	1	y
C	2	x
C	2	y

Рис.4.Подання відношень R_1,R_2,R_3 таблицями

Список імен атрибутів називається схемою відношення. Якщо відношення називається **R**, а його атрибути мають імена $\mathbf{A_1,A_2,...,A_k}$, то схема відношення позначатиметься так: $\mathbf{R(A_1, A_2,..., A_k)}$.

Відношення можна розглядати і як абстрактне подання файла певного обмеженого типу. Такий файл складається з послідовності записів, по одному на кожний кортеж, причому однакові записи забороняються. Всі записи мають бути одного типу, повинні містити однакову кількість полів, і у відповідних полях різних записів повинна зберігатися інформація одного типу. Наприклад, файл зарплат службовців певної організації, який розглядається як відношення, може бути зображений так (рис.5):

1994	Петренко	150
1994	Сидоренко	200
1994	Іваненко	170
...	...	...
1997	Кравченко	350

Рис.5. Подання вмісту файлу таблицею

Відповідність між відношенням, файлом і таблицею можна подати так:

ВІДНОШЕННЯ	ФАЙЛ	ТАБЛИЦЯ
кортеж	запис	рядок
ім’я атрибута	ім’я поля	ім’я колонки
ім’я домена	тип поля	тип колонки

Далі відповідні поняття (“атрибут” і “поле”, “кортеж” і “запис” тощо) будуть вживатись рівноправно.

Реляційна база даних містить скінченну множину відношень. Схему реляційної бази даних можна подати у вигляді:

$R(A_{11}, A_{12},..., A_{k1})$;
$R(A_{21}, A_{22},..., A_{k2})$;
.....
$R(A_{n1}, A_{n2},..., A_{kn})$.

Розглянемо обмеження реляційної моделі. Основним обмеженням є неможливість подання у відношенні однакових кортежів. Це обмеження означає, що кожне відношення повинно мати принаймні один первинний ключ. Поняття ключа є важливим в реляційній моделі даних, тому розглянемо його докладніше.

3.2. КЛЮЧІ ВІДНОШЕННЯ

Ключем відношення називається підмножина атрибутів його схеми, сукупність значень яких однозначно ідентифікує кортеж у відношенні.

Наприклад, у відношенні СПІВРОБІТНИК(№_ПАСПОРТА, ПРІЗВИЩЕ, ІМ'Я, ПО_БАТЬКОВІ) ключем або ключовим полем є №_ПАСПОРТА, оскільки номер паспорта є унікальним для кожної людини. Ключем може бути і звичайне число – номер рядка в таблиці.

Часто виникають випадки, коли одного атрибута недостатньо для однозначної ідентифікації кортежу, тоді ключем є кілька зчіплених полів (конкатенація полів). Розглянемо, наприклад, відношення ЖИВ(ПРІЗВИЩЕ, МІСТО) (рис.6а), яке пов'язує людей і міста, в яких вони жили. Атрибут ПРІЗВИЩЕ не може бути ключовим, оскільки одна людина могла в різний час проживати в різних містах (див. кортежі <Коваль, Київ> і <Коваль, Чернігів>). Не може бути ключовим і атрибут МІСТО, бо в одному місті могло проживати кілька осіб (див., наприклад, кортежі <Петренко, Київ> і <Коваль, Київ>). Отже, в даному випадку ключ утворює комбінація прізвища особи і назви міста, тобто ключем є весь кортеж. Додавши до схеми відношення ЖИВ ще один атрибут, наприклад, РОКИ, який позначає кількість років, прожитих певною особою в даному місті, отримаємо відношення ЖИВ-У(ПРІЗВИЩЕ, МІСТО, РОКИ) (рис.6б). Ключем при цьому залишиться комбінація прізвища і міста, і ця комбінація однозначно визначає значення атрибута РОКИ. Кажуть, що кількість років функціонально залежить від прізвища і міста. Відношення ЖИВ-У є поданням функції двох аргументів: прізвища і міста, значенням якої є кількість років.

ЖИВ

ПРІЗВИЩЕ	МІСТО
Петренко	Київ
Петренко	Донецьк
Шевчук	Донецьк
Коваль	Київ
Коваль	Чернігів

а)

ЖИВ-У

ПРІЗВИЩЕ	МІСТО	РОКИ
Петренко	Київ	15
Петренко	Донецьк	5
Шевчук	Донецьк	20
Коваль	Київ	15
Коваль	Чернігів	5

б)

ОСОБА

ПРІЗВИЩЕ	ВІК	СТАТЬ	ЗРІСТ
Петренко	20	Ч	1.8
Шевчук	20	Ж	1.6
Коваль	36	Ч	1.7

в)

Рис.6.Подані таблицями відношення ЖИВ, ЖИВ-У, ОСОБА

Відношення, які містять ключ з кількох зчіплених полів і кілька атрибутів, що функціонально залежать від ключа, часто зустрічаються на практиці.

Розглянемо кілька прикладів відношень і їх ключів. Відношення ОСОБА(ПРІЗВИЩЕ, ВІК, СТАТЬ, ЗРІСТ), ключ – ПРІЗВИЩЕ (вважаємо, що атрибут ПРІЗВИЩЕ однозначно ідентифікує особу); відношення ВИВЧАЄ(СТУДЕНТ, ДИСЦИПЛІНА, ОЦІНКА), ключ СТУДЕНТ, ДИСЦИПЛІНА; ВІДНОШЕННЯ ГРУПА(НОМЕР_ГРУПИ, КУРС, КУРАТОР), ключ – НОМЕР_ГРУПИ, КУРС.

Отже, у відношенні можуть існувати кілька одиничних чи складених атрибутів, які однозначно ідентифікують кортеж відношення. Такі атрибути називаються можливими ключами. Один з них вибирається як первинний ключ для забезпечення доступу до кортежу.

Наприклад, у відношенні СЛУЖБОВЕЦЬ(№_ПАСПОРТА, ПРІЗВИЩЕ, ІМ'Я, ВІК, ОКЛАД) можливими ключами виступають №_ПАСПОРТА і конкатенація атрибутів ПРІЗВИЩЕ, ІМ'Я, ВІК (вважаємо, що №_ПАСПОРТА унікальний для кожної людини).

Серед відношень, які складають базу даних, виділяють так звані об'єктні і зв'язні відношення. Розподіл відношень на ці два класи був запропонований у 1976 р. Ченом в його об'єктно-зв'язній моделі (entity-relationship model).

Об'єктне відношення зберігає дані про всі об'єкти одного й того самого типу, по одному кортежу для кожного екземпляра деякого

об’єкта. Кожний кортеж має ключовий атрибут (атрибути), що ідентифікує відповідний екземпляр, а також інші функціонально залежні атрибути, які описують властивості об’єкта (наприклад, вік, стать, зріст для людини).

Зв’язне відношення зв’язує ключі двох або більшої кількості об’єктних відношень. Воно може містити і інші атрибути, які функціонально залежать від цього зв’язку.

Атрибут відношення, який є первинним ключем деякого іншого відношення, називається стороннім ключем цього відношення.

Наприклад, відношення ЖИВ-У(ПРІЗВИЩЕ, МІСТО, РОКИ) містить первинний ключ відношення ОСОБА(ПРІЗВИЩЕ, ВІК, СТАТЬ, ЗРІСТ) (рис.6в) – ПРІЗВИЩЕ. Атрибут ПРІЗВИЩЕ є для відношення ЖИВ-У стороннім ключем.

Як приклад застосування об’єктно-зв’язної моделі розглянемо базу даних, яка складається з трьох відношень: СТУДЕНТИ(ПРІЗВИЩЕ, ВІК, СТАТЬ) (ключ – ПРІЗВИЩЕ), ДИСЦИПЛІНИ(ПРЕДМЕТ, КІЛЬКІСТЬ_СЕМЕСТРІВ) (ключ – ПРЕДМЕТ), ВИВЧАЄ(СТУДЕНТ, ПРЕДМЕТ) (ключ – СТУДЕНТ, ПРЕДМЕТ).

Перші два відношення є об’єктними, вони містять дані про студентів і предмети відповідно. Третє відношення описує той факт, що певний студент вивчає певну дисципліну, отже, виражає зв’язок між поняттями “студенти” і “предмети”. Відношення ВИВЧАЄ містить ключі відношень СТУДЕНТИ І ДИСЦИПЛІНИ – атрибути ПРІЗВИЩЕ і ПРЕДМЕТ. Можна також внести до бази даних зв’язне відношення ВИВЧАЄ'(СТУДЕНТ, ПРЕДМЕТ, ОЦІНКА), яке містить один залежний атрибут – ОЦІНКА. Оцінка залежить як від студента, так і від предмета, тому цей атрибут віднесений до зв’язного, а не об’єктного відношення.

Розглянемо відношення ВИВЧАЄ"(СТУДЕНТ, ВІК, ПРЕДМЕТ, КІЛЬКІСТЬ_СЕМЕСТРІВ). Це відношення не слід вносити до бази даних, оскільки атрибут ВІК залежить тільки від студента і тому повинен знаходитись в об’єктному відношенні СТУДЕНТИ. Кількість семестрів залежить лише від дисципліни і тому повинна входити у відношення ДИСЦИПЛІНИ.

3.3. ПОСИЛАЛЬНА ЦІЛІСНІСТЬ

Наступним обмеженням реляційної моделі є так звана посилальна цілісність, яка має виконуватись для того, щоб база

даних була корисною і несуперечливою. Суть обмеження така: кожному входженню стороннього ключа в деякий кортеж відношення повинен відповідати кортеж деякого іншого відношення, для якого цей ключ є первинним.

Розглянемо, як забезпечується посилальна цілісність у попередньому прикладі (рис.7).

СТУДЕНТИ

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Шевчук	20	Ж
Макаренко	18	Ч

ДИСЦИПЛІНИ

ПРЕДМЕТ	КІЛЬКІСТЬ СЕМЕСТРІВ
Основи інф.	1
Чисельні мет.	2
Історія Укр.	1

ВИВЧАЄ’’

СТУДЕНТ	ПРЕДМЕТ	ОЦІНКА
Петренко	Основи інф.	5
Петренко	Чисельні мет.	4
Шевчук	Основи інф.	3
Макаренко	Історія Укр.	5

Рис.7. Подані таблицями відношення СТУДЕНТИ, ДИСЦИПЛІНИ, ВИВЧАЄ’’

Тут атрибут СТУДЕНТ і атрибут ПРЕДМЕТ є сторонніми ключами відношення ВИВЧАЄ'. Двом входженням значення “Петренко” у відношення ВИВЧАЄ' відповідає кортеж <Петренко, 20, Ч> відношення СТУДЕНТИ, входженням значень “Шевчук” і “Макаренко” відповідають кортежі <Шевчук, 20, Ж> і <Макаренко, 18, Ч>. Аналогічно для входжень значень атрибуту ПРЕДМЕТ у відношення ВИВЧАЄ' існують відповідні кортежі відношення ДИСЦИПЛІНИ.

При невиконанні посилальної цілісності може статися, що деякий сторонній ключ посилається на об’єкт, про який нічого невідомо.

3.4. НОРМАЛІЗАЦІЯ ВІДНОШЕНЬ

Розглянемо ще одне обмеження, яке накладає реляційна модель. За цим обмеженням відношення повинно знаходитись в так званій нормальній формі. Це означає, що значення атрибутів мають бути атомарними: вони не можуть бути ні списками значень, ні іменами відношень. Наприклад, у відношенні ПРОЖИВАЄ(ПРІЗВИЩЕ, ВІК, СТАТЬ, ЗРІСТ, МІСТО) не можна розширити кортежі так, щоб вони містили список міст, наприклад, <Коваль, 36, Ч, 1.7, <Київ, Чернігів>>. В даному випадку допустиме лише розбиття такої інформаці на два окремі кортежі: <Коваль, 36, Ч, 1.7, Київ>, <Коваль, 36, Ч, 1.7, Чернігів>.

Вибір відношення, де повинні зберігатися дані, які відповідають тому чи іншому атрибуту – досить складна задача, яка носить здебільшого дослідницький, творчий характер. Взагалі не існує єдиного методу для моделювання об'єктів реального світу, тобто для їх представлення у вигляді об'єктів, атрибутів або зв'язків бази даних. Розглянемо деякі основні ідеї, які використовуються на практиці для вирішення цієї проблеми.

Досить часто використовується так звана бінарна реляційна модель, в рамках якої проводиться декомпозиція (розбиття) всіх відношень бази даних на бінарні відношення.

Наприклад, виконаємо декомпозицію відношення ОСОБА(ПРІЗВИЩЕ, ВІК, СТАТЬ, ЗРІСТ) на кілька бінарних відношень. Для цього в кожному бінарному відношенні будемо зберігати ключ (ПРІЗВИЩЕ) і один із залежних атрибутів. Так будуть отримані відношення ВІК'(ПРІЗВИЩЕ, ВІК), ЗРІСТ'(ПРІЗВИЩЕ, ЗРІСТ), СТАТЬ'(ПРІЗВИЩЕ, СТАТЬ) (рис.8).

ВІК'

ПРІЗВИЩЕ	ВІК
Петренко	20
Шевчук	20
Коваль	36

ЗРІСТ'

ПРІЗВИЩЕ	ЗРІСТ
Петренко	1.8
Шевчук	1.6
Коваль	1.7

СТАТЬ'

ПРІЗВИЩЕ	СТАТЬ
Петренко	Ч
Шевчук	Ж
Коваль	Ч

Рис.8 Подані таблицями відношення ВІК', ЗРІСТ', СТАТЬ'

Як можна бачити з прикладу, декомпозиція об'єктних відношень, в яких ключ складається з одного атрибута, проводиться досить легко. Для декомпозиції зв'язних відношень треба застосовувати спеціальні прийоми. Розглянемо як приклад відношення ЖИВ-У(ПРІЗВИЩЕ, МІСТО, РОКИ). Вводити таке бінарне відношення, як ЖИВ'(ПРІЗВИЩЕ, РОКИ) не можна, оскільки втрачається ключ (а він складається з атрибутів ПРІЗВИЩЕ і МІСТО). Для декомпозиції такого відношення необхідно ввести додатковий атрибут для ідентифікації кортежу, так званий сурогатний ключ. Значеннями такого ключа можуть бути числа, що мають смисл в предметній області, або генеруються системою управління базами даних (сучасні СУБД забезпечують таку можливість) і означають дещо, що має смисл тільки для обчислювальної системи. Таким чином і введемо сурогатний ключ для відношення ЖИВ-У. Позначимо його № і проведемо бінарну декомпозицію. Отримані відношення: ПРІЗВИЩЕ', МІСТО' І РОКИ' (рис.9). Цей спосіб є досить штучним, оскільки не зрозуміло, як обирається значення для №. Частіше на практиці використовують n-арні відношення, в яких разом з ключем зібрані всі залежні атрибути.

ПРІЗВИЩЕ'

№	ПРІЗВИЩЕ
137	Петренко
216	Петренко
104	Шевчук
121	Коваль
122	Коваль

МІСТО'

№	МІСТО
137	Київ
216	Донецьк
104	Донецьк
121	Київ
122	Чернігів

РОКИ'

№	РОКИ
137	15
216	5
104	20
121	15
122	5

Рис.9. Подані таблицями відношення ПРІЗВИЩЕ', МІСТО', РОКИ'

В загальному випадку при проведенні декомпозиції відношень користуються такими основними правилами.

Атрибути відношення повинні залежати від ключа (причому від усіх його складових) і тільки від нього. При невиконанні цієї умови відношення підлягає декомпозиції.

Якщо можна провести декомпозицію відношення таким чином, що його за допомогою спеціальних операцій (вони розглядатимуться нижче) можна буде точно відновити, то така декомпозиція має бути виконана.

Найчастіше на практиці для визначення відношень, які моделюють об'єкти і зв'язки між ними, використовують об'єктно-зв'язну модель, а потім розглядають отримані відношення з тим, щоб з'ясувати, чи не можна їх розбити далі у відповідності з ідеями нормалізації. Далі різні способи декомпозиції відношень на більш "дрібні" і композиції відношень у більш "великі" розглядатимуться докладніше.

Розглянемо кілька прикладів основних типів нормалізації.

1.Часткова залежність. Наприклад, у відношенні ВИВЧАЄ(СТУДЕНТ, ВІК, ПРЕДМЕТ, КІЛЬКІСТЬ_СЕМЕСТРІВ) атрибут ВІК залежить не від всього ключа – СТУДЕНТ, ПРЕДМЕТ, а лише від його частини – атрибуту СТУДЕНТ. Тому атрибут ВІК слід виділити в окреме відношення, ключем якого є атрибут СТУДЕНТ.

2.Транзитивна залежність. Розглянемо відношення АВТОМОБІЛЬ(№_МОДЕЛІ, ВИРОБНИК, АДРЕСА, РІК, КОЛІР), ключем якого є атрибут №-МОДЕЛІ. Тут атрибут АДРЕСА насправді залежить від атрибута ВИРОБНИК. Отже, мають існувати окремі відношення, одне з яких є об'єктним і зберігає дані про виробника і його адресу, а інше пов'язує виробника з номером моделі.

3.Багатозначна залежність. Нехай є відношення ПРОДАЄ(ВИРОБНИК, №_МОДЕЛІ, КРАЇНА), кортежі якого містять наступну інформацію: Форд продає модель 257 у Великобританії, Форд продає модель 246 у Великобританії, Форд продає модель 257 в Японії, Форд продає модель 246 в Японії. Якщо кожний виробник продає всі свої моделі в кожній країні, то тут виділяються два незалежні відношення: ПРОДАЄ(ВИРОБНИК, №_МОДЕЛІ) і ЕКСПОРТУЄ(ВИРОБНИК, КРАЇНА), оскільки список моделей, що експортуються, не залежить від країни (і навпаки). Ключ вихідного відношення складається з усіх трьох атрибутів, але воно підлягає відновленню після декомпозиції. Проте якщо припустити, що в майбутньому доведеться зберігати дані про виробника, який експортує тільки деякі моделі в окремі країни, то краще залишити дане відношення в початковому вигляді.

4. Незвичайні випадки. Як, наприклад, виконати декомпозицію відношення НАВЧАЄТЬСЯ(СТУДЕНТ, ПРЕДМЕТ, ВИКЛАДАЧ)? Якщо у

кожного предмета є один певний викладач (тобто має місце зв'язок викладач<–>>предмет), то можна розбити дане відношення на два наступних: ВЧИТЬ(СТУДЕНТ, ПРЕДМЕТ) І ВИКЛАДАЄ(ПРЕДМЕТ, ВИКЛАДАЧ). Якщо кожний викладач викладає один певний предмет, а один і той самий предмет викладається різними викладачами (зв'язок викладач<<–>предмет), то комбінація СТУДЕНТ, ВИКЛАДАЧ є первинним ключем, і дане відношення розіб'ється на ВЧИТЬСЯ(СТУДЕНТ, ВИКЛАДАЧ) І ВИКЛАДАЄ'(ВИКЛАДАЧ, ПРЕДМЕТ). Тепер припустимо, що комбінація СТУДЕНТ, ПРЕДМЕТ також є можливим ключем даного відношення. Це може трапитись, якщо кожний студент вчиться тільки у одного з можливих викладачів кожного з предметів (зв'язки студент<<–>викладач, викладач<<–>>предмет). В цьому випадку відношення ВЧИТЬСЯ(СТУДЕНТ, ВИКЛАДАЧ) і ВИКЛАДАЄ'(ВИКЛАДАЧ, ПРЕДМЕТ) не будуть незалежними, і їх не можна буде поновлювати незалежно одне від одного. Наприклад, не можна додати кортежі, які пов'язують даного студента з різними викладачами: <Іваненко, Смирнов> і <Іваненко, Попович>, якщо ці два викладачі є в іншому відношенні як викладачі одного предмета: <Смирнов, фізика> і <Попович, фізика>. В цьому випадку вихідне відношення краще не змінювати. Слід також враховувати те, що деякі функціональні залежності можуть з часом змінюватись, наприклад, в даний час викладачі можуть викладати тільки один предмет, але в майбутньому ситуація може змінитися, і за відомим викладачем вже не можна буде однозначно визначити предмет. Останній приклад показує, наскільки непростим може виявитись відшукання функціональних залежностей при побудові відношень.

Розділ 4

ОПРАЦЮВАННЯ ВІДНОШЕНЬ

Для отримання інформації з відношень необхідна мова маніпулювання даними, яка визначає відповідні операції над відношеннями. Найбільш важливою частиною мови маніпулювання даними є її розділ, який розглядає правила формулювання запитів до відношень реляційної бази даних. В загальному випадку запити є довільними функціями над відношеннями. Розроблені три абстрактні теоретичні мови маніпулювання даними:

1) реляційна алгебра;

2) реляційне числення кортежів;

3) реляційне числення доменів.

Мови запитів першого типу – алгебраїчні мови – дозволяють виражати запити засобами спеціалізованих операторів, застосовуваних до відношень.

У мовах другого і третього типів запити формулюються з використанням предикатів, які мають задовольняти потрібні кортежі або домени.

Ці мови є еталоном для оцінки реально існуючих мов запитів. Вони були запропоновані Коддом для подання мінімальних можливостей будь-якої мови запитів для реляційної моделі даних. За своєю виразністю всі три мови є еквівалентними між собою. Реальні мови (ISBL, SQL, QBE і ін.) забезпечують не лише функції відповідної теоретичної мови або їх комбінації, але й реалізують деякі додаткові операції – арифметичні операції, команди присвоювання, друку тощо.

4.1. РЕЛЯЦІЙНА АЛГЕБРА

Розглянемо більш докладно операції реляційної алгебри.

1. Об’єднання відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:

$$\mathbf{R = R_1 \cup R_2.}$$

Об’єднання – це операція отримання відношення, яке повністю об’єднує кортежі відношень $\mathbf{R_1}$ і $\mathbf{R_2}$ (рис.10а, б). Для виконання цієї операції відношення $\mathbf{R_1}$ і $\mathbf{R_2}$ повинні мати однакові схеми. Це обмеження має виконуватись також для операцій 2 і 6.

R_1

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Шевчук	20	Ж
Макаренко	18	Ч

R_2

ПРІЗВИЩЕ	ВІК	СТАТЬ
Сидоренко	21	Ж
Романюк	22	Ч
Шевчук	20	Ж

R_3

ЗРІСТ	ВАГА
1.7	70
1.6	50
1.8	75

а)

$R = R_1 \cup R_2$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Шевчук	20	Ж
Макаренко	18	Ч
Сидоренко	21	Ж
Романюк	22	Ч

б)

$R = R_1 - R_2$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Макаренко	18	Ч

в)

$R = R_1 \times R_2$

ПРІЗВИЩЕ	ВІК	СТАТЬ	ЗРІСТ	ВАГА
Петренко	20	Ч	1.7	70
Шевчук	20	Ж	1.7	70
Макаренко	18	Ч	1.7	70
Петренко	20	Ч	1.6	50
Шевчук	20	Ж	1.6	50
Макаренко	18	Ч	1.6	50
Петренко	20	Ч	1.8	75
Шевчук	20	Ж	1.8	75
Макаренко	18	Ч	1.8	75

г)

$\mathbf{R = \pi_{Прізвище,\ Вік}\ (R_1)}$

ПРІЗВИЩЕ	ВІК
Петренко	20
Шевчук	20
Макаренко	18

$\mathbf{R = \pi_{Стать}(R_2)}$

СТАТЬ
Ж
Ч

д)

$\mathbf{R = \sigma_{ПРІЗВИЩЕ = Петренко}(R_1)}$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч

$\mathbf{R = \sigma_{СТАТЬ \neq Ж}(R_1)}$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Макаренко	18	Ч

$\mathbf{R = \sigma_{(СТАТЬ = Ж) \wedge (ВІК \leq 20)}(R_2)}$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Шевчук	20	Ж

е)

$\mathbf{R = R_1 \cap R_2}$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Шевчук	20	Ж

ж)

Рис.10. Операції над відношеннями

2. Різниця відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:

$\mathbf{R = R_1 - R_2}$.

Це операція отримання відношення, яке складається з кортежів, що є кортежами відношення $\mathbf{R_1}$ і не є кортежами відношення $\mathbf{R_2}$ (рис.10 а,в).

3. Декартів добуток відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:

$\mathbf{R = R_1 \times R_2}$.

В цій операції з відношення $\mathbf{R_1}$ арності $\mathbf{m_1}$ і відношення $\mathbf{R_2}$ арності $\mathbf{m_2}$ отримують відношення $\mathbf{R}$ арності $\mathbf{m_1 + m_2}$, причому перші $\mathbf{m_1}$ елементів кортежу результуючого відношення утворюються

кортежем з відношення $\mathbf{R_1}$, а останні $\mathbf{m_2}$ елементів – кортежем з відношення $\mathbf{R_2}$ (рис.10 а,г) .

4. Проекція відношення $\mathbf{R_1}$ на компоненти $\mathbf{i_1, i_2...,i_r}$:

$\mathbf{R = \pi_{i1, i2...,ir}(R_1).}$

Тут $\mathbf{i_1, i_2...,i_r}$ – імена атрибутів відношення $\mathbf{R_1}$. При цій операції з відношення $\mathbf{R_1}$ вибираються вказані атрибути і комбінуються у вказаному порядку. Якщо в результуючому відношенні виникають однакові кортежі, то з них залишають по одному екземпляру (рис.10д).

5. Селекція відношення $\mathbf{R_1}$ за формулою **F**:

$\mathbf{R = \sigma_F (R_1)}$,

де **F** – формула, утворена:

а) операндами, які є іменами атрибутів;

б) логічними операторами $\wedge$ (і), $\vee$ (або), $\neg$ (не);

в) арифметичними операторами порівняння:

$\mathbf{<,\ =,\ >,\ \leq,\ \geq,\ \neq.}$

У формулі можуть використовуватись дужки.

У протилежність проекції, при якій з відношення виділяють потрібні стовпці, в операції селекції відношення досліджують по рядках і виділяють множину рядків. що задовольняють задані умови (рис.10е).

Наступні операції реляційної алгебри можуть бути отримані за допомогою основних, але вони мають самостійне значення.

6. Перетин відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:

$\mathbf{R = R_1 \cap R_2 = R_1 - (R_1 - R_2).}$

В даній операції отримують відношення, яке складається із спільних кортежів відношень $\mathbf{R_1}$ і $\mathbf{R_2}$ (рис.10ж).

7. З'єднання відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:

$$\mathbf{R = R_1 \underset{i\ \theta\ j}{><} R_2 = \sigma_{i\,\theta\,j}(R_1 \times R_2)}$$

Цю операцію називають також θ-з'єднанням .

Тут θ – арифметичний оператор порівняння ($<,\ =,\ >,\ \leq,\ \geq,\ \neq$), **i, j** – імена атрибутів відповідно у відношеннях $\mathbf{R_1}$ і $\mathbf{R_2}$ (рис. 11). Якщо

θ є арифметичним оператором рівності, то операцію називають еквіз'єднанням (рис.12).

R_1

ПРІЗВИЩЕ	ПОСАДА	ОКЛАД
Марчук	інженер	100
Іващенко	бухгалтер	120

R_2

СТАЖ	ПРЕМІЯ
10	100
20	200

$$R = R_1 \underset{\text{ОКЛАД>ПРЕМІЯ}}{><} R_2$$

ПРІЗВИЩЕ	ПОСАДА	ОКЛАД	СТАЖ	ПРЕМІЯ
Іващенко	бухгалтер	120	10	100

Рис.11.Операції над відношеннями

R_1

ПРІЗВИЩЕ	НАСЕЛЕНИЙ ПУНКТ	РОКИ
Іваненко	Київ	10
Петренко	Чернігів	15
Сидоренко	Київ	20
Тарасенко	Суми	10

R_2

МІСТО	НАСЕЛЕННЯ
Чернігів	0.4
Суми	0.3

$$R = R_1 \underset{\text{Н\_ПУНКТ=МІСТО}}{><} R_2$$

ПРІЗВИЩЕ	Н_ПУНКТ	РОКИ	МІСТО	НАСЕЛЕННЯ
Петренко	Чернігів	15	Чернігів	0.4
Тарасенко	Суми	10	Суми	0.3

Рис.12. Операції над відношеннями

Можливість порівняння атрибутів **i** і **j** при з'єднанні відношень має бути передбачена при проектуванні бази даних: повинна існувати узгодженість у іменах атрибутів, їх смисловому значенні і використовуваних доменах. Існують ситуації, коли з'єднання неможливе навіть при однакових атрибутах, наприклад, відношення СЛУЖБОВЕЦЬ(ІМ'Я, ВІК) і МОРСЬКІ_СУДИ (ІМ'Я, СТРОК_СЛУЖБИ) не підлягають з'єднанню, а відношення СЛУЖБОВЕЦЬ(ІМ'Я, ВІК) і

ПОХОДЖЕННЯ(ЛЮДИНА, МІСЦЕ_НАРОДЖЕННЯ) підлягають з'єднанню при умові ІМ'Я=ЛЮДИНА.

Отже, при виконанні операції з'єднання виконуються такі дії:

1) отримується декартів добуток відношень,

2) з отриманого відношення вибираються кортежі, в яких значення вказаних атрибутів задовольняють поставлену умову.

8. Природне з'єднання відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:

Нехай відношення $\mathbf{R_1}$ і $\mathbf{R_2}$ з точністю до порядку атрибутів мають відповідно схеми:

$\mathbf{R_1 (A_1 , A_2,..., A_k, B_1, B_2, ..., B_n)}$;

$\mathbf{R_2 (A_1, A_2, ..., A_k, C_1, C_2, ..., C_m)}$,

де імена $\mathbf{A_1 , A_2,..., A_k}$ у обох відношень збігаються (точніше мають однакові домени), а решта розрізняються.

Тоді

$$\mathbf{R = R_1 >< R_2 = \pi_{B1, ..., Bn, A1,..., Ak, C1, ..., Cm}(\sigma_{R1.A1=R2.A2, ..., R1.Ak=R2.Ak}(R_1 \times R_2))}$$

де $\mathbf{R_1.A_1}$ – ім'я стовпця відношення $\mathbf{R_1 \times R_2}$, який відповідає стовпцю $\mathbf{A_1}$ у відношенні $\mathbf{R_1}$, $\mathbf{R_2.A_1}$ – ім'я стовпця відношення $\mathbf{R_1 \times R_2}$, який відповідає стовпцю $\mathbf{A_1}$ у відношенні $\mathbf{R_2}$ (рис.13).

Отже, при виконанні операції природного з'єднання

виконується декартове множення заданих відношень,

з отриманого відношення вибираються кортежі, в яких збігаються значення спільних атрибутів,

в результуючому відношенні залишають по одному екземпляру спільних атрибутів.

$\mathbf{R_1}$

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Шевчук	20	Ж
Макаренко	18	Ч

$\mathbf{R_2}$

ПРІЗВИЩЕ	ВІК	ЗРІСТ	ВАГА
Петренко	20	1.8	70
Шевчук	20	1.6	50
Макаренко	21	1.7	70

$\mathbf{R = R_1 \succ\!\prec R_2}$

ПРІЗВИЩЕ	ВІК	СТАТЬ	ЗРІСТ	ВАГА
Петренко	20	Ч	1.8	70
Шевчук	20	Ж	1.6	50

Рис.13. Операції над відношеннями

При виконанні з'єднання двох відношень можуть трапитись особливі випадки:

відношення $\mathbf{R_1}$ і $\mathbf{R_2}$: не мають спільних атрибутів. В цьому випадку $\mathbf{R_1 \succ\!\prec R_2 = R_1 \times R_2}$, тобто з'єднанням відношень є їх декартів добуток;

всі атрибути відношень є спільними, тобто відношення мають однакові схеми. В цьому випадку $\mathbf{R_1 \succ\!\prec R_2 = R_1 \cap R_2}$, тобто з'єднання відношень визначається як їх перетин.

Пропозиції Кодда по використанню реляційної алгебри були реалізовані в проекті Астрід, який використовувався в різних організаціях на протязі ряду років. Основні операції реляційної алгебри в термінах мови Астрід записуються так:

1. Об'єднання відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:
   R_1 union R_2 (або $\mathbf{R_1 + R_2}$).
2. Різниця відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:
   R_1 without R_2 (або $\mathbf{R_1 - R_2}$).
3. Декартів добуток відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:
   R_1 produced_with R_2 (або $\mathbf{R_1 ** R_2}$).
4. Проекція відношення **R** на компоненти $\mathbf{i_1, i_2,...,i_r}$:
   R projected_to i_1, i_2,...,i_r .
5. Селекція відношення **R** за формулою **F**:
   R selected_on [F].
6. Перетин відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:
   R_1 intersect_with R_2 (або $\mathbf{R_1 \cdot R_2}$).
7. Природне з'єднання відношень $\mathbf{R_1}$ і $\mathbf{R_2}$:
   R_1 joined_to R_2 (або $\mathbf{R_1 * R_2}$).

Основною перевагою реляційної алгебри є те, що вона є замкненою по відношенню до реляційних операцій. Це означає, що жодна з алгебраїчних операцій не може створити об'єкт, який виходить за рамки алгебри. На відміну від цього в мовах, що базуються на реляційному численні, для формулювання деяких складних запитів доводиться користуватися засобами, які не

надаються численням. Складні запити, сформульовані в термінах реляційної алгебри, легше читати і розуміти, ніж створені за допомогою реляційного числення. Тим не менше реляційне числення є потужним засобом для формулювання запитів, і реальні мови, які базуються на ньому, широко використовуються на практиці.

4.2. РЕЛЯЦІЙНЕ ЧИСЛЕННЯ КОРТЕЖІВ (МОВА SQL)

Однією з розповсюджених мов, що базуються на реляційному численні зі змінними-кортежами, є мова SQL (Structured Query Language – структурована мова запитів), розроблена спеціалістами фірми IBM.

Мова SQL використовує поняття змінної-кортежу, значенням якої може бути будь-який кортеж деякого відношення. Значення атрибутів вибраного кортежу можна отримати, використовуючи позначення для вибору компонент із запису, аналогічно мові Паскаль. Тобто доступ до компонент кортежу здійснюється за допомогою складених імен.

Наприклад, якщо змінна S може мати своїми значеннями кортежі відношення ЖИВ-У (рис.6б), то атрибути кортежу задаються як S.Прізвище, S.Місто, S.Роки. Синтаксис мови SQL дозволяє у більшості випадків не використовувати префікс у вигляді змінної-кортежу (S).

В загальному випадку запити на SQL мають вигляд:

SELECT [UNIQUE] <список змінних>
FROM <список відношень>
WHERE <предикат>

Тут **select** (вибрати), **unique** (єдиний), **from** (із), **where** (де) – ключові слова; <список змінних> – один або кілька імен атрибутів із кортежів відношень у <списку відношень>. Потрібні кортежі вибираються у відповідності з <предикатом>. Необов'язковий параметр **unique** запобігає видачі у результуючому відношенні однакових кортежів.

Приклад 1. Нехай, наприклад, треба сформулювати запит до відношення ЖИВ-У (рис.6б), щоб з'ясувати прізвища людей, які жили в Донецьку більше 10 років. Цей запит на SQL буде записаний так:

SELECT прізвище
FROM ЖИВ-У
WHERE місто='Донецьк' **AND** роки > 10

Результатом запиту буде відношення (рис.14):

ПРІЗВИЩЕ
Шевчук

Рис.14. Результат запиту до відношення ЖИВ-У

В термінах реляційної алгебри цей запит запишеться так:

$\pi_{\text{ПРІЗВИЩЕ}}(\sigma_{(\text{МІСТО='Донецьк'})\wedge(\text{РОКИ}>10)}(\text{ЖИВ-У}))$

Приклад 2. Розглянемо запит з двома змінними-кортежами. Можна вибирати кортежі з одного відношення в залежності від значень з іншого відношення. Нехай треба отримати інформацію про прізвища, вік і стать людей, які живуть в Києві не менше 15 років (рис.6б,в). Відповідний запит на SQL виглядатиме так:

SELECT UNIQUE прізвище, вік, стать
FROM ОСОБА, ЖИВ-У
WHERE місто='Київ' **AND** роки >= 15

Тут в останньому рядку треба вказувати імена змінних-кортежів, оскільки ідентифікатор "прізвище" неоднозначний, – він зустрічається в обох відношеннях, – на відміну від ідентифікаторів "місто" і "роки". Фактично тут відбувається з'єднання двох відношень за спільним атрибутом "прізвище".

Відповіддю на даний запит буде відношення (рис.15):

ПРІЗВИЩЕ	ВІК	СТАТЬ
Петренко	20	Ч
Коваль	36	Ч

Рис.15. Результат запиту до відношень ОСОБА, ЖИВ-У

У термінах реляційної алгебри запит запишеться так:

$\pi_{\text{ПРІЗВИЩЕ}}(\sigma_{(\text{МІСТО='Київ'})\wedge(\text{РОКИ}>=15)}(\text{ОСОБА} >< \text{ЖИВ-У}))$

Останній запит можна записати інакше – з використанням вкладеного підзапиту, який має такий самий синтаксис, що і основний запит, і генерує неіменоване проміжне відношення:

SELECT UNIQUE прізвище, вік, стать
FROM ОСОБА
WHERE прізвище **IN** (**SELECT** прізвище
FROM ЖИВ-У
WHERE місто='Київ' **AND** роки=15)

(Якщо відношення мають більше одного спільного атрибута, список цих атрибутів береться у подвійні кутові дужки << >>).

Частина запиту після другого **SELECT** означає відношення, яке містить потрібні прізвища. Зовнішнє **WHERE** вимагає видати кортежі зі значеннями прізвищ, які зустрічаються в проміжному

відношенні. На це вказує оператор **IN**, який звичайно використовується для з'ясування належності елемента множині.

В термінах реляційної алгебри можна отримати той самий результат, надавши ім'я результату підзапиту:

$$R' = \pi_{ПРІЗВИЩЕ}(\sigma_{(МІСТО='Київ')\wedge(РОКИ>=15)}(ЖИВ\text{-}У))$$

$$R = П_{ПРІЗВИЩЕ,СТАТЬ,ЗРІСТ}(R' >< ОСОБА)$$

Наведений спосіб не застосовний, якщо в результуючому відношенні треба отримати атрибути з обох відношень. Наприклад, якщо у попередньому прикладі потрібні відомості про прізвище, вік і роки, то треба використати перший метод:

SELECT UNIQUE прізвище, вік, роки
FROM ОСОБА, ЖИВ-У
WHERE місто='Київ' **AND** роки>=15
AND ОСОБА.прізвище = ЖИВ-У.прізвище

Крім оператора **IN** можна також використовувати **NOT IN** (не є елементом множини). Наприклад, з розглядуваних відношень ЖИВ-У і ОСОБА треба отримати інформацію про вік людей, які не живуть в Києві з використанням оператора **NOT IN** запит виглядатиме так:

SELECT UNIQUE вік
FROM ОСОБА
WHERE Київ **NOT IN**
(**SELECT** місто
FROM ЖИВ-У
WHERE прізвище = ОСОБА.прізвище)

Цей самий запит можна сформулювати так:

SELECT UNIQUE вік
FROM ОСОБА, ЖИВ-У
WHERE місто <> 'Київ'
AND ОСОБА.прізвище = ЖИВ-У.прізвище

Результатом запиту буде відношення (рис.16):

ВІК
20

Рис.16. Результат запиту до відношень ОСОБА, ЖИВ-У

В мові SQL існує можливість працювати з множинами і застосовувати теоретико-множинні операції і функції. Це робить мову SQL дещо схожою на реляційну алгебру.

Операції, які можна застосовувати для множин, можуть застосовуватись і до підзапитів у мові SQL, а саме:

S_1 UNION S_2 – об'єднання множин;
X IN S – елемент множини;
X NOT IN S – доповнення множини;
EXISTS S – S – непорожня множина;
NOT EXISTS S – S – порожня множина;
COUNT (S) – кількість елементів.

Операції **IN, NOT IN, EXISTS, NOT EXISTS** є предикатами, і їх можна об'єднувати з іншими, використовуючи логічні зв'язки **AND, OR** і **NOT** після **WHERE**.

Наступні операції застосовні лише в тому випадку, коли S – множина цілих чисел. **SUM(S)** – сума всіх елементів S, **MAX(S), MIN(S)** – максимальний і мінімальний елементи з S, **AVG(S)** – середнє арифметичне елементів з S, **AVG(S) = SUM(S)/COUNT(S)**. Операції **COUNT, SUM, MAX**, **MIN, AVG** є вбудованими функціями. Вони застосовні до множин, і їх результатом є число, яке можна порівнювати з іншими числами або використовувати у виразах, наприклад: N=COUNT(S), 2*AVG(S)>0.6.

Розглянемо приклади застосування теоретико-множинних операцій об'єднання і різниці в SQL. Нехай з відношень ЖИВ-У і ОСОБА треба отримати прізвища всіх чоловіків, а також всіх, хто проживав у Донецьку. Цей запит можна записати так:

(**SELECT UNIQUE** прізвище
FROM ЖИВ-У
WHERE місто='Донецьк')
UNION
(**SELECT UNIQUE** прізвище
FROM ОСОБА
WHERE стать = 'Ч')

Результатом запиту буде відношення (рис.17):

ПРІЗВИЩЕ
Петренко
Шевчук
Коваль

Рис.17. Результат запиту до відношень ОСОБА, ЖИВ-У

Деякі версії SQL дозволяють використовувати операцію **MINUS** для отримання різниці двох множин. Розглянемо, наприклад, запит для отримання прізвищ всіх жінок з відношення ОСОБА.

(**SELECT UNIQUE** прізвище
FROM ОСОБА)

MINUS
(**SELECT UNIQUE** прізвище
FROM ОСОБА
WHERE стать = ‘Ч’)

4.3. РЕЛЯЦІЙНЕ ЧИСЛЕННЯ ДОМЕНІВ (МОВА QBE)

Широко розповсюдженою мовою, яка базується на реляційному численні зі змінними-доменами, є мова QBE (Query By Example – запит за зразком), яка була розроблена приблизно одночасно з SQL. Мова QBE суттєво відрізняється від SQL тим, що областю значень змінних є домен відповідного атрибута (наприклад, множина всіх прізвищ або міст), а не всі кортежі деякого відношення. Тому QBE називається численням, орієнтованим на домени.

Для задання запитів мовою QBE користувачеві надається екран, на якому він може створити “шаблон” (тобто форму таблиці), який містить в собі кортежі потрібного відношення. В цих кортежах користувач записує так звані зразки елементів, тим самим вказуючи, які кортежі треба вибрати. Після цього система вибирає потрібні кортежі з відповідних відношень. Наприклад, запит про прізвище і вік людей, які жили в Києві 15 років, матиме вигляд (рис.18):

ЖИВ-У

ПРІЗВИЩЕ	МІСТО	РОКИ
A.	Київ	15

ОСОБА

ПРІЗВИЩЕ	ВІК	СТАТЬ	ЗРІСТ
P.A.	**P.**		

Рис.18. Запит до відношень ОСОБА, ЖИВ-У, сформульований у термінах QBE

Тут нам потрібно отримати кортежі з відношення ОСОБА, для яких значення прізвища (A .) збігається зі значенням прізвища з кортежів відношення ЖИВ-У (тут A використовується також для зв’язку таблиць), для яких, в свою чергу, місто=”Київ”, а роки=15. Символ **P**. (print) в атрибутах ПРІЗВИЩЕ і ВІК показує, що це значення має бути виведене у результуючому відношенні. Використання підкреслювання для символу А показує, що це зразок (як приклад) елемента такого виду. Як зразок може бути використаний будь-який символ або послідовність символів. Часто як зразок використовують

значення з домену атрибута. Це уявне значення буде замінене правильним, коли буде потрібно вивести результат. Отже, уявні значення відрізняються від реальних констант, наприклад, “Київ” і “15”, які не підкреслюються. Колонки таблиць, що відповідають атрибутам, які не цікавлять користувача в даному запиті, лишаються порожніми. Замість підкреслювання зразка можна писати символ підкреслювання перед першим його символом, наприклад, _x, _y.

Зазначимо, що всі змінні в QBE неявно зв’язані квантором існування і зображаються підкресленими символьними рядками. Не підкреслені рядки є константами.

В мові SQL для запобігання видачі дублікатів кортежів результуючого відношення використовується ключове слово UNIQUE. В мові QBE дублікати вилучаються за замовчуванням. Якщо ж потрібно виводити дублікати, в шаблоні запиту у відповідному атрибуті вказують ключове слово **ALL** (всі), наприклад, **P.ALL**.

В мові QBE допускаються порівняння не лише за рівністю. Нехай, наприклад, треба дізнатися, які особи мають зріст, більший, ніж Коваль. Відповідний запит виглядатиме так (рис.19):

ОСОБА

ПРІЗВИЩЕ	**ВІК**	**СТАТЬ**	**ЗРІСТ**
Коваль	N		
P._x	> N		

Рис.19.Запит до відношення ОСОБА

Існує простий спосіб переведення запитів, записаних мовою SQL, на мову QBE. Нехай в термінах SQL запит має вигляд:

SELECT F_1,F_2,...

FROM R_1,R_2,...

WHERE (F_1 θ F_2) AND (F_3 θ F_4) AND ... де θ – оператор порівняння.

Щоб сформулювати цей запит в термінах QBE, треба виконати такі дії:

в шаблонах для відношень R_1,R_2, ... під кожним вибраним полем F_1,F_2, ... записуємо ‘**P**.’;

під тими полями, які зустрічаються у **WHERE** праворуч від оператора порівняння θ, записуємо зразки елементів;

під кожним полем, яке зустрічається ліворуч від оператора порівняння, записуємо вираз для порівняння, який використовує цей зразок елемента.

Наприклад, F1>F2 запишеться так: розмістимо _x під F2, а вираз >_x під F1. Якщо відбувається порівняння на рівність, то знак рівності опускається.

Якщо до одного поля відносяться кілька умов, QBE надає можливість використовувати блок умов. Це колонка, в яку користувач може записати будь-яку умову, яка допустима в SQL після WHERE. Наприклад, запит до відношення ЖИВ-У: перелічити прізвища людей, які проживали в Києві або Чернігові терміном від 6 до 12 років, виглядатиме так (рис.20):

ЖИВ-У

ПРІЗВИЩЕ	МІСТО	РОКИ	УМОВА
P.	_x	_y	_x = ″Київ″ OR _x = ″Чернігів″
			_y >= 6 AND _y <= 12

Рис.20. Запит до відношення ЖИВ-У

Розглянуті досі запити на QBE генерували результати з кортежів одного відношення. Кортежі з інших відношень використовувались лише в умовах відбору. Часто ж треба отримати інформацію з кількох відношень і, крім того, обчислити деякі значення, які безпосередньо в базі даних не зберігаються – так звані обчислювальні поля. Щоб це зробити, потрібен окремий шаблон для результуючого відношення. Для створення такого шаблону користувачеві надаються відповідні можливості. Значеннями в цьому шаблоні є вирази, записані з використанням змінних, які були визначені в раніше створених шаблонах. Користувач може створювати свої імена атрибутів для колонок, вони можуть не співпадати з існуючими іменами атрибутів. Наприклад, потрібно отримати інформацію про прізвища і стать людей, які жили в Києві, і кількість років, які ці особи не жили в Києві. Відповідний запит можна подати у вигляді (рис.21):

ЖИВ-У

ПРІЗВИЩЕ	МІСТО	РОКИ
_x		_a

ОСОБА

ПРІЗВИЩЕ	ВІК	СТАТЬ	ЗРІСТ
_x	_b	_y	

РЕЗУЛЬТАТ

ПРІЗВИЩЕ	МІСТО	СТАТЬ	НЕ ЖИВ
Р. _x	**Р**.Київ	**Р**.	**Р**._b – _a

Рис.21. Запит до відношень ОСОБА, ЖИВ-У і його результат

Тут значення НЕ-ЖИВ обчислюється по значенням полів ВІК і РОКИ у відношеннях ОСОБА і ЖИВ-У. Кортежі мають відноситись до одної і тієї самої особи. Це вказується введенням змінної-зразка _x у спільному атрибуті двох таблиць – ПРІЗВИЩЕ.

Мови SQL і QBE забезпечують великі можливості формулювання запитів високого рівня до реляційних баз даних. Вони спираються на реляційне числення, і метод формулювання запитів значною мірою незалежний від реалізації бази даних.

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Що називають автоматизованою інформаційною системою? Для чого призначені АІС, яка їх класифікація?
2. Який зміст понять “інформація” і “дані”?
3. Що називається предметною областю?
4. В чому полягає інфологічне проектування АІС?
5. Як здійснюється датологічне проектування АІС?
6. Що називається банком даних? Які сфери застосування банків даних?
7. Які основні складові мають банки даних?
8. Які функції виконують мова опису даних, мова маніпулювання даними?
9. Що таке словник даних, яке його призначення?
10. Який склад адміністрації бази даних, які функції вона виконує?
11. Як відбувається захист даних у банку даних?
12. Як здійснюється моделювання предметної області?
13. Що таке модель даних? які моделі даних використовуються в банках даних, чим вони характерні?
14. Що називається моделлю бази даних?
15. Як здійснюється проектування бази даних?
16. Які основні характеристики моделі “об’єкт-атрибут-зв’язок”?
17. Які види зв’язків між об’єктами розглядаються в рамках моделі “об’єкт-атрибут-зв’язок”?.
18. Що називають відношенням реляційної моделі даних? Який зміст понять “домен”, “кортеж”, “атрибут”?.

19. Як відбувається подання відношень таблицями? Що називається схемою відношення?
20. Що означає поняття “ключ відношення”? Яки види ключів розглядаються в рамках реляційної моделі даних?
21. Що називають об’єктними і зв’язними відношеннями?
22. В чому полягає принцип посилальної цілісності?
23. Для чого і як використовується бінарна реляційна модель?
24. Які основні правила декомпозиції використовуються при опрацюванні відношень?
25. Які основні типи нормалізації відношень використовуються в рамках реляційної моделі?
26. Для чого призначена реляційна алгебра? Які операції використовуються в реляційній алгебрі?
27. Які основні характеристики мови SQL? Як формулюються запити мовою SQL?
28. Які основні характеристики мови QBE? Як формулюються запити мовою QBE?
29. Як перевести запит, записаний мовою SQL, на QBE?

Розділ 5

ОСНОВИ РОБОТИ З СУЧАСНИМИ СИСТЕМАМИ УПРАВЛІННЯ БАЗАМИ ДАНИХ

5.1. ОСНОВНІ ПОНЯТТЯ

Перед розглядом конкретних прикладів застосування сучасних СУБД слід повторити основні теоретичні положення, викладені вище.

Як уже зазначалося, базою даних називають сукупність взаємопов'язаних даних деякої предметної області, які зберігаються в пам'яті ЕОМ і організовані таким чином, що можуть використовуватись для розв'язування багатьох задач багатьма користувачами.

Для автоматичного опрацювання даних їх треба певним чином формалізувати. Формалізація даних відбувається на основі певної моделі даних. До основних моделей даних відносяться: ієрархічна, мережева і реляційна.

В ієрархічній моделі даних інформація про об′єкти предметної області подається у вигляді дерева. Мережева модель даних допускає більш складні зв′язки між об′єктами, ніж ієрархічна.

Найбільш поширеною є реляційна модель даних (РМД), запропонована у 1970 р. англійським математиком Коддом. В цій моделі база даних являє собою набір взаємопов′язаних двовимірних таблиць або відношень (від англ. relation – відношення). Таблиця складається з рядків, які в термінології РМД називаються записами (або кортежів). Кожен запис є сукупністю фіксованої кількості полів (атрибутів). Таблиця має заголовок (назву відношення), стовпці, або поля таблиці, також мають заголовки. Дані в кожному полі повинні бути одного типу (текстові, числові тощо). Можна вважати, що кожен запис таблиці зберігає дані про окремий об'єкт предметної області, а поля запису описують певні властивості кожного з об'єктів.

Основне обмеження РМД: будь-яка таблиця не повинна містити однакових записів. Ця вимога очевидна – кожен об'єкт предметної області є унікальним.

Для виконання цього обмеження в РМД вводиться поняття ключа.

Ключ – це поле таблиці або сукупність полів, значення в яких не повторюються і тим самим забезпечують унікальність кожного запису таблиці.

Ключ, що складається з одного поля, називається простим, з кількох полів – складеним.

Приклад. Нехай предметна область – деяка фірма. Створимо базу даних, яка зберігає дані про її співробітників і зроблені їм виплати заробітної плати протягом деякого часу. Об'єктами предметної області є співробітники фірми, серед їх властивостей, що мають бути відображені у базі даних, визначимо такі: ідентифікаційний код; прізвище, ім'я, по-батькові; дата народження; домашня адреса; посада у фірмі; стаж роботи; дата отримання заробітної плати; отримана сума заробітної плати.

Вказані властивості слід віднести до двох різних таблиць, одна з яких зберігає дані, що змінюються рідко і зберігаються тривалий час (анкетні дані – код; ПІБ; дата народження; адреса; посада; стаж), інша – дані, що часто поновлюються і несуть, фактично, допоміжну інформацію (дані про виплати – дата виплати; сума, а також поле, яке буде далі вибране для ідентифікації співробітника). Назвемо ці таблиці відповідно СПІВРОБІТНИКИ і ВИПЛАТИ. Визначимо ключі цих таблиць. Очевидно, що ключем (простим ключем) таблиці СПІВРОБІТНИКИ може бути поле КОД, як таке, що дійсно однозначно ідентифікує кожного громадянина. Зрозуміло також, що поле КОД повинно бути ключовим і для таблиці ВИПЛАТИ, щоб логічно зв'язати таблиці, які містять різні дані про одних і тих самих людей. Але КОД для таблиці ВИПЛАТИ простим ключем бути не може: ця таблиця має зберігати дані про виплати, зроблені протягом деякого періоду (наприклад, року), тому для будь-якої людини в таблицю може бути занесено більше одної виплати. Таким чином виникне повторення даних у полі КОД, що суперечитиме означенню ключа. Отже, визначити ключ таблиці ВИПЛАТИ слід на основі певних домовленостей щодо предметної області. Якщо домовитись, що кожному співробітнику в один день робиться не більше однієї виплати, комбінація значень у полях КОД і ДАТА ВИПЛАТИ однозначно ідентифікуватиме кожний запис таблиці ВИПЛАТИ. Таким чином, таблиця ВИПЛАТИ має складений ключ, що складається з двох полів: КОД і ДАТА ВИПЛАТИ. Слід зауважити, що дані в кожному окремо взятому з цих полів можуть повторюватись, але комбінації даних є неповторюваними (див. рис.22)

Співробітники

Код*	ПІБ	Дата народження	Адреса	Посада	Стаж
111	Іваненко І.І.	12.05.55	вул.Широка, 33/15	директор	22
222	Петренко П.П.	30.01.70	просп.Миру, 29/147	інженер	10
333	Сидоренко С.С.	5.09.60	вул.Зелена, 88	бухгалтер	17
...	...	...	...	...	...

Виплати

Код*	Дата виплати*	Сума
111	18.01.01	400,00
111	3.02.01	500,00
222	18.01.01	200,00
222	3.02.01	300,00
333	18.01.01	300,00
333	3.02.01	400,00
...	...	...

Рис.22. Приблизний вміст таблиць СПІВРОБІТНИКИ і ВИПЛАТИ (зірочкою позначені ключові поля)

Крім забезпечення унікальності записів таблиць ключі призначені для встановлення зв'язків між таблицями, що дозволяє розглядати окремі таблиці як складові однієї цілісної бази даних про певну предметну область. Так, таблиці СПІВРОБІТНИКИ і ВИПЛАТИ пов′язані за ключовим полем КОД.

Нагадаємо основні типи зв'язків між таблицями:

один-до-одного (1:1). При цьому зв'язку одному запису першої таблиці відповідає не більше одного запису другої таблиці. Для встановлення такого зв'язку ключі двох таблиць повинні бути рівними;

один-до-багатьох (1:∞). При цьому зв'язку одному запису першої таблиці може відповідати довільна кількість записів другої таблиці. Для встановлення такого зв'язку ключ другої таблиці повинен містити в собі, як підмножину, ключ першої;

багато-до-одного (∞:1). Це зв'язок один-до-багатьох, розглянутий з протилежної точки зору: перша таблиця стає другою і навпаки;

багато-до-багатьох (∞:∞).При цьому зв'язку кожному запису першої таблиці може відповідати довільна кількість записів другої

таблиці і навпаки. Для встановлення такого зв'язку ключі повинні бути складеними, перетинатися і не міститися один в одному.

Наприклад, між таблицями СПІВРОБІТНИКИ і ВИПЛАТИ можна встановити зв'язок один-до-багатьох (відповідно, між таблицями ВИПЛАТИ і СПІВРОБІТНИКИ – багато-до-одного).

Якщо відношення СПІВРОБІТНИКИ розбити на два: СПІВРОБІТНИКИ(КОД*, ПІБ, ПОСАДА, СТАЖ) і АНКЕТИ(КОД*, ДАТА НАРОДЖЕННЯ, АДРЕСА), між утвореними таблицями можна встановити зв'язок один-до-одного.

Комплекси програм, призначені для створення, опрацювання і підтримки баз даних, називаються системами управління базами даних (СУБД). В залежності від підтримуваної моделі даних СУБД бувають ієрархічні, мережеві і реляційні. Найбільшого розповсюдження у прикладних галузях набули реляційні СУБД. Цей клас програм далі будемо розглядати на прикладі СУБД Paradox і Access.

5.2. СИСТЕМА УПРАВЛІННЯ БАЗАМИ ДАНИХ PARADOX

5.2.1. ОСНОВНІ ПРИНЦИПИ РОБОТИ З СИСТЕМОЮ PARADOX

`Paradox` – реляційна СУБД, у якій база даних являє собою сукупність таблиць, зв'язки між якими користувач створює самостійно на етапі опрацювання даних. Кожна таблиця зберігається у файлі з розширенням `.db`. З цим файлом може бути пов'язана група файлів з тим самим ім'ям, що і таблиця, які містять певну додаткову інформацію щодо цієї таблиці (наприклад – `*.val` – файл контролю вхідних даних, `*.f*` – опис звіту тощо). При вилученні таблиці вилучається також вся сім'я. Така організація даних створює певні незручності, зокрема при перенесенні бази даних між комп′ютерами.

При роботі з системою `Paradox` створюються допоміжні тимчасові таблиці, які знаходяться лише в оперативній пам'яті. Вони мають такі імена: `Answer` – таблиця, що містить результати запитів, `Deleted` – таблиця, яка зберігає вилучені записи, `Struct` – таблиця, яка зберігає структуру певного відношення тощо. При потребі зберегти такі таблиці на диску слід перейменовувати їх

засобами Paradox для запобігання втрати їх змісту при повторному створенні однойменної тимчасової таблиці.

Після запуску системи (файл запуску – paradox.exe) екран розподіляється на три частини (рис. 23):

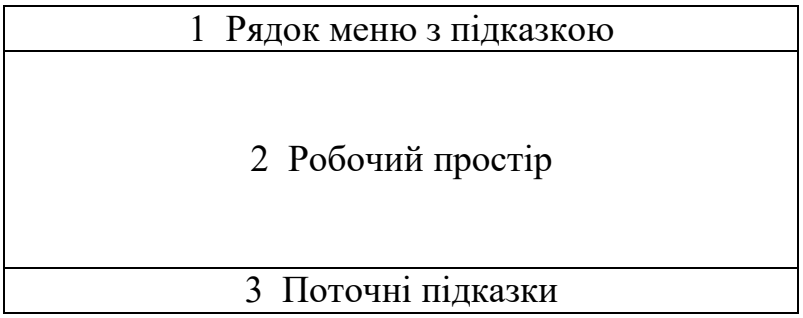


Рис. 23. Структура екрану при роботі з СУБД Paradox

Головне меню системи (яке з′являється на екрані після завантаження), містить такі пункти: **View** (перегляд таблиць), **Ask** (створення запитів), **Report** (створення звітів), **Create** (створення структури таблиць), **Modify** (зміна структури і вмісту таблиць), **Image** (управління візуальним поданням даних), **Forms** (створення форм), **Tools** (допоміжні засоби), **Scripts** (сценарії), **Help** (допомога), **Exit** (вихід) Меню має ієрархічну структуру. Під кожним з пунктів меню виводяться короткі відомості про його призначення.

Paradox може працювати в різних режимах (створення структури таблиці, перегляд таблиці, редагування даних тощо). Індикатор режиму виводиться в правому верхньому куті екрана. Кожен з режимів має власне меню, поява якого на екрані і активізація одного з пунктів здійснюється натисненням функціональної клавіші F10. Закінчивши роботу в певному режимі, треба вибрати в меню цього режиму команду DoIt! або натиснути клавішу F2 для виходу з режиму із збереженням результатів роботи. Для виходу без збереження результату слід вибрати з меню команду Cancel. Для виходу в надменю треба натиснути клавішу Esc.

В робочому просторі системи Paradox можуть знаходитись так звані образи (графічні зображення), що подають певні об’єкти бази даних: таблиці, бланки запитів, форми, звіти. Якщо в робочому просторі знаходиться декілька образів, один з них є активним, тобто підлягає опрацюванню в даний момент. Активний образ виділяється кольором, в ньому розташовується курсор.

Рядок поточної підказки містить інформацію про комбінації клавіш, що можуть використовуватися в даній ситуації.

Функціональні клавіші при роботі системи Paradox мають таке призначення:

F1 – допомога (Help) – отримання довідкової інформації про роботу системи; повторне натиснення F1 призводить до появи індексного вказівника;

F2 – завершення поточної дії зі збереженням результату;

F3 – переміщення між образами в робочому просторі вгору;

F4 – переміщення між образами в робочому просторі вниз;

F5 – введення зразка при формулюванні запитів;

F6 – вибір поля для виведення в результаті запиту;

F7 – перемикач подання образу на екрані: у вигляді таблиці або у вигляді форми;

F8 – вилучення активного образу (в якому знаходиться курсор) з робочого простору; натиснення ALT-F8 приводить до очищення робочого простору;

F9 – перехід в режим редагування активної таблиці;

F10 – вихід в поточне меню.

Зауваження. Перед роботою з Paradox слід встановити потрібний робочий каталог. Для цього використовується команда головного меню: Tools-More-Directory, яка потребує введення шляху до нового робочого каталогу. Після цього для підтвердження зміни каталогу слід вибрати Ok, для відміни – Cancel.

5.2.2. СТВОРЕННЯ ТАБЛИЦЬ БАЗИ ДАНИХ

Таблиця – це основний об'єкт бази даних, призначений для зберігання даних про предметну область.

Створення таблиць у Paradox відбувається в два етапи.

1. Створення структури таблиці. На цьому етапі вводяться назви полів таблиці, визначаються їх типи, вказуються ключові поля. Отже, системі управління базами даних повідомляється схема відношення.

2. Внесення даних у таблицю і їх редагування.

У Paradox допускаються такі основні типи полів:

An – алфавітно-цифрове (довжиною від 1 до 255 символів), n вказує кількість символів у полі;

N – числове (цілі і дійсні числа, до 17 значущих цифр у числі);

$ – грошове (після числа дописуються 2 розряди після коми);

S – коротке ціле;

D – поле дати (у форматі мм/дд/рр або дд.мм.рр.).

Щоб позначити поле як ключове, після позначення його типу слід поставити зірочку.

Порядок створення структури таблиці у Paradox розглянемо на прикладі таблиць СПІВРОБІТНИКИ і ВИПЛАТИ. Також введемо до розгляду спеціальну допоміжну таблицю ШТАТНИЙ РОЗКЛАД, що складається з єдиного поля – ПОСАДА, і містить перелік можливих посад фірми. Ця таблиця буде використовуватись для автоматизації введення даних у таблицю СПІВРОБІТНИКИ.

Для створення структури таблиць призначена послуга головного меню Create. Після вибору послуги система запитує ім'я нової таблиці (Table:). Слід ввести його і натиснути Enter. Назвемо створювану таблицю Spivrob:

```
Table: Spivrob↵
```

Зауваження. Запит на введення імені таблиці для опрацювання видається системою при роботі у будь-якому режимі. Створення структури таблиці – єдиний випадок, коли це ім'я (нове) необхідно вводити з клавіатури. В решті випадків, коли опрацьовується таблиця, записана на диск, її ім'я можна ввести двома способами:

ввести ім'я таблиці з клавіатури;

натиснути Enter, і з переліку імен таблиць робочого каталогу, як з меню, вибрати потрібне ім'я. Для прискорення введення можна ввести першу літеру імені таблиці, і з'явиться перелік імен, що починаються на вказану літеру.

Створення структури таблиці у Paradox полягає у заповненні спеціальної таблиці Struct. Така допоміжна таблиця є у кожної таблиці бази даних і зберігає її назви, порядок і типи полів, а також визначення ключових полів. Таблиця Struct може бути в будь-який момент змінена за допомогою послуги головного меню Modify-Restructure, що вимагає вказування імені таблиці, структура якої підлягає редагуванню.

У таблиці Struct слід заповнити стовпці Field Name (ім'я поля) і Field Type (тип поля). При цьому на екран виводиться підказка щодо доступних типів полів системи Paradox.

Структура таблиці Spivrob має такий вигляд (Рис. 24):

Struct	Field Name	Field Type
1	Код	N*
2	ПІБ	A30
3	Дата народження	D
4	Адреса	A40
5	Посада	A15
6	Стаж	S

Рис. 24. Структура таблиці Spivrob

Заповнивши таблицю Struct, слід натиснути F2 для її збереження і вилучення з робочого простору. Після цього в робочому каталозі вже існує порожня таблиця Spivrob, що має визначену структуру.

Аналогічно створимо структури таблиць ШТАТНИЙ РОЗКЛАД (за правилами MS-DOS назвемо її Shtat) і ВИПЛАТИ (Vypl) Ці структури зображені на рис.25:

Struct	Field Name	Field Type
1	Посада	A15

а) Структура таблиці Shtat

Struct	Field Name	Field Type
1	Код	N*
2	Дата виплати	D
3	Сума	$

б) Структура таблиці Vypl

Рис.25. Структура таблиць ШТАТНИЙ РОЗКЛАД і ВИПЛАТИ

Заповнення таблиць даними розпочнемо з допоміжної таблиці ШТАТНИЙ РОЗКЛАД. Для введення даних у таблиці в системі Paradox призначена послуга головного меню Modify-Edit, яка потребує введення імені таблиці. Одним з розглянутих вище способів введемо ім'я Shtat. Таблиця з'являється в робочому просторі у звичайному вигляді – рядки і стовпці, ширина яких відповідає встановленому типу даних. При введенні даних у поля таблиці використовуються такі клавіші і їх комбінації:

←, → – перехід між полями;

↑, ↓ – перехід між записами;

Delete – вилучення поточного запису;

Ctrl-BS – вилучення даних з поточного поля;

BS – вилучення останнього символу у поточному полі. Так можна частково редагувати поля таблиці;

Alt-F5 – переведення поточного поля в режим редагування. В цьому режимі можна вільно пересуватись по полю за допомогою клавіш управління курсором і виконувати необхідні зміни у полі. Для виходу з режиму редагування слід натиснути Enter;

Ctrl-D – введення в поле значення з відповідного поля попереднього запису;

Ctrl-U – відміна попередньої дії.

Після закінчення введення даних слід натиснути F2 для їх запису на диск.

Таблицю можна вивести на екран для перегляду (без можливості внесення змін). Для цього використовується послуга головного меню View, яка потребує введення імені таблиці. Таким способом можна розмістити в робочому просторі декілька таблиць. Для переходу між ними використовуються клавіші F3 – перехід на одну таблицю вгору, і F4 – перехід на одну таблицю вниз. Таблицю, викликану для перегляду, можна перевести в режим редагування натисненням клавіші F9. Після внесення необхідних змін для їх збереження і повернення в режим перегляду таблиці треба натиснути F2.

СУБД Paradox дозволяє здійснювати контроль значень, що вводяться в таблиці бази даних. Для цього в режимі редагування таблиці, до внесення в неї даних, на поля таблиці встановлюються певні умови. Якщо при внесенні даних у таблицю ці умови порушуються, Paradox видає попереджувальне повідомлення, і дані не вводяться доти, поки користувач не врахує вказані у повідомленні умови. Такий контроль дозволяє підтримувати цілісність і несуперечливість даних у базах даних.

Для встановлення обмежень слід викликати таблицю на екран в режимі редагування (Modify-Edit), увійти в меню цього режиму, натиснувши F10, і вибрати послугу ValCheck (value checking – перевірка значень).

Існують такі типи обмежень (у термінології Paradox):

LowValue – найменше допустиме значення для поля;

HighValue – найбільше допустиме значення для поля;

Required – поле обов’язково повинно бути заповнене;

Default – встановлення значення за замовчуванням. При вказуванні цього обмеження вводиться значення, яке при заповненні таблиці з'являтиметься у відповідному полі при натисненні Enter;

TableLookUp – перевірка значення, що вводиться у поле, на його наявність у іншій таблиці (довіднику).

Застосування можливостей системи Paradox щодо контролю даних розглянемо на прикладі таблиці СПІВРОБІТНИКИ. Нехай при заповненні таблиці повинні виконуватись такі умови:

поле КОД повинно обов'язково бути заповнене;

в поле ДАТА НАРОДЖЕННЯ повинні заноситись дати не більш ранні, ніж 1 січня 1937 р.;

в полі АДРЕСА при натисненні Enter повинно з'являтись "м. Київ, вул.";

значення в поле ПОСАДА повинні вводитись з таблиці ШТАТНИЙ РОЗКЛАД;

значення у полі СТАЖ повинні знаходитись в межах від 1 до 55 років, за замовчуванням – 1 рік.

Відкриємо таблицю в режимі редагування:

Modify-Edit; Table: Spivrob↵

У меню режиму редагування виберемо послугу Valcheck:

F10, Valcheck-Define↵

Зауваження. Підпослуга Define призначена для встановлення обмеження; для зняття обмеження з поля використовується підпослуга Clear.

Після цього треба помістити курсор в потрібне поле і натиснути Enter. На екрані з'явиться меню із можливих типів обмежень. Слід вибрати потрібне, і для всіх типів обмежень, крім Required, ввести додаткові параметри. Про те, що обмеження записане, свідчить повідомлення в правому нижньому куті екрану.

Коротко опишемо дії, які повинні виконуватись для встановлення на поля таблиці вказаних обмежень.

Поле КОД:

F10, Valcheck-Define, помістити курсор у поле КОД, ↵, Required ↵

Поле ДАТА НАРОДЖЕННЯ:

F10, Valcheck-Define, помістити курсор у поле ДАТА НАРОДЖЕННЯ, LowValue, у відповідь на запит Value: ввести 01.01.37 ↵

Поле АДРЕСА:

F10, Valcheck-Define, помістити курсор у поле АДРЕСА, ↵ , Default ↵,

у відповідь на запит Value: ввести м.Київ, вул. ↵

Поле ПОСАДА:

F10, Valcheck-Define, помістити курсор у поле ПОСАДА,↵ , TableLookUp, у відповідь на запит Table: ввести ім'я таблиці-довідника Shtat. Меню, що з'явиться, містить дві послуги: JustCurrentField (перевіряти тільки поточне поле) і AllCorrespondingFields (перевіряти всі відповідні поля). В даному випадку слід вибрати першу послугу. Наступне меню містить послуги PrivateLookUp (схована перевірка: якщо введене значення відсутнє в таблиці-довіднику, виникає помилка, але побачити довідник не можна) і HelpAndFill (заповнення поля з допомогою). Слід вибрати другу послугу. В цьому випадку перед заповненням поля ПОСАДА треба натиснути клавішу F1. На екрані з'явиться таблиця-довідник, в якій треба поставити курсор на те значення, яке має бути введене у таблицю СПІВРОБІТНИКИ, і натиснути F2. Таблиця-довідник з екрану зникне, а в основній таблиці у полі ПОСАДА з'явиться потрібне значення.

Поле СТАЖ:

F10, Valcheck-Define, помістити курсор у поле СТАЖ, LowValue, у відповідь на запит Value: ввести 1 ↵

F10, Valcheck-Define, помістити курсор у поле СТАЖ, HighValue, у відповідь на запит Value: ввести 55 ↵

F10, Valcheck-Define, помістити курсор у поле СТАЖ, Default, у відповідь на запит Value: ввести 1 ↵

При створенні таблиці ВИПЛАТИ треба врахувати, що вона зв'язана з таблицею СПІВРОБІТНИКИ за ключовим полем КОД. Отже, всі значення поля КОД таблиці ВИПЛАТИ повинні знаходитись у відповідному полі таблиці ВИПЛАТИ. Оскільки Paradox контроль посилальної цілісності бази даних не забезпечує, про це повинен потурбуватися розробник бази даних. Отже, доцільно при введенні даних у поле КОД таблиці ВИПЛАТИ використовувати таблицю СПІВРОБІТНИКИ (використати тип контролю TableLookUp).

5.2.3. СОРТУВАННЯ ДАНИХ У ТАБЛИЦЯХ

СУБД Paradox дозволяє виконувати сортування даних у полях таблиць, тобто впорядковувати дані за зростанням або спаданням коду першого символу у текстовому полі або значення у числовому полі. Для сортування даних використовується послуга головного меню Modify-Sort, яка потребує введення імені таблиці. Якщо вибрана таблиця не містить ключових полів, пропонується результат сортування (таблицю, в якій змінено порядок записів) зберегти під ім'ям вихідної таблиці (Same) або під новим ім'ям (New). Якщо ж таблиця містить ключові поля, результат сортування можна зберегти тільки під новим ім'ям. Підготовка до сортування полягає у заповненні спеціальної форми, яка являє собою перелік імен полів таблиці. Зліва від полів можуть бути вказані цифри 1, 2, і т.д. до кількості полів у таблиці. Після цифри може бути вказана літера D, наприклад, 1D, 2D тощо. Цифра означає чергу поля при сортуванні (1- сортувати за вказаним полем, 2 – за цим полем, сортуються дані, для яких значення у першому полі рівні і т.д.). За замовчуванням сортування відбувається за зростанням, при вказуванні літери D сортування відбувається у зворотному порядку.

Наприклад, якщо для таблиці СПІВРОБІТНИКИ форма для сортування складена як на рис.5, дані будуть виведені у порядку спадання значень у полі СТАЖ, а для тих співробітників, що мають однаковий стаж, їх прізвища будуть виведені в алфавітному порядку.

	Код
2	ПІБ
	Дата народження
	Адреса
	Посада
1D	Стаж

Рис.26. Форма для сортування таблиці Співробітники

Для виконання сортування і виведення на екран таблиці-результату слід натиснути F2.

5.2.4. ЗАПИТИ У СИСТЕМІ PARADOX

Запит – це об'єкт бази даних, призначений для опрацювання даних, а саме: вибірки даних із таблиць за певними критеріями, виконання обчислень над даними з таблиць, додавання, вилучення і зміни даних у таблицях.

Paradox підтримує мову запитів QBE (Query By Example). Отже, для завдання запиту заповнюється спеціальний бланк, в якому вказуються необхідні критерії опрацювання даних.

Для створення запитів у системі Paradox призначена послуга головного меню Ask, яка потребує введення імені таблиці. Після виконання послуги на екрані з'являється бланк запиту, структура якого відповідає структурі таблиці. Слід заповнити бланк, і для виконання запиту натиснути F2.

Результатом запиту може бути:

а) тимчасова таблиця Answer, яка містить дані, відповідні умовам, вказаним у бланку. Ця таблиця зберігається в оперативній пам'яті до виконання наступного запиту і появи нової таблиці Answer. Якщо треба зберегти результат запиту на диску, таблицю Answer необхідно перейменувати за допомогою послуги головного меню Tools-Rename-Table, вказати ім'я таблиці, що перейменовується (Answer), і ім'я нової таблиці.

б) у випадку виконання запиту на модифікацію даних у таблицях (додавання, вилучення записів, зміну даних у полях таблиць) таблиця Answer не створюється, а змінюється вміст таблиці, до якої робився запит, і створюється спеціальна тимчасова таблиця, яка може бути використана для повернення бази даних у початковий стан.

Запити на вибірку даних

Це найбільш розповсюджений і простий тип запитів до таблиць бази даних. В результаті запитів на вибірку утворюється таблиця Answer, що містить підмножину записів і полів вихідної таблиці. Для виведення поля в таблиці Answer у бланку запиту в цьому полі натисненням клавіші F6 ставиться спеціальна позначка (√ – Check) . Для вибору потрібних записів після Check записуються умови, яким повинні задовольняти дані вихідної таблиці.

При завданні умов можна використовувати такі засоби:
метасимволи для створення шаблонів даних:

.. – будь-яка кількість будь-яких символів у полі;

@ – будь-який один символ у полі;

=, >, <, <=, >= – оператори діапазону для числових, грошових полів і полів типу дата (знак "=" не пишеться);

like – оператор пошуку даних, що співпадають з указаними на 2/3 (початок тексту повинен бути вказаний точно);
not, and, or – логічні операції для утворення складених умов. Замість and переважно пишуть кому, складові частини умови or записують в різних рядках бланку запиту (при цьому позначки Check) повинні стояти в усіх заповнених рядках бланку запиту;
blank – логічна функція, що має значення "істина" в незаповненому полі;
today – функція, що повертає поточну дату.
Створення запитів розглянемо на прикладі таблиці СПІВРОБІТНИКИ.

Запит 1. Сформувати список прізвищ співробітників з указуванням їх посад.

Для виклику на екран бланку запиту треба вибрати з головного меню послугу Ask і вказати ім'я таблиці, до якої робитиметься запит: Spivrob. Потім заповнити бланк, як вказано на рис.27:

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
		√			√	

Рис.27. Бланк запиту 1

Для виконання запиту і появи на екрані таблиці Answer слід натиснути клавішу F2. Таблиця Answer включає в себе поля ПІБ і ПОСАДА з усіх записів таблиці СПІВРОБІТНИКИ.

Якщо в результаті запиту не виводяться ключові поля, може виникнути ситуація, коли таблиця Answer містить однакові записи. За замовчуванням Paradox виводить на екран тільки різні записи, ігноруючи повтори. Так, якщо в таблиці СПІВРОБІТНИКИ є два співробітника із значеннями у полях ПІБ і ПОСАДА відповідно "Іваненко І.І." і "інженер", в результаті запиту 1 виведеться один рядок з такими даними (рис. 28):

Answer	ПІБ	Посада
	...	...
	Іваненко І.І.	інженер
	...	...

Рис. 28. Результат виконання запиту 1

Якщо необхідно виводити на екран всі, навіть однакові, записи, які задовольняють умову запиту, замість позначки Check у бланку запиту ставлять позначку √+ (CheckPlus) натисненням комбінації клавіш Alt-F6. Отже, якщо бланк запиту 1 буде заповнений так, як

на рис.29а), таблиця Answer матиме вигляд, як на рис 29б). У наступних запитах без втрати загальності будемо використовувати позначку Check.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
		√+			√+	

а)

Answer	ПІБ	Посада
	...	...
	Іваненко І.І.	інженер
	Іваненко І.І.	інженер
	...	...

б)

Рис.29 Бланк запиту 1 для виведення однакових записів і результат його виконання

Запит 2. Сформувати список даних про співробітників (ПІБ, Дата народження, Посада, Стаж), стаж роботи яких: а) не менше п'яти років; б) від 10 до 15 років; в) один рік або більше 20 років. Вміст бланків відповідних запитів зображено на рис.30.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
		√	√		√	√ **>=5**

а)

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
		√	√		√	√ **>=10,<=15**

б)

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
		√	√		√	√ **1**
		√	√		√	√ **>20**

в)

Рис.30. Бланки запитів 2а), 2б), 2в)

Запит 3. Вивести всі дані про директора організації. Вміст бланку запиту зображено на рис.31.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
	√	√	√	√	√ **"директор"**	√

Рис.31. Бланк запиту 3

Зауваження 1. При вказуванні умови на точний збіг текстових даних (як у попередньому прикладі), дані, записані кирилицею, беруться у лапки, латиницею – записуються без лапок.

Зауваження 2. Щоб поставити позначки `Check` *у всіх полях бланку запиту, слід натиснути F6 в найлівішому полі бланку, під назвою таблиці.*

Запит 4. Сформувати список даних про співробітників (ПІБ, ДАТА НАРОДЖЕННЯ, АДРЕСА), що проживають на проспекті Миру. Вміст бланку запиту зображено на рис.32.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
		√	√	√ **.. Просп. Миру..**		

Рис.32.Бланк запиту 4

Запит 5. В організації працюють співробітники на прізвища Сидоров, Сидорко, Сидорук. Вивести всі дані про цих співробітників. Вміст бланку запиту зображено на рис. 33.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
	√	√ **like “Сидоров”**	√	√	√	√

Рис.33.Бланк запиту 5

Запит 6. Відомо, що в таблиці Співробітники поле АДРЕСА заповнене не для всіх співробітників. Вивести всі відомості про співробітників, для яких це поле заповнене. Вміст бланку запиту зображено на рис. 34.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
	√	√	√	√ **not blank**	√	√

Рис.34. Бланк запиту 6

Запити до зв’язаних таблиць

Як вказувалося вище, при створенні багатотабличної бази даних у СУБД `Paradox` користувач повинен сам піклуватися про цілісність даних, вносячи у відповідні однотипні поля, за якими зв’язуються таблиці, однакові дані. Певна перевірка цілісності даних відбувається при створенні запитів, у яких беруть участь кілька таблиць. Якщо треба отримати у таблиці `Answer` дані з двох і більше таблиць, ці таблиці послідовно вказуються у послузі `Ask`. В результаті на екрані з’являється відповідна кількість бланків запитів. Для пов’язування таблиць у спільних ключових полях всіх бланків встановлюються так звані зразки (вони повинні бути однакові). Для встановлення зразку треба натиснути клавішу F5, а потім ввести певний алфавітно-цифровий рядок, що починається з літери і не містить пропусків. Зразок на бланку виділяється кольором. Потім всі бланки

заповнюються звичайним способом залежно від умови задачі. Для виконання запиту натискається F2.

Запит 7. Вивести відомості про всі виплати всім співробітникам. В результаті повинні міститися поля ПІБ, ДАТА ВИПЛАТИ, СУМА.

Оскільки таблиця `Answer` повинна містити поля з двох таблиць, треба послідовно застосувати до вказаних таблиць послугу `Ask` і заповнити відповідні бланки запитів так, як вказано на рис.35а). Таблиця `Answer` для цього запиту вказана на рис.35б)

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
	x	√				

а)

Vypl	Код	Дата виплати	Сума
	x	√	√

Answer	ПІБ	Дата виплати	Сума
	…	…	…

б)

Рис.35 Бланк запиту 7 і результат його виконання

Зауваження. Порядок полів у таблиці Answer залежить від порядку виклику таблиць послугою Ask.

Обчислення у запитах

Розглянуті запити здійснюють вибірку даних із таблиць, не змінюючи дані ні у вихідних таблицях, ні у таблиці `Answer`. Проте існує можливість за допомогою запитів виконувати над даними певні обчислення, відображаючи результати у таблиці `Answer` і не змінюючи вихідних таблиць.

Для виконання обчислення над даними у відповідному полі бланку запиту використовується спеціальний оператор **`calc`** (обчислення) у наступному форматі:

`calc <вираз> [as "ім'я поля" ]`

Результатом застосування цього оператора є нове поле в таблиці `Answer`. Ім'я цього поля вказується в операторі після службового слова as (як). Якщо слово as не вказується, ім'я новому полю дається автоматично за певними правилами. У виразі оператора `calc` використовуються константи, посилання на поля таблиць за допомогою зразків, знаки арифметичних операцій (+, -, *, /) та спеціальні оператори: **`Sum`** (сума значень у відповідному полі), **`Count`**

(кількість записів у полі), **Average** (середнє арифметичне), **Min** (мінімальне значення), **Max** (максимальне значення у полі).

Запит 8. Вивести дані про розміри всіх виплат співробітнику Петренку П.П. за останні 60 днів з указанням відрахувань у пенсійний фонд (1%). Вміст бланку запиту зображено на рис.36а). Таблиця `Answer` вказана на рис.36б).

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
	x	**√ "Петренко П.П."**				

Vypl	Код	Дата виплати	Сума
	x	**√ >today-60**	**√ z, calc z*0.01 as "Пенс. фонд"**

а)

Answer	ПІБ	Дата виплати	Сума	Пенс. фонд
	Петренко П.П.	…	100	1
	Петренко П.П.	…	200	2
		…		

б)

Рис.36. Бланк запиту 8 і результат його виконання

Як бачимо, зразки використовуються не тільки для зв'язку таблиць, а і для організації обчислень над даними у полях таблиці.

Запит 9. Обрахувати загальний фонд заробітної плати та розмір середньої виплати за 2001 рік. Вміст бланку запиту зображено на рис.37. Таблиця `Answer` міститиме один запис з двох полів, назви яких вказані у бланку запиту.

Vypl	Код	Дата виплати	Сума
		..01	calc sum as "Всього", calc average as "Середня"

Рис.37. Бланк запиту 9

Оператори, що використовуються у виразах `calc`, можуть застосовуватись у запитах для вибірки інформації. Продемонструємо це на наступному прикладі.

Запит 10. Вивести прізвища співробітників, які за 2001 рік отримали більше 5000 гривень. Вміст бланку запиту зображено на рис. 38.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
	x	√				

Vypl	Код	Дата виплати	Сума
	x	√ >=1.1.01, <=31.12.01	√ sum>5000

Рис.38. Бланк запиту 10

Зауваження. На основі розглянутих прикладів можна зробити висновок, що таблиця Answer містить таку кількість полів: кількість позначок Check + кількість операторів calc у бланку запиту.

Запити на модифікацію таблиць

Як вказувалось вище, мова QBE дає можливість не тільки вибирати дані з таблиць, а і змінювати вміст таблиць бази даних. Існують три типи запитів, які дозволяють змінювати таблиці:

запити на зміну даних у таблицях;

запити на додавання даних до таблиць;

запити на вилучення даних з таблиць.

Перевагою системи Paradox є можливість повернути таблицю, змінену одним з таких запитів, у вихідний стан.

Запити на вилучення даних з таблиць

Такі запити дозволяють вилучати з таблиці записи, у яких значення певних полів відповідають вказаним умовам. Умови записуються за такими ж правилами, як у запитах на вибірку даних. Якщо в такому запиті не вказати жодної умови, із таблиці будуть вилучені всі записи. Для вилучення даних з таблиць використовується оператор **delete**, який записується в самому лівому стовпці бланку запиту, під назвою таблиці. Позначки Check у бланках запиту не ставляться.

Запит 11. Вилучити з таблиці СПІВРОБІТНИКИ всі відомості про інженерів, стаж роботи яких більше 20 років. Вміст бланку запиту зображено на рис.39.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
Delete					“інженер”	>20

Рис.39. Бланк запиту 11

В результаті виконання запиту на вилучення даних:

з вихідної таблиці вилучаються вказані записи;

створюється тимчасова таблиця Deleted, яка має таку ж структуру, як і вихідна таблиця, і містить записи, що були вилучені в результаті запиту. Таблиця Deleted існує до виконання наступного

запиту на вилучення даних або до виходу із СУБД. Отже, якщо необхідно зберегти вилучені записи на диску, таблицю Deleted слід перейменувати.

Для повернення вилучених даних до таблиці можна скористатись послугою об′єднання двох таблиць однакової структури. Для цього слід вибрати в головному меню СУБД Paradox команду Tools-More-Add, і вказати послідовно імена двох таблиць: спочатку тієї, вміст якої додається (Deleted або нове ім'я цієї таблиці), а потім тієї, до якої додаються записи (вихідної таблиці).

Запити на зміну даних у таблицях

Такі запити дозволяють змінювати значення окремих полів вихідної таблиці. Для полів, що підлягають зміні, можуть бути вказані певні умови. Позначки Check у бланках запиту не ставляться.

Для модифікації даних у таблицях використовується оператор **changeto**, який записується в тому полі бланку запиту, де необхідно змінити дані.

Запит 12. В таблицю ВИПЛАТИ була помилково введена дата виплати 1 травня 2001 року. Потрібно змінити її скрізь у таблиці на 11 травня 2001 року. Вміст бланку запиту зображено на рис. 40.

Vypl	Код	Дата виплати	Сума
		1.05.01, changeto 11.05.01	

Рис.40. Бланк запиту 12

В результаті виконання запиту на зміну даних:

модифікується таблиця, до якої зроблено запит;

створюється тимчасова таблиця Changed, яка має таку структуру, як і вихідна таблиця, і містить оригінали записів, що зазнали змін при запиті.

Отже, після виконання запиту 12 таблиця ВИПЛАТИ містить замість значень дати 01.05.01 значення 11.05.01, а ті записи, які містили дату 1.05.01, в незміненому вигляді винесені в таблицю Changed. Оскільки ця таблиця тимчасова, вона існує до виконання наступного запиту на зміну даних або до виходу із СУБД. Отже, якщо необхідно зберегти початкові значення даних на диску, таблицю Changed слід перейменувати.

Запит 13. Збільшити стаж всіх співробітників на один рік. На відміну від попереднього запиту потрібно змінити не окремі записи таблиці, а всі. В подібних випадках перед оператором changeto не вказується умова відбору даних. Вміст бланку запиту зображено на

рис.41. Звернемо увагу на те, що тут використано зразок для посилання на поле таблиці.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
						s, changeto s+1

Рис.41. Бланк запиту 13

Для повернення зміненої таблиці у початковий стан можна:

– якщо це можливо, створити запит на зміну даних, що має зворотну дію. Наприклад, бланк запиту, що скасовує результати запиту 13, зображено на рис.42.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
						s, changeto s-1

Рис.42.

– за допомогою запиту на вилучення даних вилучити з таблиці записи, що зазнали змін, а потім до отриманої таблиці під′єднати таблицю Changed.

Запити на додавання даних до таблиць

Такі запити дозволяють додавати до вихідної таблиці записи з однієї або кількох інших таблиць. Всі таблиці, що беруть участь у запиті, повинні бути викликані на екран командою Ask. Відповідні поля всіх таблиць повинні бути позначені однаковими зразками. Бажано, щоб вказані таблиці мали однакову або схожу структуру. Отже, подібні запити застосовуються переважно до багатотабличних баз даних із заздалегідь продуманою і втіленою структурою таблиць. Для додавання даних використовується оператор **insert**, який записується в самому лівому стовпці бланку запиту тієї таблиці, до якої додаються дані. Умови, яким повинні відповідати дані з інших таблиць для додавання у вихідну, записуються у відповідних полях бланків запитів цих таблиць. Якщо не вказувати умов, у таблицю додадуться всі записи з указаних таблиць. Позначки Check у бланках запиту не ставляться.

Запит 14. Нехай у базі даних є додаткова таблиця АНКЕТА, в якій зберігаються відомості про бажаючих вступити на роботу до даної організації. Таблиця містить такі поля: КОД, ПІБ, ДАТА НАРОДЖЕННЯ, ХАРАКТЕРИСТИКА, ЗАРАХОВАНИЙ. В останнє поле занесені результати співбесіди: “+”(позитивний результат) і “–“(негативний результат). Треба додати до таблиці СПІВРОБІТНИКИ дані про тих осіб з таблиці АНКЕТА, для яких у полі ЗАРАХОВАНИЙ вказано “+”.Вміст бланку запиту зображено на рис. 43.

Spivrob	Код	ПІБ	Дата народження	Адреса	Посада	Стаж
insert	**K**	**P**	**D**			

Anketa	Код	ПІБ	Дата народження	Характеристика	Зарахований
	K	**P**	**P**		**“+”**

Рис.43. Бланк запиту 14

В результаті виконання запиту на додавання даних:

до вихідної таблиці додаються вказані записи, причому заповненими в них є лише ті поля, в яких у бланку запиту були поставлені зразки;

створюється тимчасова таблиця `Inserted`, яка має таку структуру, як і вихідна таблиця, і містить записи, що були додані у вихідну таблицю.

5.2.5. ФОРМИ У СИСТЕМІ PARADOX

Форма – це об’єкт бази даних, призначений для зручного і наочного перегляду, введення і редагування даних в таблицях.

Будь-яка таблиця може подаватись на екрані у двох виглядах: табличному або формному.

Основні відмінності в графічному образі на екрані при перегляді таблиці і форми полягають у наступному:

образ таблиці може одночасно подавати кількість записів, обмежену розмірами екрану монітора, форма – або один, або кількість записів, визначену користувачем;

образ таблиці може одночасно подавати кількість полів, обмежену розмірами екрану монітора, форма може містити більше полів;

образ таблиці може перевищувати ширину екрана монітора, форма не може бути ширшою за екран;

дані в образі таблиці завжди організовані у вигляді рядків і стовпців, дані у формі можна розмістити різноманітними способами.

Форми можуть бути зв’язаними або багатотабличними, тобто одночасно подавати дані з кількох таблиць, пов’язаних за допомогою ключових полів, причому кожна із зв’язаних форм може розглядатися як окремий образ на екрані.

Для покращення зовнішнього вигляду форми користувач може оформляти її різними кольорами і рамками.

В СУБД `Paradox` існує поняття стандартної форми. Під час перегляду або редагування таблиці можна в будь-який момент

натиснути клавішу F7 (перемикач форм), щоб перейти від перегляду таблиці до перегляду форми і навпаки. При цьому за замовчуванням викликається стандартна форма. Крім того, для кожної таблиці користувач може створити до 14 власних форм і організувати виклик будь-якої з них при натисненні F7.

Для створення і редагування форм у системі Paradox призначена послуга головного меню **Forms**, містить дві підпослуги:

Design (розробити): розробити форму користувача для таблиці;
Change (змінити): змінити або заново створити форму.

Design. За допомогою цієї послуги можна розробити форму користувача для таблиці системи Paradox, зручно розташовуючи інформацію в формі. Можна, наприклад:

вибрати поля, які будуть відображуватись на екрані;
включити обчислювальні поля;
“розгорнути” значення в полі на два (або більше) рядки;
вставити пояснення і повідомлення;
виділити інформацію за допомогою різних засобів;
виділити поля в прямокутники, створити межі форм;
розмістити форму на кількох сторінках;
відобразити у формі кілька записів в табличному вигляді;
вбудувати у форму, що розробляється, форми користувача для інших таблиць.

Щоб створити форму користувача, потрібно вибрати послугу Forms-Design і виконати наступні дії:

Вказати ім’я таблиці, для якої буде створюватись форма.

Вибрати номер форми з меню, яке містить літеру F і номери від 1 до 14. При переміщенні по цьому меню під кожним пунктом виводиться пояснювальний текст (до 40 символів). Літера F позначає стандартну форму, відповідним пояснювальним текстом для неї є Standard form. Для створення нової форми користувача слід вибрати такий номер форми, пояснювальним текстом для якого є Unused form (невикористана форма) і натиснути Enter. Після цього на запит “Form description:” слід ввести пояснювальний текст для нової форми. Цей текст буде в подальшому з’являтись під відповідним номером форми замість Unused form.

Після вказування загальних характеристик форми Paradox переходить в режим створення форми. В цьому режимі є внутрішнє меню, яке активізується натисненням клавіші F10. На екрані монітора

з’являється робочий простір, в якому можна розміщувати зафарбовані прямокутники (команда внутрішнього меню Style-Color-Area), кольорові рамки (Style-Color-Border), тексти, які вказують назви таблиць і полів, пояснювальні тексти, а також шаблони полів (далі – поля) таблиці (Field-Place), які при використанні форми будуть заповнюватись конкретними даними з таблиці. Для переміщення по робочому простору можна користуватися клавішами управління курсором, пояснювальні тексти набираються безпосередньо в позицію курсору, як в текстовому редакторі.

Поля, що розміщуються на формі, можуть бути взяті безпосередньо з таблиці (Regular), отримані в результаті обчислень над даними з таблиці (Calculated), або такі, що допускають лише перегляд без можливості редагування (DisplayOnly). При потребі вилучити з робочого простору певні елементи застосовуються команди внутрішнього меню: для прямокутників і рамок відповідно Area-Erase, Border-Erase, для полів – Field-Erase, пояснювальні тексти редагуються загальноприйнятими способами.

Після створення форми для її збереження слід натиснути клавішу ‘F2’ або вибрати послугу Do_It! з меню Forms. Для перегляду таблиці у вигляді створеної форми слід викликати таблицю на екран командою View, і вибрати з головного меню команду Image -Pickform, вказавши потім номер потрібної форми. Для повернення в режим таблиці можна натиснути F7. Повторне натиснення F7 викличе на екран стандартну форму. Щоб зробити можливим перемикання клавішею F7 між таблицею і формою користувача, слід скористатись послугою Image-KeepSet, коли потрібна форма знаходиться на екрані.

При розробці багатотабличних форм необхідно виконати ці дії кілька разів, потім вбудувати одні форми в інші.

Створення форм розглянемо на прикладі бази даних, що складається з таблиць СПІВРОБІТНИКИ і ВИПЛАТИ. Необхідно створити форму, що містить дані з обох таблиць, причому для кожного співробітника разом з його анкетними даними виводиться декілька записів про зроблені йому виплати.

При створенні зв’язаних або багатотабличних форм спочатку створюється форма для підлеглої таблиці, потім – для головної, і в кінці відбувається включення підлеглої форми у головну.

Для розв’язання поставленої задачі слід виконати наступні дії:

1.1. Створити екранну форму для допоміжної таблиці Vypl. Для цього потрібно:

а) увійти до головного меню СУБД Paradox; вибрати послугу Forms-Design; у відповідь на запит ввести або вибрати зі списку ім'я Vypl; з наведеного списку номерів і імен форм для даної таблиці обрати 'Unused form'; у відповідь на запит 'Form description:' ввести ім'я створюваної форми, наприклад, Form2;

б) в робочому просторі, що з'явився на екрані, встановити курсор в позицію <1,1> (це бажано при створенні форми для допоміжної таблиці); увійти до меню, натиснувши F10; обрати послугу Style-Color-Area; натисненням Enter зафіксувати верхній лівий кут області форми (позиція <1,1>), клавішами управління курсором встановити правий нижній кут області форми і зафіксувати його натисненням Enter. За допомогою клавіш управління курсором з таблиці кольорів вибрати кольори для області форми і тексту (горизонтальними стрілками встановлюється колір тексту, вертикальними – колір фону) і натиснути Enter;

в) розмістити в області форми поля з таблиці Vypl, що мають виводитися на екран при перегляді таблиці (ДАТА ВИПЛАТИ, СУМА). Зауважимо, що поле КОД, за яким будуть пов'язуватись таблиці, у підлеглій формі виводитись не повинно. Для розміщення поля в формі необхідно: увійти в меню (F10), обрати послугу Field-Place-Regular; горизонтальними стрілками вибрати назву потрібного поля і натиснути Enter; в області форми встановити курсор в позицію, починаючи з якої дане поле буде розташоване в формі, натиснути Enter, горизонтальними стрілками встановити ширину поля і підтвердити вибір натисненням Enter; встановити курсор над отриманою пунктирною лінією і написати ім'я поля.

1.2.Зробити створену форму багатозаписною

Для цього потрібно увійти в меню (F10); обрати послугу Multi-Records-Define; обрати і відмітити натисненням Enter ліву і праву межі області шаблонів полів (пунктирних ліній), які треба розмножити; вертикальними стрілками встановити необхідну кількість записів, які мають виводитися і натиснути Enter. Закінчити розробку форми, натиснувши F2.

2.1. Створити екранну форму для головної таблиці Spivrob. Для цього:

Повторити дії, описані в пункті 1.1, з урахуванням нового імені таблиці (Spivrob).

2.2. Включити форму допоміжної таблиці Vypl до форми для основної таблиці Spivrob. Для цього: увійти до меню (F10); вибрати послугу Multi-Tables-Place-Linked; у відповідь на запит ввести ім'я допоміжної таблиці – Vypl; вибрати форму, попередньо створену для Vypl; у відповідь на запит Select field to match Код in Vypl із наведеного списку полів вибрати КОД – поле, за яким відбувається зв'язок таблиць. Область форми для Vypl, що з'явилась на екрані, клавішами управління курсором помістити в потрібне місце і натиснути Enter. Закінчити розробку форми натисненням клавіші F2.

3. Продивитися базу даних, використовуючи створену форму таблиці Spivrob. Для цього викликати таблицю Spivrob на екран для перегляду (View), потім вибрати послугу головного меню Image-PickForm, і вказати номер створеної форми. Для швидкого перемикання між таблицею і створеною формою (клавішею F7) вийти у меню і вибрати послугу Image-KeepSet.

5.2.6. ЗВІТИ У СИСТЕМІ PARADOX

Звіт – це об'єкт бази даних, призначений для підготовки даних з однієї або кількох таблиць бази даних до друку.

Звіти можуть виводитись засобами СУБД безпосередньо на принтер, на екран, або записуватись у текстові файли для їх подальшого опрацювання текстовими процесорами.

При розв'язуванні реальних задач звіти переважно створюються не до самих таблиць, що зберігають дані, а до результатів певних запитів.

СУБД Paradox дозволяє створювати звіти двох типів: звіти табличної форми і звіти вільної форми. Для роботи зі звітами призначена послуга головного меню **Report**.

У табличних звітах (Tabular) дані подаються у вигляді, наближеному до табличного, що полегшує виконання різноманітних обчислень підсумкових значень і створення обчислювальних полів. Звіти вільної форми (FreeForm) призначені для створення листів, запрошень і інших документів, які не потребують табличного

подання даних і дозволяють розташовувати дані з таблиць у довільній формі.

Створення звітів розглянемо на прикладі бази даних, що складається з таблиць СПІВРОБІТНИКИ і ВИПЛАТИ. Необхідно створити звіт, що містить дані з обох таблиць.

Як і у випадку форми, при створенні зв'язаних звітів спочатку створюється звіт для підлеглої таблиці, в який включаються необхідні дані з головної таблиці.

Для розв'язання поставленої задачі слід виконати наступні дії:

1.1. Створити звіт по допоміжній таблиці Vypl. Для цього:

а) увійти до головного меню СУБД Paradox; вибрати послугу Report-Design; у відповідь на запит ввести або вибрати зі списку ім'я Vypl; з наведеного списку номерів і імен звітів для даної таблиці обрати 'Unused report'; у відповідь на запит 'Report description:' ввести ім'я створюваного звіту, наприклад, Vyplreport; вибрати тип звіту – Tabular;

Як і у випадку форм, після вказування загальних характеристик Paradox переходить в режим створення звіту, який має внутрішнє меню. На екрані монітора з'являється робочий простір, в якому розміщуються пояснювальні тексти і шаблони полів. Робота з ними відбувається загалом так, як при роботі з формами.

Робочий простір розділений горизонтальними лініями на ряд областей. Назви областей виводяться у верхньому рядку екрану при встановленні курсору в ту чи іншу область. Зовнішній вигляд звіту залежить від того, в яку область поміщені ті чи інші дані. Так, в області Report Header (заголовок звіту) розміщують ті дані, які повинні з'являтися один раз на початку звіту. В області Report Footer (підвал звіту) розміщують дані, що виводяться один раз в кінці звіту.

Текст звіту розбивається на сторінки, при цьому користувач розробляє на екрані шаблон однієї сторінки, який має області Page Header (заголовок сторінки) і Page Footer (підвал сторінки), які містять дані, що повторюються відповідно на початку і в кінці кожної сторінки звіту.

На сторінці табличного звіту розташовується область Table, яка за замовчуванням містить шаблони всіх полів таблиці, розташовані в один рядок і супроводжені підписами. Склад і порядок полів може бути змінений користувачем.

У звіті вільної форми замість області Table є область Form, яка містить шаблони всіх полів таблиці з підписами, розташовані в один стовпець. Користувач може розташувати потрібні поля довільним чином.

Для більш зручного подання даних у звіті їх можна групувати за певними ознаками, зокрема, за однаковими значеннями у певному полі. У випадку створення групи на екрані з'являється відповідна область, яка включає в себе область таблиці, і містить області Group Header (заголовок групи) і Group footer (підвал групи), в яких розташовують дані, що мають виводитись на початку і в кінці кожної групи даних.

Шаблони полів для виведення в табличному вигляді поміщуються у спеціальні колонки області Table, які називаються TableBand. Для роботи з колонками призначений однойменний пункт внутрішнього меню. Колонку можна вставити зліва від поточної (TableBand-Insert), вилучити разом з даними, що в ній записані (TableBand-Erase), змінити ширину (TableBand-Reformat), перемістити в інше місце області таблиці(TableBand-Move). Робота з шаблонами полів звіту відбувається за допомогою пункту меню Field так, як при створенні форм. Поля, що розміщуються у звіті, можуть бути взяті безпосередньо з таблиці (Regular), отримані в результаті обчислень над даними з таблиці (Calculated), поля для обчислення підсумкових значень за даними з таблиці (Summary), крім того, є стандартні шаблони для номеру сторінки (Page), встановлення поточних дати (Date) і часу (Time).

На будь-якому етапі розробки звіту проміжний результат можна продивитись на екрані за допомогою команди внутрішнього меню Output-Screen. Меню Output містить крім цього пункти Printer (виведення звіту на принтер) і File (виведення звіту у текстовий файл з розширенням rpt і ім'ям, яке вказує користувач).

б) в робочому просторі, що з'явився на екрані, встановити курсор в область Report Header; (зверніть увагу на те, що на екрані з'являються всі поля з Vypl, а також стандартні поля для виведення дати друку звіту dd.mm.yy (число.місяць.рік) та номерів сторінок звіту Page 999; як назва звіту на екрані присутнє слово Vyplreport. Можна написати нову назву звіту, що буде друкуватися, наприклад, 'Звіт про виплати співробітникам'. В області

Page header вилучити шаблон для поля Дата – dd.mm.yy. Для цього увійти до меню, натиснувши F10; вибрати послугу Field-Erase, встановити курсор на шаблон і натиснути Enter. Вилучити слово Page, назву звіту, шаблон для поля Сторінка – 999. В центрі області розмістити новий шаблон для номера сторінки, якому має передувати слово Сторінка. Для цього виконати наступні дії: F10, Field-Place-Page, зафіксувати шаблон натисненням Enter, клавішами управління курсором встановити кількість цифр у шаблоні і натиснути Enter, після чого написати Сторінка.

1.2. Пов'язати таблицю Vypl із таблицею Spivrob для розміщення в звіті даних з таблиці Spivrob. Для цього:

а) виконати наступні дії: F10, Field-Lookup-Link. На запит про ім'я таблиці, що пов'язується з поточною, ввести Spivrob; На запит Select Vypl field to match Код in Spivrob із наведеного списку назв полів вибрати Код;

б) розмістити в звіті поля з таблиці Spivrob: ПІБ і ПОСАДА. Для цього: в області Table вставити порожню колонку для розміщення одного з цих полів, виконавши команди F10, Tableband-Insert. Колонка вставляється зліва від тієї колонки, в якій знаходився курсор під час виконання команди. В колонці, що з'явилась, розмістити шаблон для поля з Spivrob. Для цього виконати такі дії: F10, Field-Place-Regular; у списку назв полів таблиці Vypl, що з'явився, вибрати [Spivrob ->]. У списку назв полів з Spivrob вибрати потрібну назву (наприклад, ПРІЗВИЩЕ). Встановити курсор в потрібне місце порожньої колонки і двічі натиснути Enter, відкоригувавши довжину шаблону. Над шаблоном, що з'явився, надписати потрібну назву. Друге поле розмістити аналогічно.

в) Продивитися отриманий звіт, виконавши команди F10 => Output-Screen.

2.1. Виконати групування даних у звіті за полем ПРІЗВИЩЕ. Для цього слід виконати такі дії:

а) встановити курсор в область Page Header, виконати команди F10, Group-Insert-Field; із наведеного списку назв полів вибрати поле ПРІЗВИЩЕ з таблиці Spivrob. Клавішами управління курсором вказати місце для групи (над границею області Table) і натиснути Enter. Встановити курсор в область Group Header, написати назву поля: ПРІЗВИЩЕ, праворуч розмістити шаблон для

поля: F10, Field-Place-Regular, [Spivrob->], ПРІЗВИЩЕ, Enter, Enter. Аналогічним способом в області Group Header праворуч від розміщеного поля ПРІЗВИЩЕ розмістити поле ПОСАДА;

б) з області Table вилучити поля ПОСАДА і ПРІЗВИЩЕ за допомогою команд F10, Tableband-Erase;

в) продивитися отриманий звіт, виконавши команди F10, Output-Screen.

3.1.Підрахувати і вивести у звіті загальну суму виплат для кожного співробітника. Для цього виконати наступні дії:

а) встановити курсор в область Groop Footer; під шаблоном для поля СУМА розмістити шаблон для поля загальної зарплати одного робітника. Для цього слід виконати такі дії: F10, Field-Place-Summary-Regular, з наведеного списку вибрати поле СУМА, в меню, що з'явилось, обрати послуги Sum-Pergroup і тричі натиснути Enter, відкоригувавши довжину цілої і дробової частин числа. Зліва від шаблону написати назву поля: ВСЬОГО;

б) Продивитися отриманий звіт, виконавши команди F10, Output-Screen.

3.2.Підрахувати і вивести у звіті загальну суму виплат всім робітникам. Для цього потрібно:

а) встановити курсор в область Report Footer; під шаблоном для поля ВСЬОГО розмістити шаблон для поля загальної зарплати всіх робітників. Для цього виконати такі дії: F10,

Field-Place-Summary-Regular, з наведеного списку вибрати поле СУМА, в меню, що з'явилось, вибрати послуги Sum-Overall і тричі натиснути Enter. Зліва від шаблону написати назву поля: ЗАГАЛОМ;

б) продивитися отриманий звіт, виконавши команди F10, Output-Screen.

3.3.Для кожної виплати кожному робітникові вивести у окреме поле ПРОФВНЕСОК розмір профвнеска з виплати у розмірі 1% від суми виплати. Для цього вставити в область таблиці останню колонку (TableBand-Insert), і в ній розмістити обчислювальне поле, вибравши з меню команду Field-Place-Calculated і ввести вираз, за яким буде отримане потрібне значення: [СУМА]*0.01. Потім розмістити в колонці відповідний шаблон і підписати його.

3.4.Підрахувати кількість сплат кожному робітникові. Для цього

а)використати послугу Field-Place-Summary-Regular-Count-Pergroup). Вивести отримані значення у поле ЗАГАЛЬНА КІЛЬКІСТЬ СПЛАТ РОБІТНИКОВІ.

б) продивитися отриманий звіт, виконавши команди F10, Output-Screen.

в) для закінчення розробки звіту і збереження результатів натиснути F2.

Для перегляду звіту в будь-який момент роботи з СУБД слід вибрати з головного меню команду Report-Output-Screen, після чого вказати ім’я таблиці і назву звіту.

5.3. СУБД MICROSOFT ACCESS

На сьогодні найбільш поширеною СУБД для сучасних комп'ютерів є реляційна СУБД ACCESS (доступ), яка входить до комплекту програм MS Office. Інтерфейс програми є стандартизованим для вказаного комплекту, отже, для роботи з нею необхідні навики, отримані при вивченні інших програм MS Office (Word, Excel). В СУБД ACCESS кожна база даних займає один файл, який має розширення .MDB. У файлі містяться таблиці та інші об'єкти бази даних. Основи роботи з СУБД ACCESS розглянемо на тому ж прикладі, що мав місце для СУБД Paradox.

Приклад. Нехай предметна область – деяка фірма. Створимо базу даних, яка зберігає дані про її співробітників і зроблені їм виплати заробітної плати протягом деякого часу. Об′єктами предметної області є співробітники фірми, серед їх властивостей, що мають бути відображені у базі даних, визначимо такі:

– ідентифікаційний код;
– прізвище, ім′я, по-батькові;
– дата народження;
– домашня адреса;
– посада у фірмі;
– стаж роботи;
– дата отримання заробітної плати;
– отримана сума заробітної плати.

База даних, яку назвемо Firma, міститиме дві основні таблиці: СПІВРОБІТНИКИ(КОД*, ПІБ, ДАТА НАРОДЖЕННЯ, АДРЕСА, ПОСАДА, СТАЖ) і ВИПЛАТИ(КОД*, ДАТА ВИПЛАТИ*, СУМА) і допоміжну таблицю-довідник ШТАТНИЙ РОЗКЛАД(ПОСАДА*), яка буде використовуватись для полегшення введення даних у таблицю Співробітники.

5.3.1. СТВОРЕННЯ НОВОГО ФАЙЛА БАЗИ ДАНИХ

Для створення нового файла бази даних слід виконати такі дії:

запустити програму MS ACCESS;

із запропонованих способів початку роботи з програмою (Новая база данных, Запуск мастера, Открыть базу данных) вибрати опцію Новая база данных;

у вікні Создание базы данных ввести ім’я нового файла (Firma), у полі Папка вибрати папку, в якій зберігатиметься файл, натиснути кнопку Создать (див. рис. 44). Вся подальша робота з базою даних буде відбуватися у створеному файлі.

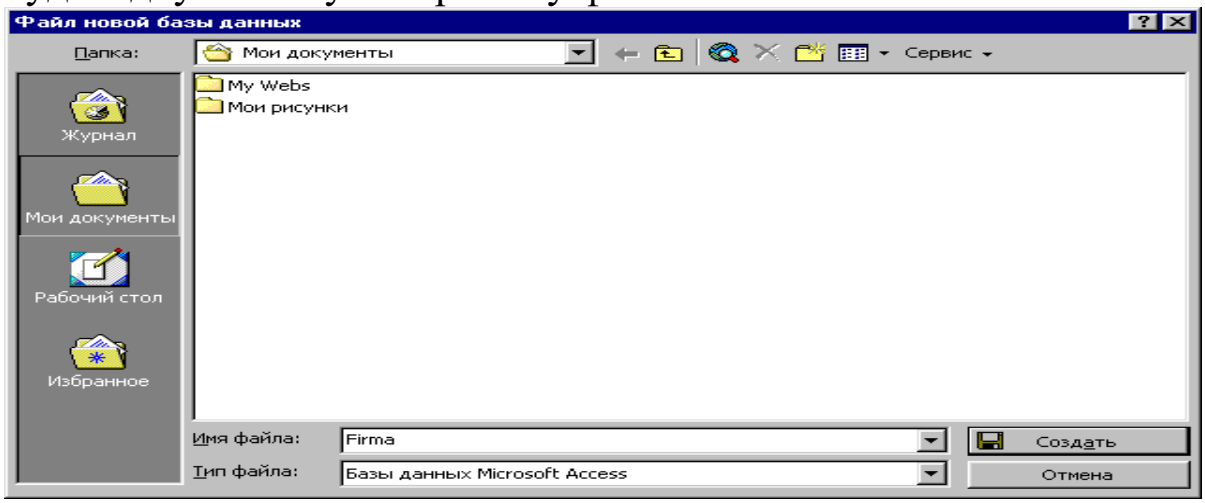


Рис.44.Вікно створення і збереження нового файла бази даних

Після створення нового файла на екран виводиться вікно бази даних (рис.45). Це вікно має ряд закладок, що відповідають об’єктам бази даних (Таблицы, Запросы, Формы, Отчеты, Макросы, Модули). Отже, для роботи з певними об’єктами слід перейти на відповідну закладку.

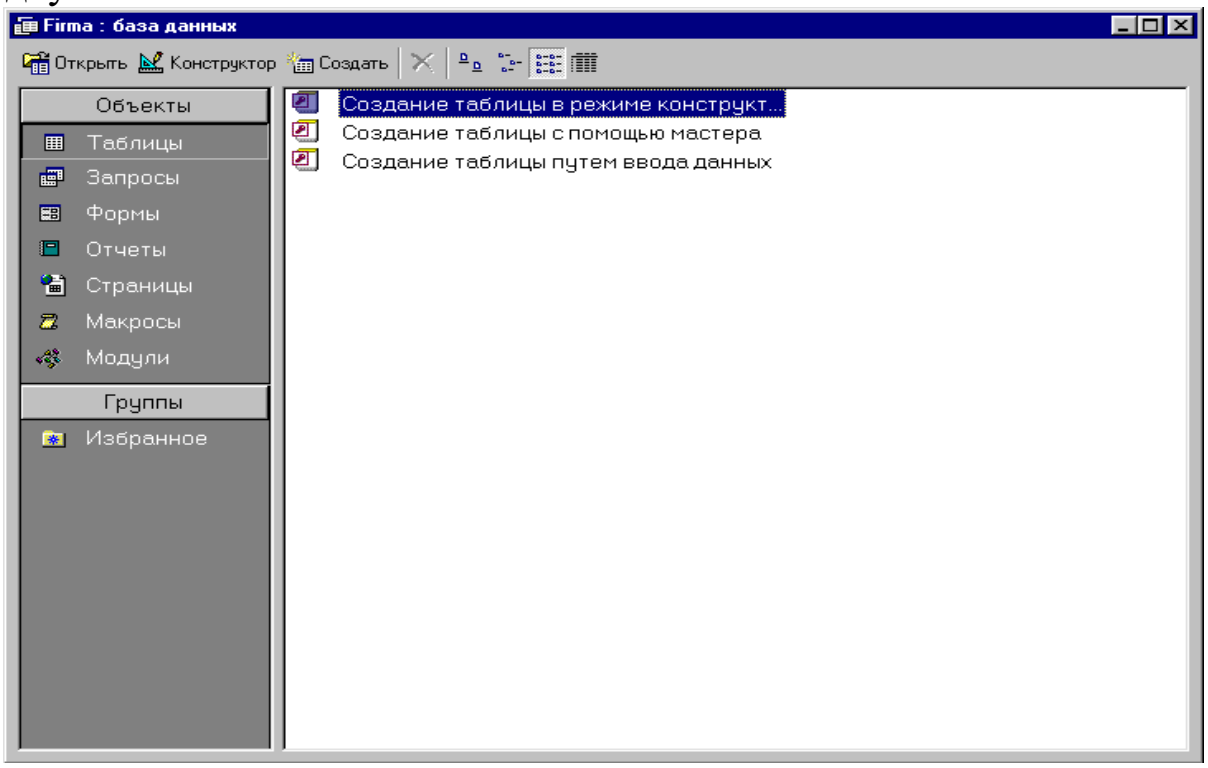


Рис.45. Вікно файла бази даних

Об’єкти бази даних подаються у вікні піктограмами з підписами. Як і інші об’єкти Windows, об’єкти бази даних зручно опрацьовувати за допомогою контекстного меню. Крім цього, на кожній закладці є власне меню і спеціальні кнопки управління об’єктами: Открыть (або Просмотр, Запуск – в залежності від особливостей об’єкта), Конструктор, Создать.

Кнопка Создать призначена для створення нового об’єкта ; Открыть – для відкриття існуючого об’єкта у тому вигляді, в якому він використовується при роботі з базою даних; Конструктор – для відкриття об’єкта в режимі конструктора, який дозволяє створювати і змінювати структуру об’єкта. Перехід між режимами конструктора і власне об’єкта можна здійснити в будь-який момент роботи з об’єктом, скориставшись кнопкою Вид на стандартній панелі інструментів.

5.3.2. РОБОТА З ТАБЛИЦЯМИ БАЗИ ДАНИХ

Загальні прийоми роботи з об’єктами бази даних можна засвоїти на прикладі роботи з таблицями.

ACCESS надає декілька способів створення таблиць, проте найбільш доцільно дотримуватись двохетапної схеми, яка передбачає: а) створення структури таблиці, б) заповнення таблиці даними. Перший етап відбувається в режимі конструктора, другий – в режимі таблиці.

Для створення нової таблиці слід виконати такі дії:

– на закладці Таблицы натиснути кнопку Создать; у вікні, де пропонуються різні способи створення таблиць (рис.46) вибрати Конструктор і натиснути Ок;
– у вікні Конструктора таблиць ввести імена полів, їх типи і властивості, визначити ключові поля;
– зберегти таблицю у файлі бази даних, вказавши її ім’я;
– перейти в режим таблиці, натиснувши кнопку Вид на стандартній панелі інструментів;
– внести у таблицю необхідні дані;
– закрити вікно таблиці.

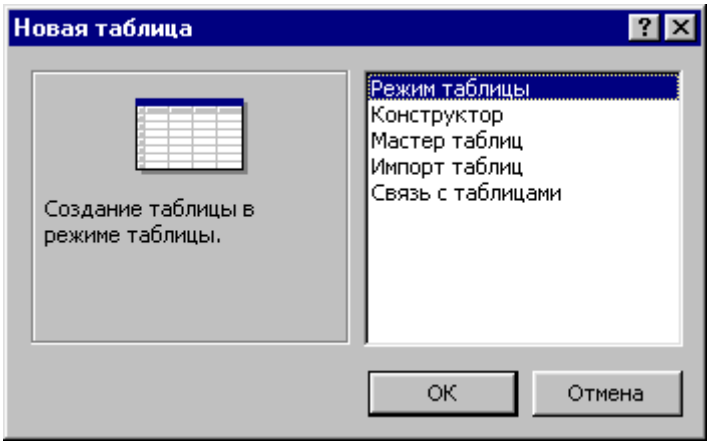


Рис.46.Вікно вибору способу створення нової таблиці

Створення бази даних Firma розпочнемо з таблиці ШТАТНИЙ РОЗКЛАД, яка має найпростішу структуру.

Вікно Конструктора таблиць складається з двох частин (рис.47).

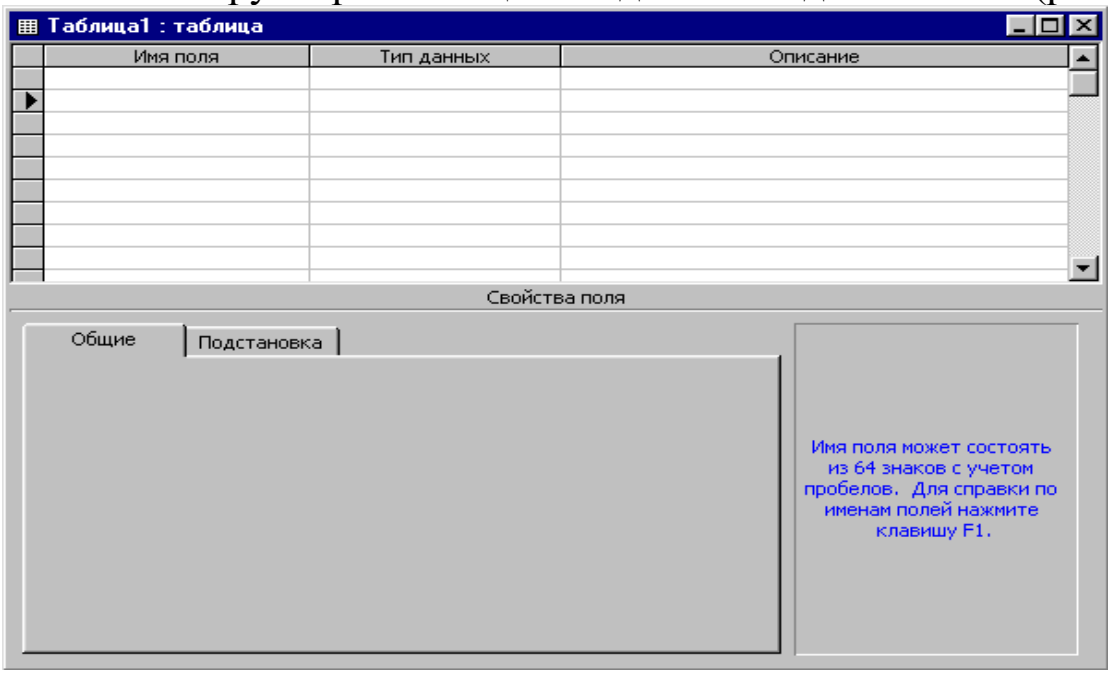


Рис.47. Конструктор таблиць

Верхня частина являє собою таблицю, в якій обов'язково слід заповнити стовпці Имя поля і Тип данных у відповідності з запланованою структурою таблиці. При цьому імена полів вводяться вручну, а тип даних для кожного введеного поля ACCESS автоматично відмічає як текстовий. Щоб встановити інший тип даних, слід перейти у клітинку з назвою типу. У клітинці з'являється кнопка розкриття списку, що містить всі доступні у ACCESS типи даних. Розкривши список, слід вибрати потрібний тип (рис.48).

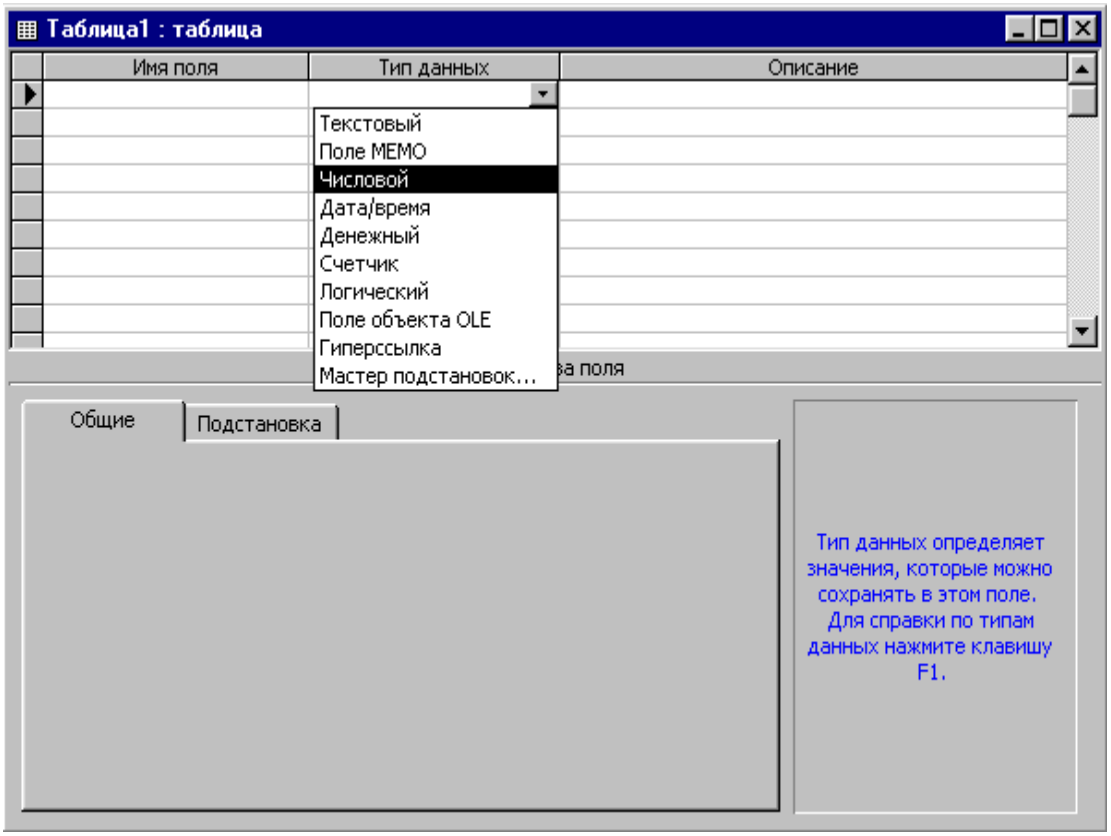


Рис.48.Вибір типів даних для полів таблиці

Основні типи даних СУБД ACCESS:

- текстовий (рядки довжиною до 255 символів);
- числовий (включає різні числові типи, зокрема байт, ціле, довге ціле, дійсне тощо);
- дата/час (можна встановлювати різні формати подання дати і часу);
- грошовий (до введених чисел автоматично дописуються десяткові знаки і позначення грошової одиниці);
- лічильник (порядковий номер запису);
- логічний (має два значення – істинне і хибне);
- мемо (для зберігання великих обсягів текстових даних);
- об’єкт OLE (для збереження об’єктів, створених в інших програмах: графічних зображень, звукових і відео фрагментів тощо);
- гіперпосилання (для збереження адреси в мережі Інтернет);
- майстер підстановок (для вибору даних у полі з іншої таблиці або списку).

Крім імені і типу, поле таблиці може мати ряд властивостей, перелік яких можна продивитися або встановити в нижній частині вікна Конструктора таблиць. Для роботи з властивостями певного поля треба попередньо активізувати це поле у верхній частині Конструктора.

Основними властивостями полів таблиць є: розмір поля, формат поля, значення за замовчуванням, умова на значення, повідомлення про помилку, обов'язкове поле, порожні рядки, індексоване поле.

Поле ПОСАДА створюваної таблиці має текстовий тип. За замовчуванням встановлюється довжина текстового поля – 50 символів (властивість Размер поля). Нехай у нашій таблиці розмір поля дорівнює 15 (рис.49).

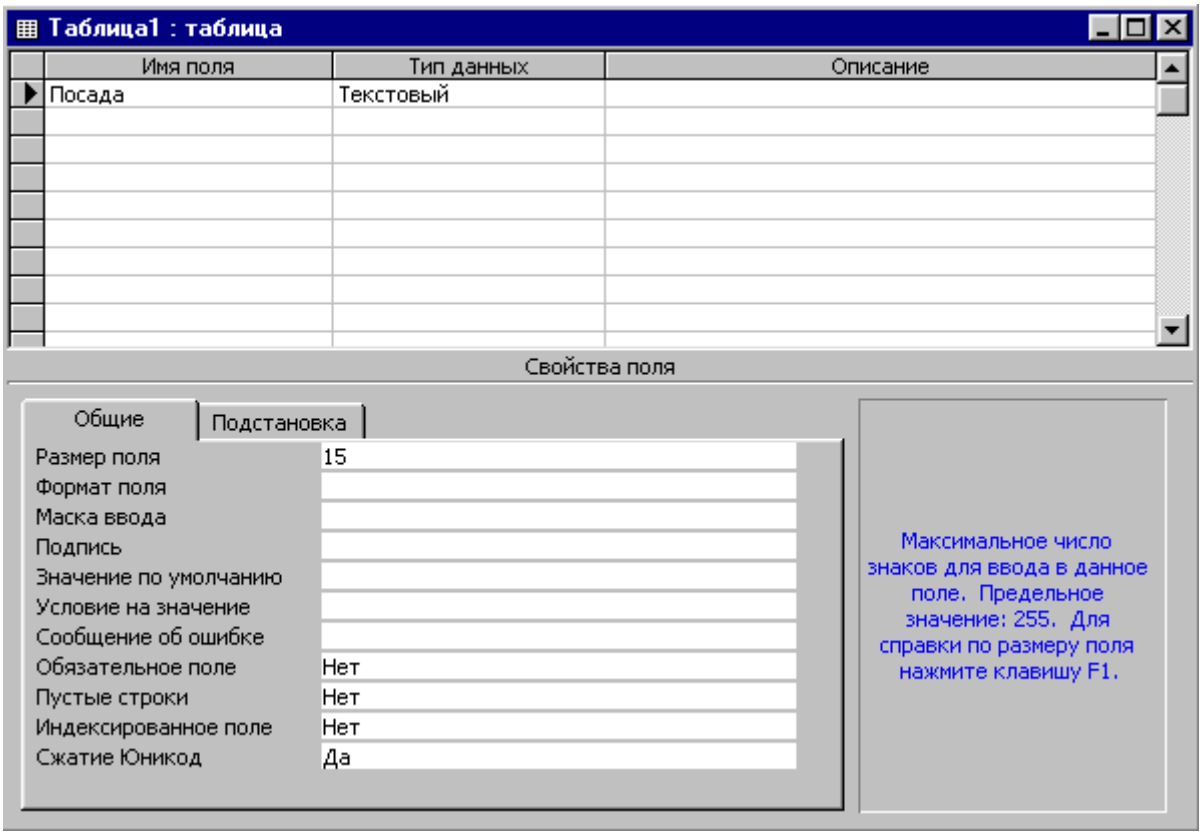


Рис.49.Структура таблиці Штатний розклад

Всі посади штатного розкладу повинні бути різними, тому єдине поле таблиці – ПОСАДА – слід зробити ключовим. Для визначення ключового поля слід виділити відповідний йому рядок у верхній частині Конструктора таблиць і натиснути на панелі інструментів кнопку Ключевое поле. Зліва від назви поля з'являється зображення ключа.

Зауваження. У випадку складеного ключа слід виділити всі поля, що складають ключ, а потім один раз натиснути кнопку Ключевое поле. Дана кнопка діє як перемикач, отже, для відмови від ключового поля слід натиснути її повторно.

Після створення структури таблиці необхідно зберегти її на диску під потрібним ім'ям (за замовчуванням ACCESS називає таблиці

Таблица1, Таблица2 …). Для цього слід натиснути кнопку Сохранить на стандартній панелі інструментів, потім у діалоговому вікні ввести ім'я таблиці і натиснути Ок. Запит щодо збереження таблиці з'являється також при спробі закрити вікно таблиці або перейти в інший режим роботи з нею, якщо структура таблиці терпіла певні зміни (зокрема створювалася).

В розглядуваному прикладі назвемо таблицю ШТАТНИЙ РОЗКЛАД. Для внесення даних у таблицю слід перейти з режиму конструктора у режим таблиці за допомогою кнопки Вид на стандартній панелі інструментів.

Зауваження. В процесі внесення даних у таблицю в кожному записі слід спочатку заповнювати ключові поля, а також поля, для яких встановлена властивість Обязательное поле: Да.

Дані, що вносяться у записи таблиці, відразу записуються на диск, отже, немає потреби у зберіганні проміжних результатів заповнення таблиці.

За розглянутою схемою створимо структури двох інших таблиць бази даних Firma: СПІВРОБІТНИКИ і ВИПЛАТИ. Нехай щодо структури і вмісту цих таблиць висунуті такі умови:

Таблиця СПІВРОБІТНИКИ.

Поле Код є числовим, ключовим; поля КОД і ПІБ повинні бути заповнені, поле ПІБ повинно містити не більше 30 символів; дата народження повинна вводитись у форматі дд.мм.рр. і знаходитись в межах від 1 січня 1930 р. до поточної дати, за замовчуванням повинна встановлюватись поточна дата; поле АДРЕСА повинно містити не більше 40 символів; поле ПОСАДА повинно містити не більше 15 символів, причому значення для цього поля повинні контролюватися таблицею ШТАТНИЙ РОЗКЛАД; значення в полі СТАЖ повинні знаходитись в межах від 1 до 50 років, за замовчуванням – 1 рік, при введенні неправильного значення стажу повинно виводитись повідомлення "Введіть значення від 1 до 50!"

Таблиця ВИПЛАТИ.

Поля КОД і ДАТА ВИПЛАТИ є ключовими, обов'язково повинні бути заповнені, значення у полі КОД повинно контролюватися таблицею СПІВРОБІТНИКИ; дата виплати повинна вводитись у форматі дд.мм.рр., поле СУМА повинно мати грошовий тип.

Для виконання цих умов треба заповнити конструктор таблиць такими даними (рис 50).

Співробітники

Назва поля	Тип даних	Властивості поля
🔑Код	Числовой	Размер поля: Целое Обязательное поле: Да
ПІБ	Текстовый	Размер поля: 30 Обязательное поле: Да
Дата народження	Дата/Время	Формат поля: Краткий формат даты Условие на значение: >=#01.01.30# and <= Date() Значение по умолчанию: Date()
Адреса	Текстовый	Размер поля: 40
Посада	Текстовый	Размер поля: 15 Мастер подстановок… …значения из таблицы или запроса ШТАТНИЙ РОЗКЛАД Посада
Стаж	Числовой	Размер поля: Байт Условие на значение: >=1 and <=50 Значение по умолчанию: 1 Сообщение об ошибке: Введіть значення від 1 до 50!

Виплати

Назва поля	Тип даних	Властивості поля
🔑Код	Числовой	Размер поля: Целое Обязательное поле: Да Мастер подстановок… …значения из таблицы или запроса... ...СПІВРОБІТНИКИ... ...Код...
🔑Дата виплати	Дата/Время	Формат поля: Краткий формат даты
Сума	Денежный	

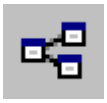
Рис.50. Структури таблиць Співробітники і Виплати

5.3.3. СХЕМА ДАНИХ

Між таблицями, що утворюють базу даних, як правило, існують логічні зв'язки (один-до-одного, один-до-багатьох тощо). В розглянутій вище СУБД Paradox ці зв'язки відслідковувались самим користувачем і активізувались при роботі з певними об'єктами бази даних: запитами, формами, звітами. У СУБД ACCESS міжтабличні

зв’язки фіксуються, автоматично контролюються і візуально подаються завдяки спеціальному об’єкту бази даних, який має назву Схема даних. На відміну від інших об’єктів БД, Схема даних не представлена у вікні бази даних. Для виклику її вікна слід виконати команду з меню Сервис-Схема данных, або скористатися однойменною кнопкою на стандартній панелі інструментів .

Створення схеми даних розглянемо на прикладі створення зв’язку “один-до-багатьох” між таблицями СПІВРОБІТНИКИ і ВИПЛАТИ. При першому відкритті порожньої схеми даних на екран видається вікно Добавление таблицы, в якому треба вибирати назви таблиць, між якими будуть встановлені зв’язки, і натисненням кнопки Добавить розміщувати графічні позначення цих таблиць у вікні схеми даних. Після розміщення всіх потрібних таблиць слід закрити вікно Добавление таблицы.



Зауваження 1.При потребі вивести на екран вікно Добавление таблицы *в процесі роботи зі схемою даних (а також при роботі в конструкторі запитів) слід натиснути кнопку* Добавление таблицы *на стандартній панелі інструментів.*

Зауваження 2. Перед створенням схеми даних слід закрити вікна всіх таблиць, що будуть задіяні в ній.

Для встановлення зв’язку між таблицями СПІВРОБІТНИКИ і ВИПЛАТИ необхідно у вікні схеми даних встановити курсор миші на ключове поле таблиці сторони відношення “один”, а саме, на поле КОД таблиці СПІВРОБІТНИКИ, і перетягти на однойменне ключове поле таблиці сторони відношення “багато” (ВИПЛАТИ). По закінченні перетягування на екран виводиться вікно Изменение связей (рис.51).

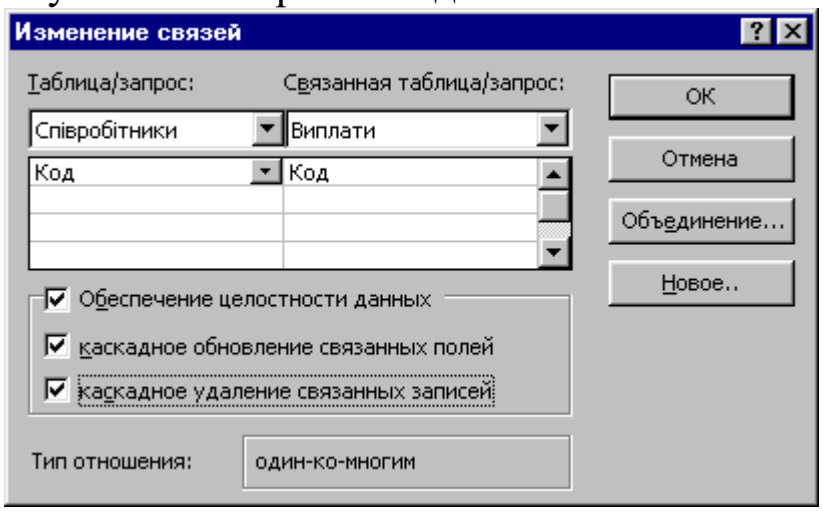


Рис.51. Вікно управління зв’язком між таблицями

В цьому вікні виводяться відомості про таблиці, що підлягають зв'язуванню, поля, за якими відбувається зв'язування, і тип зв'язку, який може бути встановлений в даному випадку. Слід пам'ятати, що поля, за якими зв'язуються таблиці, повинні мати один і той самий тип. У вікні Связи можуть бути активізовані засоби контролю за цілісністю і несуперечливістю даних (перемикач Обеспечение целостности данных). Контроль цілісності даних дозволяє автоматично виконувати дві такі важливі операції над даними у зв'язаних таблицях, як каскадне вилучення зв'язаних записів і каскадне оновлення зв'язаних полів (перемикачі Каскадное удаление связанных записей, Каскадное обновление связанных полей).

Каскадне вилучення зв'язаних записів: при вилученні певного запису з головної таблиці автоматично вилучаються всі відповідні записи (ті, що мають спільний ключ) з підлеглої таблиці. Так у базі даних зберігається посилальна цілісність даних.

Каскадне оновлення зв'язаних полів: при зміні значення у певному полі головної таблиці автоматично змінюються відповідні значення у полях підлеглої таблиці. Так зберігається несуперечливість даних.

Встановивши необхідні параметри зв'язку, слід натиснути кнопку Создать. Вікно Связи закривається, а зв'язок між таблицями зображується у вікні схеми даних у вигляді лінії між позначеннями таблиць. Написи на кінцях лінії показують тип відношення (рис.52).

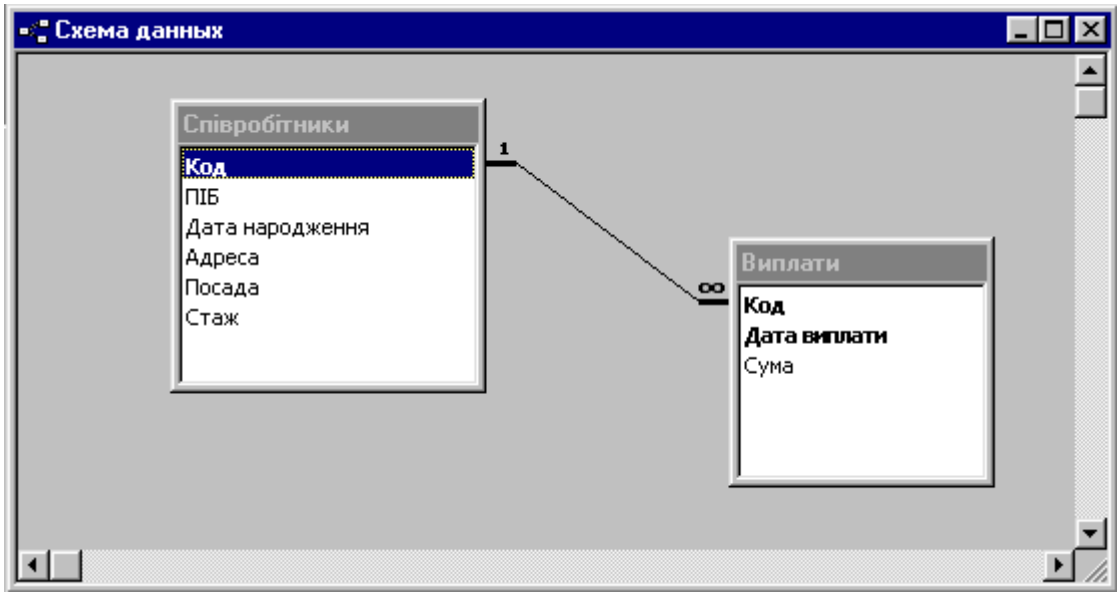


Рис.52. Вікно схеми даних для бази даних Firma

При потребі відредагувати зв'язок слід виділити лінію і в її контекстному меню вибрати команду Изменить связь. Для вилучення

зв’язка або позначення таблиці треба виділити потрібне зображення і натиснути клавішу Delete.

Після створення схеми даних слід зберегти її на диску і закрити її вікно.

Заповнення зв’язаних таблиць даними відбувається після створення схеми даних. Для внесення даних таблицю слід відкрити, виділивши її на закладці Таблицы і натиснувши кнопку Открыть.

5.3.4. ЗАПИТИ У СУБД ACCESS

Запит у СУБД ACCESS – це об′єкт бази даних, призначений для вибірки даних із таблиць за певними критеріями, виконання обчислень над даними з таблиць, додавання, вилучення і зміни даних у таблицях, подання даних у компактному вигляді, подібному до електронних таблиць.

ACCESS підтримує мови запитів QBE (Query By Example) і SQL (Structured Query Language) та забезпечує перехід між двома формами завдання запиту. Більш зручною для користувача є мова QBE, в якій запити являють собою спеціальним чином заповнені бланки.

Всі дії з запитами у СУБД ACCESS виконуються на закладці Запросы вікна бази даних. Кожен запит, що зберігається у файлі, зображується піктограмою з назвою запиту. Для опрацювання запиту можна використовувати контекстне меню і кнопки управління запитами. Создать – для створення нового запита; Открыть – для перегляду таблиці–результату запиту (у випадку запитів на вибірку інформації) або виконання певної дії з таблицями бази даних (у випадку запиту на зміну даних); Конструктор – для відкриття запиту в режимі конструктора, який дозволяє заповнювати бланк QBE, а також проглядати і редагувати його. Як і при роботі з таблицями, кнопка Вид на стандартній панелі інструментів забезпечує перемикання між режимами запиту і конструктора.

Запити на вибірку інформації

Запити на вибірку є найпростішим типом запитів у СУБД ACCESS. Такі запити створюються за замовчуванням і дозволяють вибирати з таблиць і інших запитів дані, що відповідають певним критеріям.

Початкові дії по створенню запитів всіх типів можна засвоїти на прикладі запитів на вибірку.

Для створення запиту на вибірку слід виконати такі дії:

– на закладці Запросы натиснути кнопку Создать; у вікні, де пропонуються різні способи створення запитів (рис.53) вибрати Конструктор і натиснути Ok;
– у вікні Добавление таблицы вибрати таблиці і запити, до яких буде робитися запит (робота з цим вікном відбувається так само, як при роботі зі схемою даних);
– у вікні Конструктора запитів заповнити бланк запиту, визначивши поля, що беруть участь у запиті, встановивши позначки для виведення полів та умови відбору даних;
– виконати запит натисненням кнопки Запуск на стандартній панелі інструментів;
– при потребі зберегти запит у файлі бази даних натисненням кнопки Сохранить і закрити його;
– якщо результат запиту є незадовільним, перейти в режим Конструктора натисненням кнопки Вид і відредагувати бланк запиту.

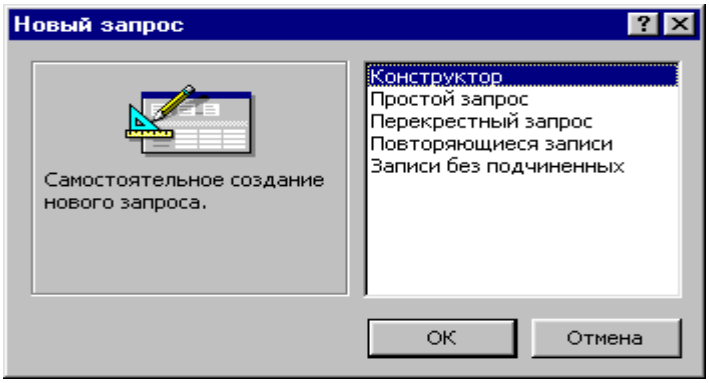


Рис.53.Вікно вибору способу створення нового запиту

Конструктор запитів складається з двох частин (рис.54). У верхній частині розміщуються графічні зображення таблиць, до яких робиться запит. Якщо між таблицями існують зв'язки, вони зображуються як на схемі даних. Можна створити і нові зв'язки. В нижній частині розташований власне бланк запиту, який підлягає заповненню.

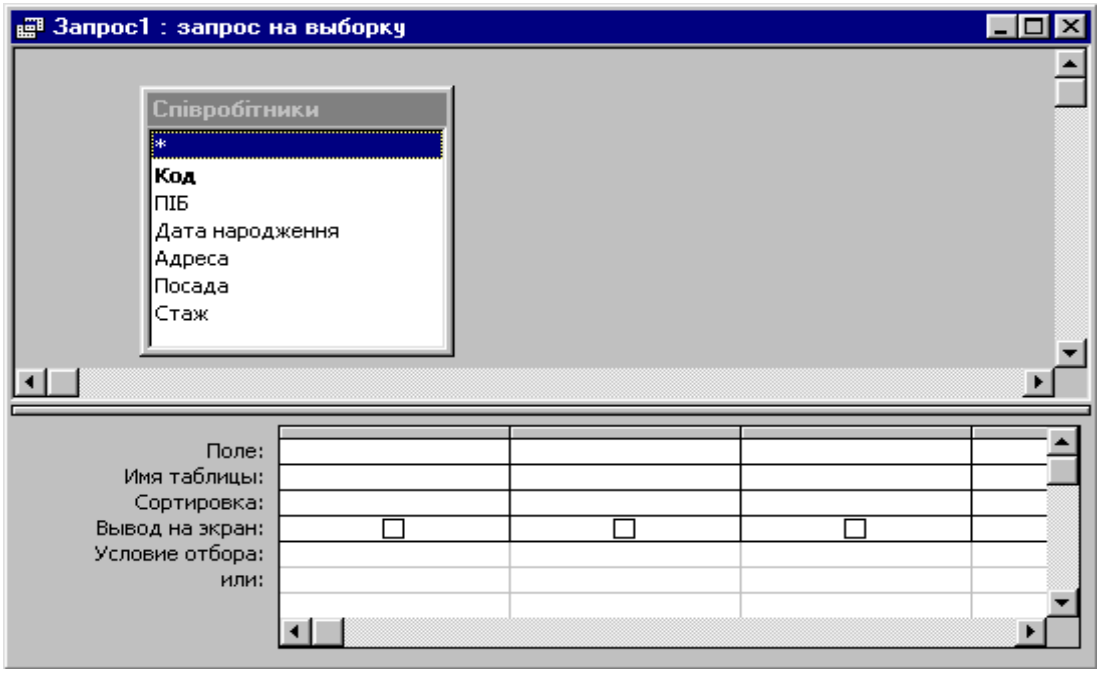


Рис.54.Вікно Конструктора запитів

Рядки бланку мають спеціальні назви, що вказують на його структуру (вони можуть розрізнятися в залежності від типу запиту).

Тип запиту можна вибрати, використавши команду меню Запрос. Доступними є такі типи запитів:

Выборка – вибір інформації з об'єктів бази даних за певними критеріями;

Перекрестный – подання результатів запиту у вигляді, аналогічному електронним таблицям;

Создание таблицы – запит на створення нової таблиці;

Обновление – запит на зміну даних у таблицях;

Добавление – запит на додавання до таблиці нових записів;

Удаление – запит на вилучення з таблиці записів, що відповідають певним умовам.

При відкритті конструктора запитів за замовчуванням створюється середовище для створення запиту на вибірку. Для створення запитів інших типів слід скористатися меню Запрос.

Розглянемо загальні принципи роботи з конструктором запитів на прикладі запитів на вибірку інформації.

Стовпці бланку слід заповнювати у відповідності з цією структурою, причому у запитах на вибірку кожний стовпець відповідає за виведення певного поля в результаті запиту. Кількість стовпців бланку необмежено продовжується вправо. При потребі вставити новий стовпець між заповненими або вилучити заповнений

стовпець зручно користуватися контекстним меню стовпця або пунктом меню Правка.

Для кожного стовпця бланку слід встановити такі характеристики:

– Поле: вказується ім'я поля таблиці, що братиме участь у запиті. Для введення імені можна клацнути у клітинці бланку, і, скориставшись кнопкою розкриття списку, вибрати потрібне ім'я поля (рис.55). Крім того, можна перетягти потрібне поле з графічного зображення таблиці у верхній частині конструктора у клітинку бланку;
– Имя таблицы: автоматично встановлюється у клітинку під назвою поля;
– Сортировка: в цій клітинці бланку можна розкрити список, що дозволяє вибрати спосіб сортування даних у вказаному полі: По возрастанию, По убыванию, Отсутствует;
– Групповая операция (цей рядок виводиться у бланку, якщо на стандартній панелі інструментів натиснути кнопку Групповая операция): дозволяє логічно групувати однакові дані у полі (елемент розкритого списку Группировка) або виконувати над даними з поля певні підсумкові обчислення (**Sum** – сума, **Avg** – середнє арифметичне, **Min** – мінімальне значення у полі, **Max** – максимальне значення у полі і ін.);
– Вывод на экран: в цій клітинці встановлюється прапорець, якщо поле повинно бути виведено на екран в результаті запиту. При розміщення нового поля на бланку прапорець встановлюється автоматично;
– Условие отбора: в цій клітинці записується умова для відбору даних (види умов будуть розглянути нижче);
– Или: записується складова умови "або", що стосується даних в одному полі або різних полях.

–

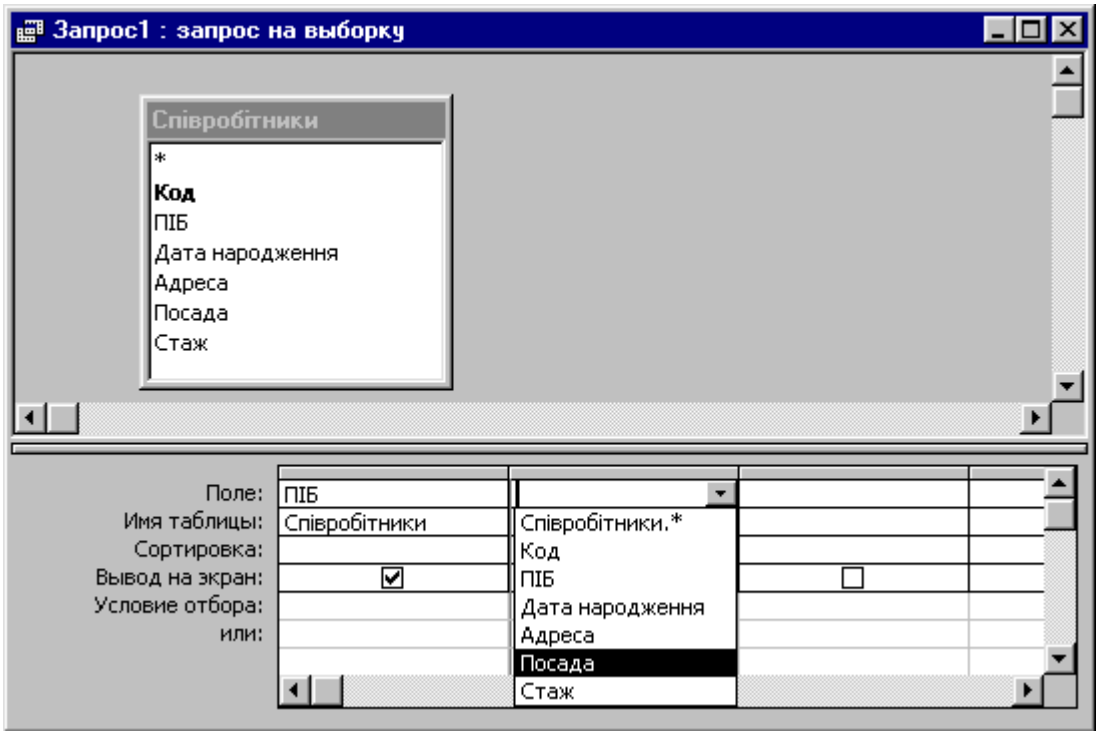


Рис.55. Вибір поля у бланку запиту

Роботу з запитами розглянемо на прикладі бази даних `Firma`. Будемо розв'язувати такі задачі, як у випадку СУБД `Paradox`.

Запит 1. Сформувати список співробітників з указанням їх посад.

Це найпростіша задача на знаходження проекції відношення СПІВРОБІТНИКИ. За правилами мови `QBE` у відповідному бланку запиту слід задати виведення на екран двох полів таблиці: ПІБ і ПОСАДА.

При створенні запиту слід у вікні Добавление таблицы вибрати таблицю СПІВРОБІТНИКИ.

Відповідне вікно конструктора запиту зображено на рис.56.

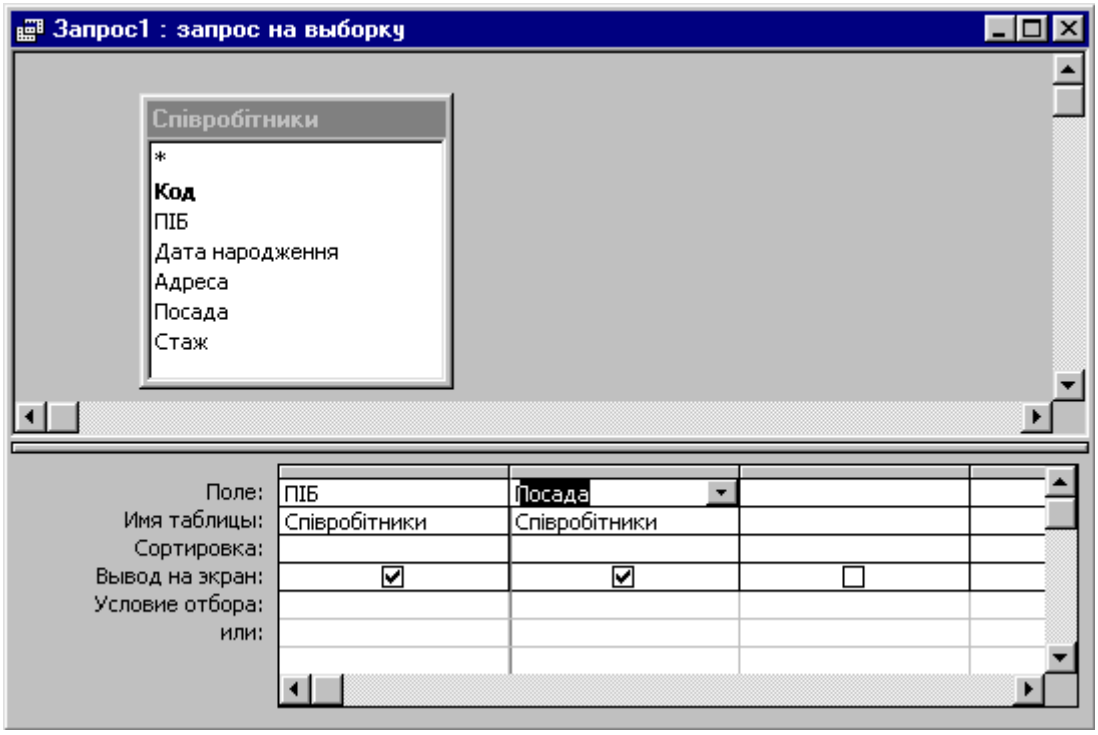


Рис.56.Вікно Конструктора для запиту 1

Для задання умов щодо даних числових, грошових типів і типу дата/час застосовуються оператори порівняння >, <, >=, <=, = (не пишеться).

Для задання кон'юнкції ("і") застосовується оператор **And**, для заперечення ("не") – **Not**, для диз'юнкції ("або") – **Or**, складові диз'юнкції можуть також записуватись в різних рядках бланку запиту.

Логічна константа **Null** (Blank у СУБД Paradox) визначає відсутність значення у полі.

Константи типу дата/час беруться в позначки #, наприклад #10.10.02#. Функція **Date()** повертає значення поточної дати (Today у СУБД Paradox) .

Для задання подвійних нестрогих нерівностей (>=…and <=…) можна застосовувати оператор діапазону Between найменше значення And найбільше значення, наприклад, умови >=1 and <=5 і between 1 and 5 еквівалентні.

Для задання умов щодо текстових полів можна застосовувати символи шаблонів * і ?, які мають загальноприйняте значення (відповідно довільна кількість довільних символів і довільний єдиний символ). Якщо потрібний повний збіг з умовою відбору, відповідний текст записується без символів шаблонів При вказуванні шаблону ACCESS автоматично бере його в лапки і дописує перед шаблоном

оператор **Like** (подібний). Наприклад, при введенні умови відбору *бухгалтер вона перетвориться у `Like` “*бухгалтер”.

Запит 2. Сформувати список даних про співробітників (ПІБ, ДАТА НАРОДЖЕННЯ, ПОСАДА, СТАЖ), стаж роботи яких: а) не менше п’яти років; б) від 10 до 15 років; в) один рік або більше 20 років.

Вміст бланків відповідних запитів зображено на рис.57.

Поле	ПІБ	…	Стаж
Имя таблицы	Співробітники	…	Співробітники
Вывод на экран	☑	…	☑
Условие отбора			>=5
Или			

а)

Поле	ПІБ	…	Стаж
Имя таблицы	Співробітники	…	Співробітники
Вывод на экран	☑	…	☑
Условие отбора			>=10 and <=15
Или			

б)

Поле	ПІБ	…	Стаж
Имя таблицы	Співробітники	…	Співробітники
Вывод на экран	☑	…	☑
Условие отбора			1
Или			>20

в)

Рис.57. Бланки запитів 2а), 2б), 2в)

Запит 3. Вивести всі дані про директора організації. Вміст бланку запиту зображено на рис.58.

Поле	Код	...	Посада	…
Имя таблицы	Співробітники	…	Співробітники	…
Вывод на экран	☑	…	☑	…
Условие отбора			Like "директор"	
Или				

Рис.58. Бланк запиту 3

Запит 4. Сформувати список даних про співробітників (ПІБ, ДАТА НАРОДЖЕННЯ, АДРЕСА), що проживають на проспекті Миру. Вміст бланку запиту зображено на рис.59.

Поле	…	Адреса	…
Имя таблицы	…		…
Вывод на экран	…	☑	…
Условие отбора		Like "*Просп. Миру*"	
Или			

Рис.59. Бланк запиту 4

Запит 5. Відомо, що в таблиці СПІВРОБІТНИКИ поле АДРЕСА заповнене не для всіх співробітників. Вивести всі відомості про співробітників, для яких це поле заповнене. Вміст бланку запиту зображено на рис. 60.

Поле	…	Адреса	…
Имя таблицы	…		…
Вывод на экран	…	☑	…
Условие отбора		Not Null	
Или			

Рис.60. Бланк запиту 5

Запити до зв’язаних таблиць

При потребі створити запит до кількох таблиць (або запитів) їх слід вибрати у вікні Добавление таблицы, так як при створенні схеми даних. Вибрані таблиці зображуються у верхній частині конструктора запитів, якщо між ними є зв’язки, вони також відображуються. Вибір полів на бланку запиту відбувається так само, як при створенні запиту до одної таблиці.

Запит 6. Вивести відомості про всі виплати всім співробітникам. В результаті повинні міститися поля ПІБ, ДАТА ВИПЛАТИ, СУМА.

Бланк запиту вказаний на рис.61.

Поле	ПІБ	Дата виплати	Сума
Имя таблицы	Співробітники	Виплати	Виплати
Вывод на экран	☑	☑	☑
Условие отбора			
Или			

Рис.61. Бланк запиту 6

Обчислення у запитах

У СУБД ACCESS існують два види обчислень у запитах:

– створення нових обчислювальних полів;
– обчислення підсумкових значень.

Для створення обчислювального поля слід у бланку запиту в рядку Поле ввести конструкцію виду:

<Назва_обчислювального_поля> : <Вираз>

Вираз задає правило, за яким обчислюються значення у створюваному полі. Він може містити знаки математичних операцій, дужки, стандартні функції ACCESS, імена полів таблиці, до якої створюється запит, взяті в квадратні дужки. Якщо треба використати поле іншої таблиці, вказується конструкція [Ім'я_таблиці].[Ім'я_поля].

Запит 7. Вивести дані про розміри всіх виплат співробітнику Петренку П.П. за останні 60 днів з указанням відрахувань у пенсійний фонд (2%). Вміст бланку запиту зображено на рис.62.

Поле	ПІБ	Дата виплати	Сума	Пенсійний фонд: CCur([Сума]*0,02)
Имя таблицы	Співробітники	Виплати	Виплати	
Вывод на экран	☑	☑	☑	☑
Условие отбора	"Петренко П.П."	>Date()-60		
Или				

Рис.62. Бланк запиту 7

Результат запиту вказаний на рис.63.

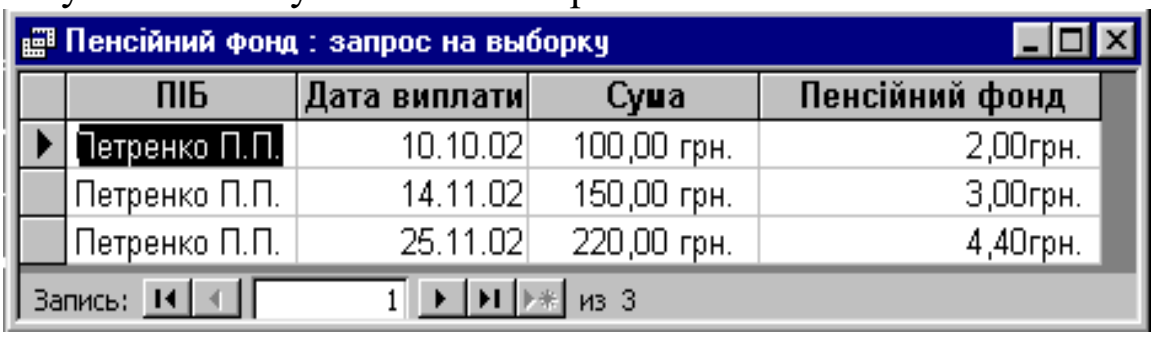


Рис.63. Результат виконання запиту 7

Зауваження. Функція CCur переводить значення числового типу у значення грошового типу, додаючи два знаки після коми і позначення грошової одиниці. Без використання цієї функції значення у полі Пенсійний фонд виглядають як звичайні цілі числа.

Для обчислення підсумкових значень необхідно включити у бланк запиту рядок Групповая операция натисненням однойменної кнопки на стандартній панелі інструментів.

СУБД ACCESS дозволяє виконувати такі групові операції над даними:

Группировка – операція призначена для групування даних у тому полі, в якому вона встановлена (відповідні записи виводяться підряд);

Sum – знаходження суми значень у відповідному полі;

Avg (Average) – знаходження середнього арифметичного;

Min – знаходження мінімального значення у полі;

Max – знаходження максимального значення у полі;

Count – знаходження кількості записів;

StDev – знаходження середньоквадратичного відхилення;

Var – знаходження дисперсії;

First – знаходження першого значення;

Last – знаходження останнього значення.

Запит 8. Отримати загальну суму виплат, зроблених кожному співробітнику. У відповіді вивести поля ПІБ, ЗАГАЛЬНА СУМА.

Бланк запиту зображений на рис. 64.

Поле	Код	ПІБ	Сума
Имя таблицы	Співробітники	Співробітники	Виплати
Групповая операция	Группировка	Группировка	Sum
Вывод на экран		☑	☑
Условие отбора			
Или			

Рис.64. Бланк запиту 8

При виконанні таких запитів ACCESS автоматично присвоює ім'я підсумковому полю за правилом Назва_групової_операції_Назва_поля_таблиці (у розглянутому випадку: Sum_Сума). Щоб присвоїти полю бажане ім'я, слід у бланку запиту встановити вказівник миші на назві поля, в якому обчислюється підсумкове значення (в нашому прикладі це поле Сума), викликати контекстне меню поля і вибрати команду Свойства. У рядку Подпись слід вказати потрібне ім'я поля. При перегляді результату запиту можна побачити нову назву підсумкового поля.

Запит 9. Отримати середнє значення виплат, зроблених кожному співробітнику протягом 2001 року. У відповіді вивести поля ПІБ, СЕРЕДНЯ ВИПЛАТА ЗА 2001 РІК.

Бланк запиту зображений на рис. 65.

Поле	Код	ПІБ	Дата виплати	Сума
Имя таблицы	Співробітники	Співробітники	Виплати	Виплати
Групповая операция	Группировка	Группировка	Условие	Avg
Вывод на экран		☑		☑
Условие отбора			Like “*01”	
Или				

Рис.65. Бланк запиту 9

Зауваження. Якщо при виконанні групових операцій на поле встановлюється певна умова (в розглянутому прикладі таке поле – Дата виплати), в рядку бланку Групповая операция *для цього поля слід встановити значення* Условие. *Виведення на екран для такого поля забороняється.*

Доцільно проекспериментувати з розглянутим запитом, змінивши у полі ДАТА ВИПЛАТИ значення Условие на Группировка і встановивши для цього поля виведення на екран.

При створенні запитів на обчислення підсумкових значень важливо стежити за поміщенням полів у бланк запиту. Навіть присутність у бланку певного поля без його виведення на екран може суттєво впливати на результат запиту.

Запит 10. Обрахувати загальний фонд заробітної плати та розмір середньої виплати за 2001 рік. В результаті вивести поля ЗАГАЛЬНА СУМА ВИПЛАТ, СЕРЕДНЯ ВИПЛАТА.

Вміст бланку запиту зображено на рис.66. Результат запиту міститиме один запис з двох полів, назви яких вказані у властивостях підсумкових полів.

Поле	Дата виплати	Сума	Сума
Имя таблицы	Виплати	Виплати	Виплати
Групповая операция	Условие	Sum	Avg
Вывод на экран		☑	☑
Условие отбора	Like “*01”		
Или			

Рис.66. Бланк запиту 10

Слід відмітити, що у бланку запиту немає поля, що ідентифікує кожний окремий запис (КОД). При його наявності (навіть без виведення на екран) підсумкові значення обчислювалися б не для всієї таблиці, а для кожного співробітника окремо.

Запити з параметрами

СУБД ACCESS дозволяє формувати умови відбору даних в процесі виконання запиту. Така “інтерактивність” запитів реалізується завдяки використанню параметрів. Щоб створити такий запит, слід у бланку запиту в рядку Условие отбора вказати не константу певного типу, з якою будуть порівнюватись дані, а пояснювальний текст в квадратних дужках. Синтаксис цього тексту довільний, а його зміст повинен повідомляти користувача, яке значення для відбору даних він повинен ввести. При виконанні такого запиту на екран виводиться вікно з назвою "Введите значение параметра", в якому зображується пояснювальний текст і поле для введення значення параметра. Після введення значення можна побачити результат запиту.

Запит 11. Сформувати список співробітників (ПІБ і СТАЖ), стаж роботи яких не менше вказаного користувачем. Бланк запиту зображений на рис. 67.

Поле	ПІБ	Стаж
Имя таблицы	Співробітники	Співробітники
Вывод на экран	☑	☑
Условие отбора		>=[Введіть мінімальний стаж]
Или		

Рис.67. Бланк запиту 11

Можна вказати у бланку запиту декілька параметрів у різних полях. Тоді при виконанні запиту послідовно з’являтимуться діалогові вікна для вказування всіх вказаних параметрів.

Запити на модифікацію таблиць

СУБД ACCESS дозволяє створювати три типи запитів, результатами яких є не нові таблиці, а зміни, внесені у вихідні таблиці бази даних. При цьому, на відміну від СУБД Paradox, ці зміни є незворотними. Отже, для попередження втрати цінних даних, при вивченні таких запитів доцільно створити у поточному файлі бази даних копії таблиць і проводити зміни цих копій. Для копіювання таблиці можна скористатися її контекстним меню (команда Копировать), а потім вставити вміст буфера в поточне вікно. В діалоговому вікні слід вказати нове ім’я таблиці-копії і підтвердити її збереження в поточному файлі бази даних.

Бланк запиту для кожного з типів запитів на зміну таблиць (оновлення, вилучення, додавання) має свої особливості. Головною відмінністю бланків цих запитів від бланка запиту на вибірку є відсутність рядка "Вывод на экран". Після запуску запиту на виконання ACCESS попереджає у діалоговому вікні про кількість записів, які будуть змінені, вилучені або додані у таблицю. Натиснення кнопки “Да” у цьому вікні призводить до виконання запиту і внесення змін у таблицю. Після виконання запитів на зміну їх результат продивляються, відкривши відповідну таблицю на закладці Таблицы вікна бази даних.

Запити на оновлення даних

Запити на оновлення дозволяють змінювати значення окремих полів вихідної таблиці. Для полів, що підлягають зміні, можуть бути вказані певні умови.

При створенні запиту на оновлення вікно конструктора запитів викликається на екран так, як у випадку запитів на вибірку. Далі в режимі конструктора слід вибрати з меню команду Запрос – Обновление. В результаті бланк запиту частково змінить вигляд. Рядки бланку на оновлення мають такі значення:

– Поле: вказується ім’я поля таблиці, що братиме участь у запиті. Це може бути поле, дані в якому підлягають зміні, або поле, дані з якого повинні задовольняти певну умову для зміни даних з інших полів;
– Имя таблицы: автоматично встановлюється у клітинку під назвою поля;
– Обновление: вказується нове значення, яке потрібно розмістити у вказаному полі. Це може бути константа або вираз, утворений за правилами, розглянутими для запитів на обчислення;
– Условие отбора: в цій клітинці записується умова для відбору даних у полях, що підлягають зміні, або інших полях, що впливають на зміни даних.
– Или: записується складова умови “або”, що стосується даних в одному полі або різних полях.

Запит 12. Замінити у таблиці ВИПЛАТИ всі входження помилково введеної дати 1 травня 2002 року на 11 травня 2002 року.

Вміст бланку запиту зображено на рис. 68.

Поле	Дата виплати
Имя таблицы	Виплати
Обновление	#11.05.02#
Условие отбора	#1.05.02#
Или	

Рис.68. Бланк запиту 12

Запит 13. Збільшити стаж всіх співробітників на один рік.

Даний запит повинен змінити всі записи таблиці. Отже, він не потребує вказування умови відбору даних. Вміст бланку запиту зображено на рис.69. Дані в полі СТАЖ оновлюються на основі формули, що містить назву поля даної таблиці.

Поле	Стаж
Имя таблицы	Співробітники
Обновление	[Стаж]+1
Условие отбора	
Или	

Рис.69. Бланк запиту 13

Запит 14. Модифікуємо попередній запит. Нехай потрібно збільшити стаж не всім співробітникам, а лише тим, хто працює на посаді інженера і чий стаж більше одного року. В цьому випадку у бланку запиту з'явиться нове поле ПОСАДА і умови для двох полів, пов'язані зв'язкою "і". Бланк запиту зображений на рис.70.

Поле	Стаж	Посада
Имя таблицы	Співробітники	Співробітники
Обновление	[Стаж]+1	
Условие отбора	>1	"інженер"
Или		

Рис.70. Бланк запиту 14

Запити на вилучення даних з таблиць

Такі запити дозволяють вилучати з таблиці записи, у яких значення певних полів відповідають вказаним умовам.

При створенні запиту на вилучення в режимі конструктора слід вибрати з меню команду Запрос – Удаление. Рядки бланку запиту на вилучення мають такі значення:

– Поле: вказується ім'я таблиці, з якої мають вилучатися записи (у форматі Ім'я_таблиці.*), а також імена полів, для яких встановлюються умови вилучення даних;

– Имя таблицы: автоматично встановлюється у клітинку під назвою поля;
– Удаление: автоматично встановлюється значення Из (у стовпці, де у клітинці Поле вказано ім'я таблиці) або Условие (в інших стовпцях);
– Условие отбора: в цій клітинці записується умова для відбору даних у полях записів, що підлягають вилученню;
– Или: записується складова умови "або", що стосується даних в одному полі або різних полях.

Запит 15. Вилучити з таблиці СПІВРОБІТНИКИ всі відомості про інженерів, стаж роботи яких більше 20 років. Вміст бланку запиту зображено на рис.71.

Поле	Співробітники.*	Посада	Стаж
Имя таблицы	Співробітники	Співробітники	Співробітники
Удаление	Из	Условие	Условие
Условие отбора		"інженер"	>20
Или			

Рис.71. Бланк запиту 15

Запити на додавання даних до таблиць

Запити на додавання дозволяють поповнювати таблицю даними з іншої таблиці або введеними безпосередньо у бланк запиту. Створення таких запитів дещо відрізняється від створення інших запитів на зміну даних. Перед розробкою запиту на додавання слід чітко визначити джерело нових даних для таблиці, враховуючи при цьому, що в нових записах обов'язково повинні бути коректно заповнені ключові поля. До головної таблиці бази даних нові записи, як правило, додаються з інших таблиць, що мають аналогічну структуру. Ключ таблиці-джерела даних повинен містити значення, відсутні у ключі таблиці, що поповнюється. Підлеглі таблиці бази даних можуть поповнюватись даними, введеними у бланку запиту. При цьому дані для ключа беруться із відповідного поля головної таблиці.

На відміну від інших запитів на зміну даних, в процесі створення запиту на додавання у вікні Добавление таблицы вказується не та таблиця, до якої робиться запит (в даному випадку – додаються записи), а та, з якої будуть братися дані (цілі записи або окремі поля). Далі в режимі конструктора слід вибрати з меню команду Запрос – Добавление і у діалоговому вікні, що з'являється

при цьому, вибрати із списку ім’я таблиці, до якої створюється запит. Рядки бланку запиту на додавання мають такі значення:

– Поле: вказуються імена полів таблиці, з якої беруться дані, а також вирази, значення яких записуються у поля доданих записів;
– Имя таблицы: ім’я таблиці, з якої додаються дані, автоматично встановлюється у клітинку під назвою відповідного поля;
– Добавление: вказуються імена полів таблиці, до якої додаються дані; ці імена повинні знаходитись під іменами відповідних полів таблиці-джерела або відповідних виразів;
– Условие отбора: в цій клітинці записується умова для відбору даних у полях таблиці, яка є джерелом даних;
– Или: записується складова умови “або”, що стосується даних в одному полі або різних полях.

Запит 16. Нехай у базі даних є додаткова таблиця АНКЕТА, в якій зберігаються відомості про бажаючих вступити на роботу до даної організації. Таблиця містить такі поля: КОД, ПІБ, ДАТА НАРОДЖЕННЯ, ХАРАКТЕРИСТИКА, ЗАРАХОВАНИЙ. В останнє поле занесені результати співбесіди: “+”(позитивний результат) і “–“(негативний результат). Треба додати до таблиці СПІВРОБІТНИКИ дані про тих осіб з таблиці АНКЕТА, для яких у полі ЗАРАХОВАНИЙ вказано “+”.

При створенні цього запиту у вікні Добавление таблицы слід вказати таблицю АНКЕТА, у конструкторі запитів вибрати команду меню Запрос-Добавление і у діалоговому вікні, що з’явиться при цьому, вибрати таблицю, яка поповнюється – СПІВРОБІТНИКИ. Потім заповнити бланк запиту, як зображено на рис.72.

Поле	Код	ПІБ	Дата народження	Зарахований
Имя таблицы	Анкета	Анкета	Анкета	Анкета
Добавление	Код	ПІБ	Дата народження	
Условие отбора				“+”
Или				

Рис.72. Бланк запиту 16

Запит 17. Внести у таблицю ВИПЛАТИ відомості про сплату кожному співробітнику премії розміром 100 гривень 22 жовтня 2002 року.

Для розв’язування цієї задачі слід створити запит на додавання, в результаті якого таблиця ВИПЛАТИ має поповнитись новими записами, утвореними наступним чином. Дані для обов’язкового поля

КОД повинні привноситись із таблиці СПІВРОБІТНИКИ, дані для полів ДАТА ВИПЛАТИ і СУМА можуть бути записані у першому рядку бланка запиту. При внесенні цих даних ACCESS автоматично дописує перед ними слово Выражение:. Слід нагадати, що при створенні цього запиту у вікні Добавление таблицы слід вказати таблицю СПІВРОБІТНИКИ. Бланк запиту зображений на рис 73.

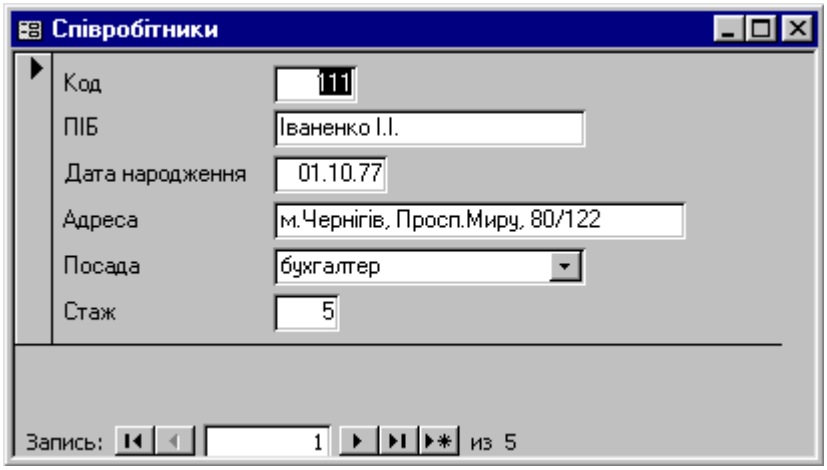
Поле	Код	Выражение: #22.10.02#	Выражение:100
Имя таблицы	Співробітники		
Добавление	Код	Дата виплати	Сума
Условие отбора			
Или			

Рис.73. Бланк запиту 17

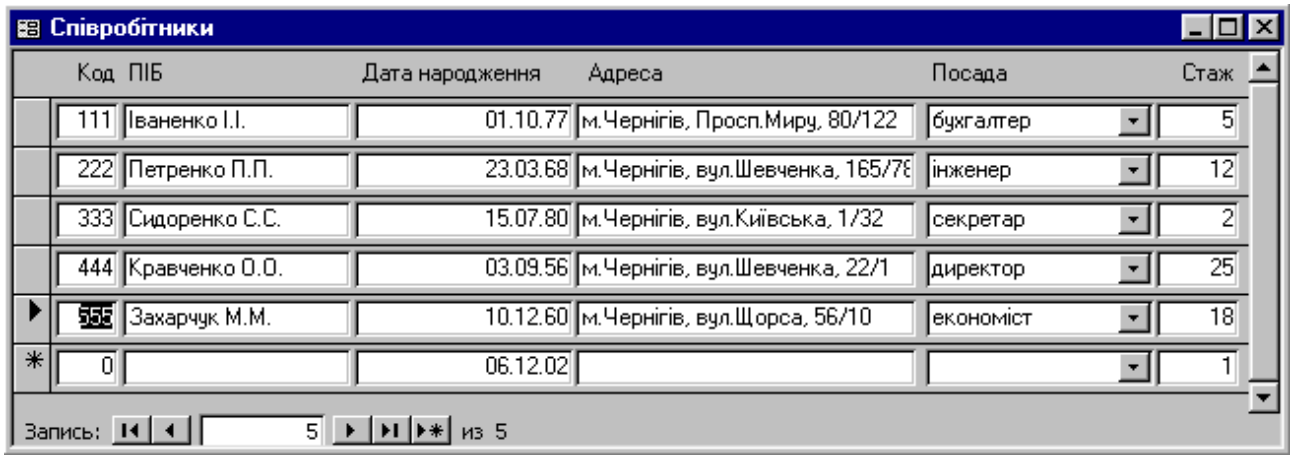
5.3.5. ФОРМИ І ЗВІТИ У СУБД ACCESS

Форма у СУБД ACCESS – це об′єкт бази даних, призначений для введення даних у таблиці, їх редагування і відображення даних з одної або кількох таблиць чи запитів у зручному для користувача вигляді. Форма може містити всі поля таблиці або деякі з полів, дозволяти або забороняти редагування даних, надавати користувачу допоміжні засоби роботи з даними у вигляді списків, що розкриваються, перемикачів тощо. При розв’язуванні реальних задач за допомогою СУБД кінцевий користувач не має доступу до таблиць бази даних, а працює тільки з формами, створеними до таблиць або запитів.

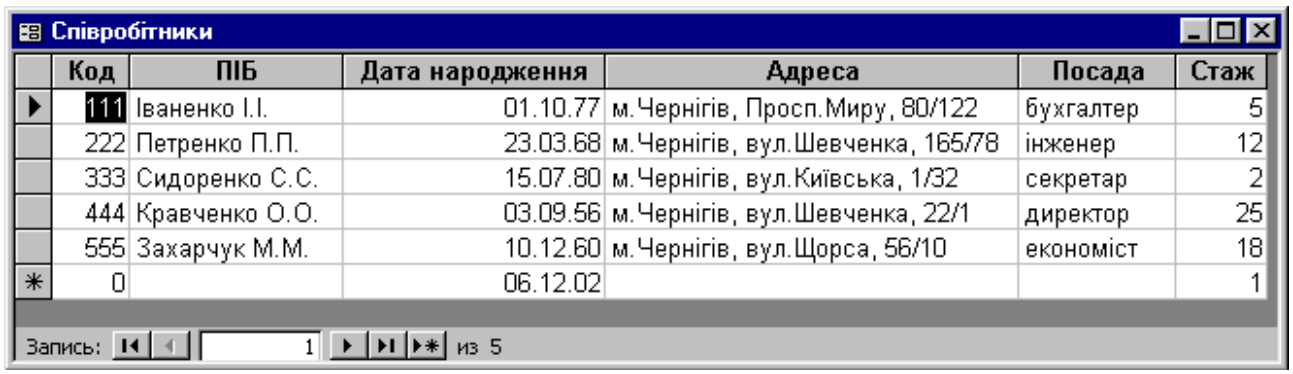
Робота з формами у СУБД ACCESS відбувається на закладці Формы вікна бази даних. Як і розглянуті вище об’єкти бази даних, форми можна створювати в режимі конструктора. Проте даний процес є досить трудомістким і потребує розвинених навиків, тому при розробці навчальних форм доцільніше навчитись створювати більш прості форми з використанням засобів автоматизації, що надаються системою ACCESS, а основи роботи з конструктором розглядати на прикладі готових форм. Повністю автоматизується процес створення так званих автоформ, частково – процес створення форм за допомогою майстра форм. Автоформа – це стандартизована форма, що створюється для одної таблиці чи запиту, містить всі наявні поля і має один з трьох видів (рис. 74).



а)автоформа "в стовпець";



б)стрічкова автоформа;



в)таблична автоформа.

Рис. 74. Види автоформ у СУБД ACCESS

В столбец (рис.74а): відображає всі поля одного запису в стовпчик (аналогічно стандартній формі `Paradox`). Така форма є зручною для введення і редагування даних;

Ленточная (рис.74б): відображає одночасно групу записів. Таку форму зручно використовувати для виведення даних;

Табличная (рис.74в): має такий самий вигляд, як вихідна таблиця.

Розглянемо процес створення автоформи в стовпець для таблиці СПІВРОБІТНИКИ.

Для створення автоформи слід на закладці Формы вікна бази даних натиснути кнопку Создать. Далі у вікні Новая форма (рис.75) у списку “Выберите в качестве источника данных таблицу или запрос“ вибрати таблицю чи запит, до якого створюється форма; вибрати тип автоформи і натиснути Ok. В результаті створюється форма, готова для роботи – відображення і редагування даних.

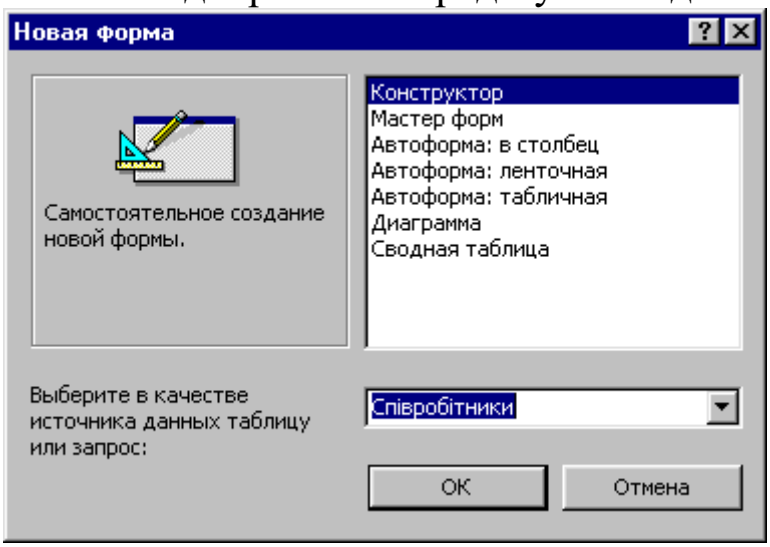


Рис.75.Вікно вибору способу створення нової форми

За допомогою Майстра форм можна створити форму для відображення даних з одної або кількох таблиць (запитів) в режимі діалогу з користувачем. Для запуску Майстра слід на закладці Формы вікна бази даних клацнути на значку Создание формы с помощью мастера. В процесі роботи з майстром виконують такі дії:

– на першому етапі вибирають таблиці і поля, що увійдуть у форму;
– на другому етапі вибирають зовнішній вигляд форми;
– на третьому етапі вибирають дизайн форми;
– на останньому етапі зберігають форму під потрібним ім’ям.

Для перегляду і редагування структури готової форми її відкривають в режимі конструктора.

Розглянемо процес створення зв’язаної форми для таблиць СПІВРОБІТНИКИ і ВИПЛАТИ.

Після запуску майстра у його першому вікні слід послідовно вибрати таблиці, для яких створюється форма (поле зі списком Таблицы и запросы) вибрати необхідні поля перенесенням їх за допомогою кнопок >, >> із списку Доступные поля у список Выбранные поля (рис.76). Кнопка > дозволяє перенести лише

виділене поле, кнопка >> – всі поля. Для повернення назад одного або всіх полів застосовуються кнопки <, <<.

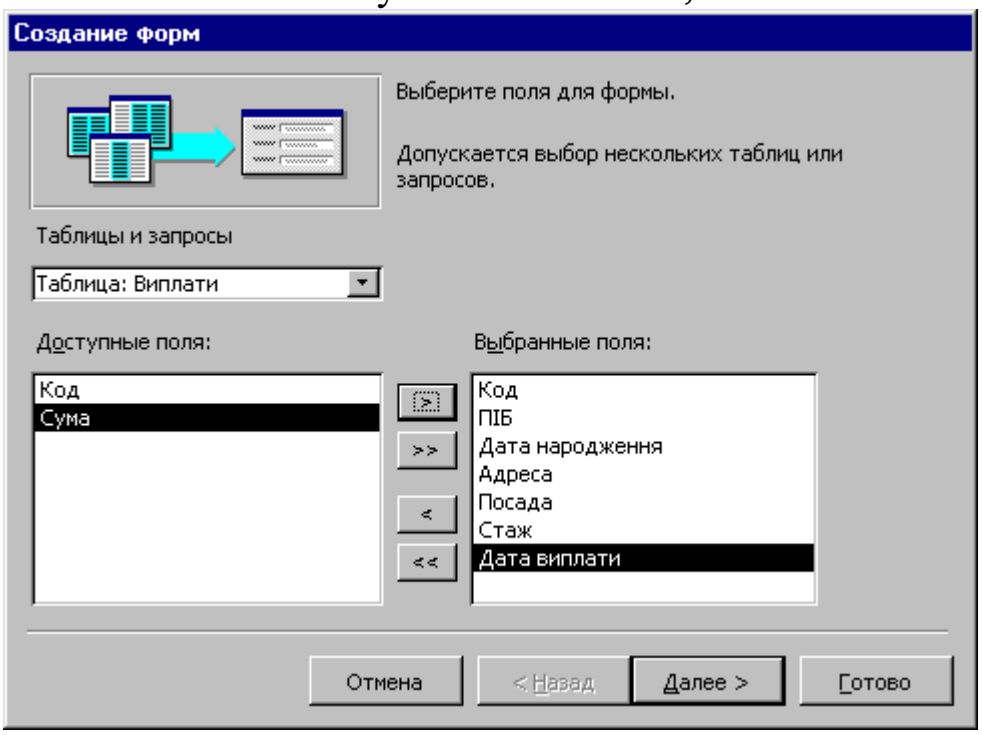


Рис.76. Вибір таблиць і їх полів для виведення у формі

Таким чином перенесемо у форму всі поля таблиці СПІВРОБІТНИКИ і поля ДАТА ВИПЛАТИ і СУМА таблиці ВИПЛАТИ.

У другому вікні майстра у полі Выберите вид представления данных слід вибрати варіант - Співробітники (рис. 77). Поля вибору Подчиненные формы і Связанные формы дозволяють встановити бажане взаємне розташування даних з двох таблиць.

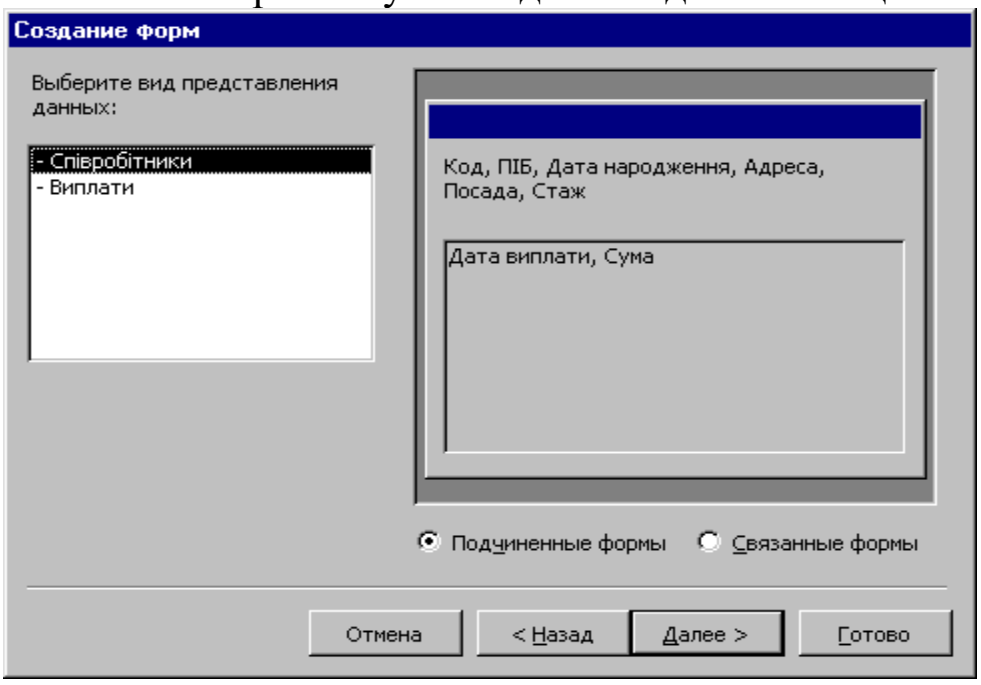


Рис.77. Вибір типу подання даних у зв'язаній формі

У третьому і четвертому вікнах майстра слід вибрати зовнішній вигляд підлеглої форми (пропонуються варіанти: Ленточный або Табличный) і один із наявних дизайнів форми (наприклад, стандартний).

У п’ятому, останньому вікні пропонуються назви зв’язаної форми і підлеглої форми, яка зберігається у вигляді окремого об’єкта і може відкриватись і опрацьовуватись без головної форми (Рис.78)

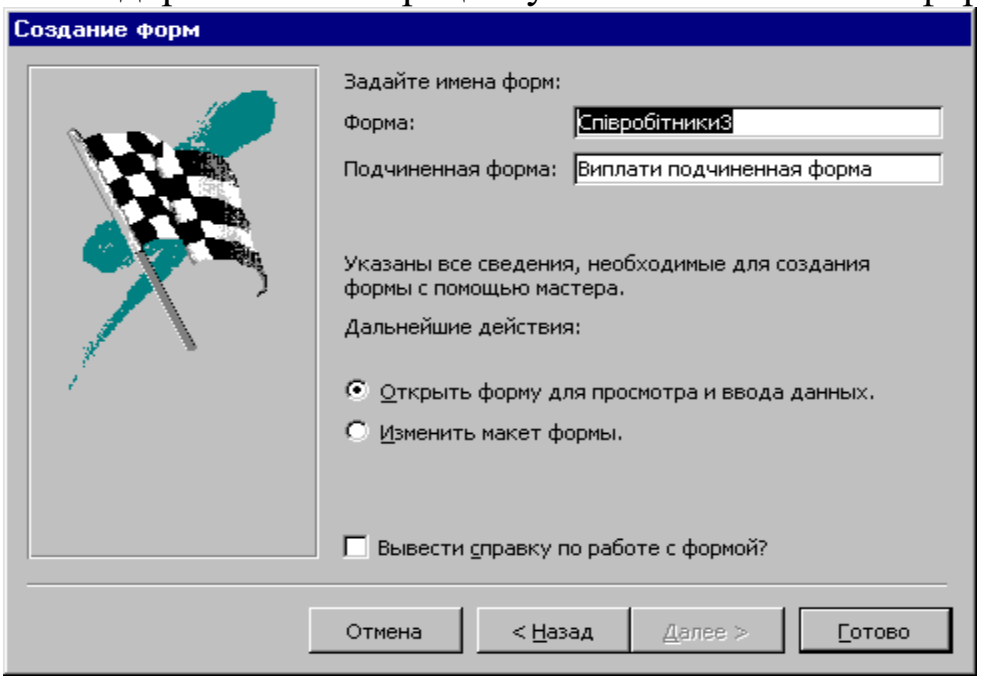


Рис.78. Вибір назв створених форм

Створена зв’язана форма зображена на рис.79.

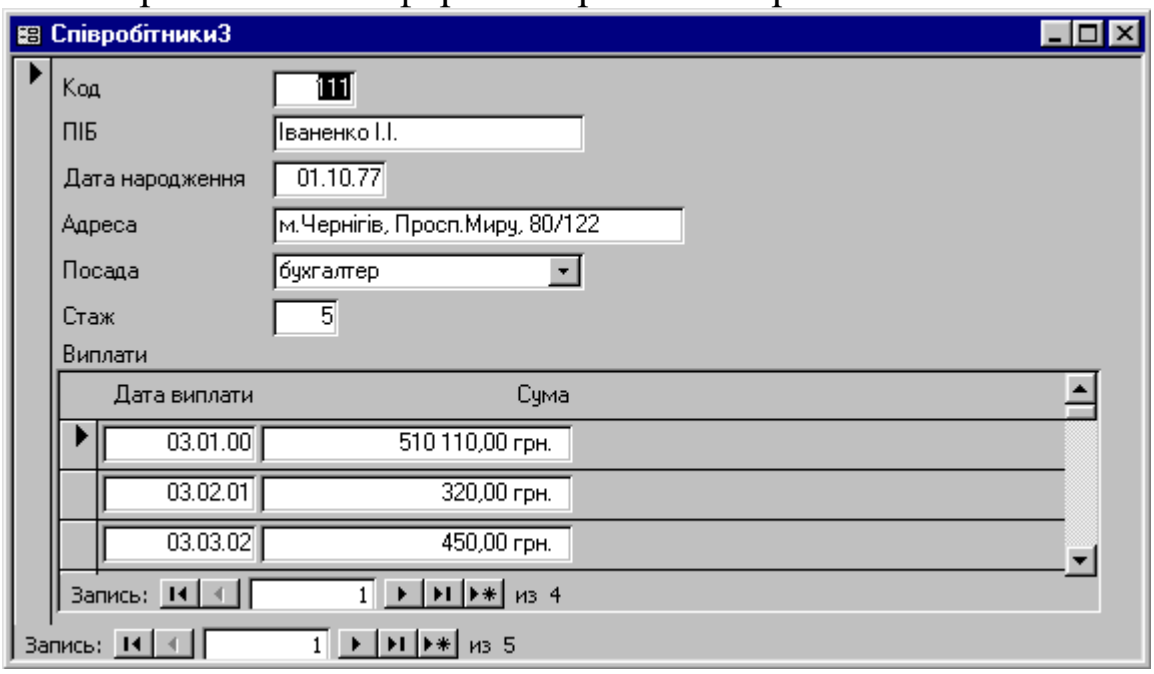


Рис.79. Зв’язана форма для таблиць Співробітники і Виплати

Структура форми

Для перегляду та редагування структури форми її слід відкрити в режимі конструктора. Основи роботи в цьому режимі розглянемо на прикладі створеної автоформи в стовпець для таблиці СПІВРОБІТНИКИ (рис.80).

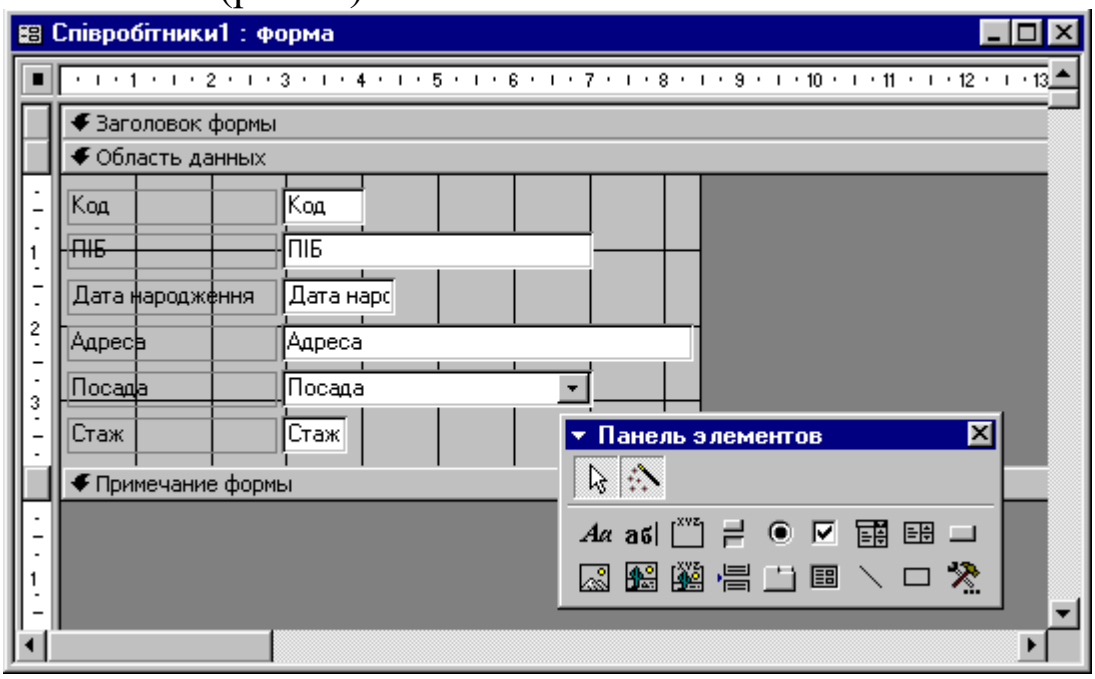


Рис.80. Структура форми у режимі Конструктора

Як можна бачити на рис.80, форма має три основні розділи: область заголовку, область даних та область примітки. Лінії, що розділяють ці розділи, можна перетягувати по вертикалі за допомогою миші –це дозволяє змінювати їх розміри за бажанням користувача.

Розділи заголовку і примітки виконують декоративну роль, їх вміст не пов’язаний з вмістом таблиці чи запиту, на якому базується форма. Змістовне значення має розділ даних – в ньому розміщуються елементи управління, що відповідають за введення та відображення даних. Для автоматизації введення даних розробник форми має можливість розміщувати в цьому розділі додаткові елементи управління, притаманні Windows-додаткам – перемикачі, списки тощо. Елементи управління форми подаються на спеціальній панелі елементів, яка виводиться на екран за допомогою команди меню Вид-Панель элементов або при натисненні відповідної кнопки на стандартній панелі інструментів. Щоб вибрати потрібний елемент управління, слід клацнути на ньому лівою кнопкою миші на панелі елементів, а потім клацнути в полі форми

там, де повинен знаходитись елемент. Разом з елементом в поле форми вставляється його приєднаний напис. За замовчуванням ці написи стандартні, наприклад, для перемикачів це Переключатель1, Переключатель2 тощо. Редагуванням властивостей елемента (за допомогою його контекстного меню) можна надати йому більш змістовний підпис.

Основними елементами оформлення форми є текстові написи і малюнки. Для створення у формі текстових написів призначені два елементи управління – Надпись і Поле. Написом може бути довільний введений користувачем текст. Елемент Поле відрізняється тим, що в ньому відображується вміст одного з полів таблиці, на якій базується форма, отже, при переході між записами текст може змінюватись.

Для створення графічних елементів оформлення призначені елементи управління Рисунок, Свободная рамка объекта і Присоединенная рамка объекта. Малюнок вибирається з графічного файла і вставляється у форму. Елемент Вільна рамка об'єкта не обов'язково є малюнком, це може бути довільний об'єкт OLE. Елемент Приєднана рамка об'єкта також може служити оформленням форми, але його вміст береться не з призначеного файла, а безпосередньо з таблиці бази даних, якщо вона має поле об'єкта OLE. Отже, при переході між записами вміст цього елемента може змінюватись.

Оскільки форма – основний об'єкт бази даних, з яким безпосередньо працюють кінцеві користувачі інформаційної системи, до форм висуваються спеціальні вимоги щодо дизайну.

Вирівнювання всіх елементів управління форми здійснюється командою меню Формат–Выровнять. Для рівномірного розподілу елементів по полю форми використовуються команди меню Формат-Интервал по горизонтали, Формат–-Интервал по вертикали.

Можна змінювати розташування і розміри елементів управління вручну. Для цього слід виділити потрібний елемент і виконати перетягування маркерів, що з'являються при виділенні. Елементи перетягуються разом з приєднаними до них написами. Виключення складає перетягування маркера лівого верхнього кута, при якому можна відірвати приєднаний напис від елемента.

При розробці дизайну форми на екран може виводитись допоміжна сітка (команда меню Вид-Сетка). Для автоматичної прив'язки елементів до вузлів сітки слід виконати команду Формат–Привязать к сетке.

Звіти у СУБД ACCESS

Звіт у СУБД ACCESS – це об′єкт бази даних, призначений для групування, форматування, обчислення підсумкових значень та підготовки до друку даних з одної або кількох таблиць чи запитів.

Робота із звітами у СУБД ACCESS відбувається на закладці Отчеты вікна бази даних і в цілому аналогічна роботі з формами.

Існує можливість створення автозвітів двох типів (Ленточный, В столбец). Порядок дій при створенні автозвіту такий самий, як при створенні автоформи.

Стандартизовані звіти зручно створювати за допомогою майстра звітів.

Майстер звітів працює в шість етапів. При його роботі здійснюється вибір таблиць або запитів, на основі даних з яких створюється звіт, вибір полів, що відображуються у звіті, вибір полів, за якими групуються дані, вибір полів і методів сортування, вибір форми друкованого макету і стилю оформлення.

Структура готового звіту відрізняється від структури форми збільшеною кількістю розділів. Крім розділів заголовка, примітки і даних, звіт може містити розділи верхнього і нижнього колонтитулів. Якщо звіт займає більше одної сторінки, ці розділи необхідні для друку службової інформації, наприклад, номерів сторінок.

Редагування структури звіту виконується в режимі Конструктора за тими ж принципами, що і у випадку форм. елементи управління у структурі звіту виконують функції елементів оформлення, оскільки друкований звіт, на відміну від форм, не є інтерактивним об’єктом. Розміщення елементів управління звіту виконується за допомогою панелі елементів (команда меню Вид-Панель элементов), склад якої практично не відрізняється від панелі елементів форми. Важливою особливістю звітів є наявність засобу для вставки в область верхнього або нижнього колонтитула поточного номера сторінки і повної кількості сторінок. Ця операція виконується за допомогою діалогового вікна Номера страниц (команда меню Вставка–Номера страниц).

ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Дайте загальну характеристику СУБД Paradox.
2. Яке призначення мають функціональні клавіші при роботі з СУБД Paradox?

3. Як встановити потрібний робочий каталог у СУБД Paradox?
4. Які типи полів доступні у СУБД Paradox?
5. Як створюється структура таблиці у СУБД Paradox?
6. Як відбувається внесення і редагування даних у таблицях?
7. Як забезпечується контроль даних у СУБД Paradox?
8. Як відбувається сортування даних у СУБД Paradox?
9. Що називається запитом? За якою схемою створюються запити у СУБД Paradox?
10. Як задаються умови відбору даних при створенні запитів на вибірку у СУБД Paradox?
11. Як створити запит до кількох таблиць?
12. Як виконуються обчислення у запитах СУБД Paradox?
13. Які особливості мають запити на модифікацію таблиць?
14. Як створюються запити на вилучення даних у СУБД Paradox?
15. Як створюються запити на зміну даних у СУБД Paradox?
16. Як створюються запити на додавання даних у СУБД Paradox?
17. Що таке форма? Які відмінності мають зображення даних у табличному вигляді і у вигляді форми?
18. Як створити однотабличну форму у СУБД Paradox?
19. Як створити зв’язану форму у СУБД Paradox?
20. Що таке звіт? Яки типи звітів дозволяє створювати СУБД Paradox?
21. Як створити однотабличний звіт у СУБД Paradox?
22. Як створити зв’язаний звіт у СУБД Paradox?
23. Дайте загальну характеристику СУБД ACCESS.
24. Опишіть роботу з конструктором таблиць. Які типи даних доступні у СУБД ACCESS?
25. Що таке схема даних? Як вона створюється?
26. Які типи запитів можна створювати у СУБД ACCESS? Опишіть роботу з конструктором запитів на вибірку.
27. Як створюються запити з параметрами у СУБД ACCESS?
28. Як виконуються обчислення у запитах СУБД ACCESS?
29. Як створюються запити на вилучення даних у СУБД ACCESS?
30. Як створюються запити на оновлення даних у СУБД ACCESS?
31. Як створюються запити на додавання даних у СУБД ACCESS?
32. Як створити автоформу у СУБД ACCESS?
33. Як створити зв’язану форму у СУБД ACCESS?
34. Як створити автозвіт у СУБД ACCESS?
35. Як створити зв’язаний звіт у СУБД ACCESS?

5.3.6. Варіанти завдань для самостійного виконання

ВАРІАНТ 1

1. Створити базу даних ГЕОГРАФІЯ, що містить таблиці з назвами КОНТИНЕНТИ і КРАЇНИ. Таблиця КОНТИНЕНТИ містить інформацію про континенти Землі, таблиця КРАЇНИ – про країни цих континентів. Таблиці мають таку структуру:

Таблиця КОНТИНЕНТИ	**Таблиця КРАЇНИ**
Континент	Континент
Площа_континенту	Країна
Населення_континенту	Площа
	Населення
	Столиця
	Населення_столиці

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв’язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці КОНТИНЕНТИ вибрати континенти, населення яких менше 1 млрд. У відповідь включити всі поля;
   b) з таблиці КРАЇНИ вибрати країни, площа яких перевищує 100 тис.кв.км., а населення столиці перевищує 1 млн. У відповідь включити поля Країна, Площа, Столиця;
   c) використовуючи дані з таблиці КРАЇНИ, підрахувати сумарну площу всіх країн Європи. Результат помістити в поле Сума_площ;
   d) знайти всі країни, площа яких знаходиться в межах від 150 тис.кв.км. до 250 тис.кв.км. У відповідь включити поля Континент і Площа_континенту з таблиці КОНТИНЕНТИ та поля Країна, Площа, Столиця з таблиці КРАЇНИ;
   e) вилучити з таблиці КРАЇНИ відомості про країни, населення столиць яких менше 3 мільйонів.
6. Створити просту форму для додавання даних про нові континенти до таблиці КОНТИНЕНТИ. В форму включити всі поля вказаної таблиці.
7. Створити зв’язану форму, що містить дані про назву, площу і населення континенту з таблиці КОНТИНЕНТ і назву країни, а також її площу, населення і відомості про столицю з таблиці КРАЇНИ. Забезпечити неможливість модифікації значень полів

таблиці КОНТИНЕНТ на цій формі. Створити обчислювальне поле для визначення щільності населення кожної країни.

8. Використовуючи запит, створити зв'язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 2

1. Створити базу даних СТУДЕНТИ, що містить дві таблиці з назвами ГРУПА і СТИПЕНДІЯ. Таблиця ГРУПА містить інформацію про студентів групи, таблиця STIPEN – про суму виплат стипендії, що отримують ці студенти. В полі ПІБ розміщені прізвище, ім'я, по батькові студента. Таблиці мають таку структуру:

Таблиця ГРУПА	**Таблиця СТИПЕНДІЯ**
Номер_заліковки	Номер_заліковки
ПІБ	Дата_виплати
Дата_народження	Сума
Вага	
Зріст	
Середній_бал	

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв'язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці ГРУПА вибрати студентів, зріст яких більше 170 см. У відповідь включити всі поля;
   b) з таблиці ГРУПА вибрати студентів, зріст яких перевищує 165 см., та народилися після 1.01.1985. У відповідь включити поля ПІБ, Дата_народження, Вага, Зріст;
   c) використовуючи дані з таблиці СТИПЕНДІЯ, підрахувати кількість виплат стипендії всім студентам за минулий рік. Результат помістити в поле Кількість_виплат;
   d) знайти всіх студентів, яким виплачувалась стипендія з 1.01 по 31.05 поточного року. У відповідь включити поля Номер_заліковки і ПІБ з таблиці ГРУПА та поля Дата_виплати і Сума з таблиці СТИПЕНДІЯ;
   e) нарахувати всім студентам стипендію визначеного розміру, датуючи її поточним днем.

6. Створити просту форму для додавання даних про нових студентів до таблиці ГРУПА. В форму включити всі поля вказаної таблиці.
7. Створити зв'язану форму, що містить дані про номер залікової книжки, ім'я, дату народження і середній бал студента з таблиці ГРУПА та дату виплати і суму стипендії з таблиці СТИПЕНДІЯ. Забезпечити неможливість модифікації значень полів таблиці ГРУПА на цій формі. Створити обчислювальне поле для визначення податку на кожну виплачену суму (20%).
8. Використовуючи запит, створити зв'язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 3

1. Створити базу даних АКЦІОНЕРИ, що містить дві таблиці з назвами ВЛАСНИКИ і АКЦІЇ. Таблиця ВЛАСНИКИ містить інформацію про акціонерів деякого акціонерного товариства, таблиця АКЦІЇ – про кількість акцій, що є у цих акціонерів. В полі ПІБ розміщені Прізвище, Ім'я, По батькові акціонера, в полі Документ – тип документу (паспорт, військовий квиток тощо), Реквізити вказують коли і ким виданий документ, поле Код_акцій містить інформацію про номер випуску акцій. Таблиці мають таку структуру:

Таблиця ВЛАСНИКИ	**Таблиця АКЦІЇ**
Код_рахунку	Код_рахунку
ПІБ	Код_акцій
Адреса	Кількість
Документ	
Реквізити	
Дата_реєстрації	

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв'язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці ВЛАСНИКИ вибрати акціонерів, що зареєстровані після 1.07 минулого року. У відповідь включити всі поля;
   b) з таблиці ВЛАСНИКИ вибрати акціонерів, у яких код рахунку не менше 500, а дата реєстрації – не раніше 1.01 поточного року. У відповідь включити поля Код_рахунку, ПІБ, Адреса, Документ;

c) використовуючи дані з таблиці АКЦІЇ, підрахувати загальну кількість акцій другого коду випуску у всіх акціонерів. Результат помістити в поле Кількість_акцій-2;
d) знайти всіх акціонерів, які володіють акціями 2 та 3 випусків. У відповідь включити поля Код_рахунку і ПІБ з таблиці ВЛАСНИКИ та поля Код_акцій і Кількість з таблиці АКЦІЇ;
e) подвоїти всім акціонерам кількість акцій третього коду випуску.

6. Створити просту форму для додавання даних про нових акціонерів до таблиці ВЛАСНИКИ. У форму включити всі поля вказаної таблиці.
7. Створити зв'язану форму, що містить дані про код рахунку, ім'я, адресу і дату реєстрації акціонера з таблиці ВЛАСНИКИ та код акцій і їх кількість з таблиці КРАЇНИ. Забезпечити неможливість модифікації значень полів таблиці ВЛАСНИКИ на цій формі. Створити обчислювальне поле для визначення вартості акцій кожного коду випуску. Вважати вартість кожної акції рівною 1,5 грн.
8. Використовуючи запит, створити зв'язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 4

1. Створити базу даних СЕСІЯ, що містить дві таблиці з назвами СТУДЕНТИ і ОЦІНКИ. Таблиця СТУДЕНТИ містить інформацію про студентів факультету, таблиця ОЦІНКИ – про оцінки, одержані ними на сесії. В полі ПІБ розміщені прізвище, ім'я, по батькові студента, в полі Спеціальність вказується скорочена назва спеціальності (наприклад, МІ, ФМ тощо), поле Контракт має логічний тип. У полі Предмет пишеться назва предмета, за яким складався екзамен. Таблиці мають таку структуру:

Таблиця СТУДЕНТИ	**Таблиця ОЦІНКИ**
Номер_заліковки	Номер_заліковки
ПІБ	Предмет
Група	Оцінка
Спеціальність	
Дата народження	
Контракт	

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв'язок між таблицями.

5. Виконати наступні запити:
   a) з таблиці СТУДЕНТИ вибрати студентів, номер групи яких більше 20. У відповідь включити всі поля;
   b) з таблиці СТУДЕНТИ вибрати студентів, які навчаються за контрактом, та народилися після 1.09.1983. У відповідь включити поля ПІБ, Група, Спеціальність, Дата_народження;
   c) використовуючи дані з таблиці ОЦІНКИ, підрахувати кількість одержаних всіма студентами п'ятірок. Результат помістити в поле Кількість_п'ятірок;
   d) знайти всіх студентів 11-19 груп, які одержали четвірки. У відповідь включити поля ПІБ і Група з таблиці СТУДЕНТИ та поля Предмет і Оцінка з таблиці ОЦІНКИ;
   e) всім студентам, що мають двійки з іноземної мови, виправити їх на трійки.
6. Створити просту форму для додавання даних про нових студентів до таблиці СТУДЕНТИ. В форму включити всі поля вказаної таблиці.
7. Створити зв'язану форму, що містить дані про номер залікової книжки, ім'я, номер групи та спеціальність студента з таблиці СТУДЕНТИ і назву предмета та оцінку з нього з таблиці ОЦІНКИ. Забезпечити неможливість модифікації значень полів таблиці СТУДЕНТИ на цій формі.
8. Використовуючи запит, створити зв'язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 5

1. Створити базу даних ТОРГІВЛЯ, що містить дві таблиці з назвами ТОВАР і ПРОДАЖ. Таблиця ТОВАР містить інформацію про товари, які завезені в деякий магазин, таблиця ПРОДАЖ – про продажу завезеного товару покупцям. В полі Виробник вказується фірма – виробник товару. Таблиці мають таку структуру:

Таблиця ТОВАР	**Таблиця ПРОДАЖ**
Код_товару	Код_товару
Назва	Дата_придбання
Виробник	Кількість
Дата_завезення	
Кількість_товару	
Ціна	

2. Вказати тип кожного поля, визначити ключові поля.

3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв’язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці ТОВАР вибрати товари, кількість яких в магазині більше 150. У відповідь включити всі поля;
   b) з таблиці ТОВАР вибрати товари, що завезені до 1.01 поточного року, та ціною не більше 200 гривень. У відповідь включити поля Назва, Дата_завезення, Кількість_товару, Ціна;
   c) використовуючи дані з таблиці ПРОДАЖ, підрахувати кількість товарів, придбаних у лютому поточного року. Результат помістити в поле Кількість_придбаних;
   d) знайти всю інформацію про товари, придбані з 1.07 минулого року по 30.06 поточного року. У відповідь включити поля Назва і Ціна з таблиці ТОВАР та поля Дата_придбання і Кількість з таблиці ПРОДАЖ;
   e) вилучити з таблиці ПРОДАЖ відомості про товари, придбані до 1.01 позаминулого року.
6. Створити просту форму для додавання даних про нові товари до таблиці ТОВАР. В форму включити всі поля вказаної таблиці.
7. Створити зв’язану форму, що містить дані про код товару, його назву, виробника і ціну з таблиці ТОВАР, а також дату придбання і кількість товару з таблиці ПРОДАЖ.. Забезпечити неможливість модифікації значень полів таблиці ТОВАР на цій формі.
8. Використовуючи запит, створити зв’язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 6

1. Створити базу даних ВІДЕОПРОКАТ, що містить дві таблиці з назвами ФІЛЬМИ і ПРОКАТ. Таблиця ФІЛЬМИ містить інформацію про відеокасети, що є в прокатному пункті, таблиця ПРОКАТ – про прокат цих касет. В полі Рік розміщений рік виробництва фільму, в полі Тривалість – тривалість фільму в хвилинах. Таблиці мають таку структуру:

Таблиця ФІЛЬМИ	Таблиця ПРОКАТ
Код_касети	Код_касети
Назва_фільму	Взято
Режисер	Повернуто
Студія	
Рік	
Тривалість	

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв’язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці ФІЛЬМИ вибрати фільми, тривалість яких перевищує 115 хвилин. У відповідь включити всі поля;
   b) з таблиці ФІЛЬМИ вибрати фільми, які були зняті в минулому році, а тривалість перевищує 110 хвилин. У відповідь включити поля Назва_фільму, Режисер, Рік, Тривалість;
   c) використовуючи дані з таблиці ПРОКАТ, знайти дату, коли в другій половині минулого року вперше деяка касета бралася в прокат. Результат помістити в поле Дата;
   d) знайти всю інформацію про фільми, касети з записами яких були взяті з 1.09 по 31.12 минулого року. У відповідь включити поля Назва_фільму і Режисер з таблиці ФІЛЬМИ та поля Взято і Повернуто з таблиці ПРОКАТ;
   e) вилучити з таблиці ФІЛЬМИ відомості про фільми режисера Т.Брасса.
6. Створити просту форму для додавання даних про нові фільми до таблиці ФІЛЬМИ. В форму включити всі поля вказаної таблиці.
7. Створити зв’язану форму, що містить дані про код касети, назву фільму, рік випуску і тривалість фільму з таблиці ФІЛЬМИ, а також дати видачі касети і її заявленого повернення з таблиці ПРОКАТ. Забезпечити неможливість модифікації значень полів таблиці ФІЛЬМИ на цій формі.
8. Використовуючи запит, створити зв’язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 7

1. Створити базу даних БІБЛІОТЕКА, що містить дві таблиці з назвами СТУДЕНТИ і КНИГИ. Таблиця СТУДЕНТИ містить інформацію про студентів навчального закладу, таблиця КНИГИ – про книжки, взяті ними в бібліотеці. В полі ПІБ розміщені прізвище, ім'я, по батькові студента, в полі Спеціальність вказується скорочена назва спеціальності (наприклад, МІ, ФМ тощо). Таблиці мають таку структуру:

Таблиця СТУДЕНТИ	**Таблиця КНИГИ**
Номер_заліковки	Номер_заліковки
ПІБ	Назва
Факультет	Дата_повернення
Група	
Спеціальність	
Дата народження	

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв'язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці СТУДЕНТИ вибрати студентів, які народились до 1.01.1984. У відповідь включити всі поля;
   b) з таблиці СТУДЕНТИ вибрати студентів п'ятого курсу фізико-математичного факультету. У відповідь включити поля Номер_заліковки, ПІБ, Факультет, Група;
   c) використовуючи дані з таблиці КНИГИ, знайти найпізнішу дату повернення будь-якої книги з алгебри. Результат помістити в поле Дата;
   d) знайти всю інформацію про студентів, які повинні повернути книги не пізніше, ніж через місяць, починаючи від сьогоднішньої дати. У відповідь включити поля ПІБ і Факультет з таблиці СТУДЕНТИ та поля Назва і Дата_повернення з таблиці КНИГИ;
   e) записати у формуляр кожного студента книгу з філософії з датою повернення 30.06 поточного року.
6. Створити просту форму для додавання даних про нових читачів до таблиці СТУДЕНТИ. В форму включити всі поля вказаної таблиці.
7. Створити зв'язану форму, що містить дані про номер залікової книжки, ім'я, факультет та номер групи з таблиці СТУДЕНТИ, а

також назву книги і заявлену дату її повернення з таблиці КНИГИ. Забезпечити неможливість модифікації значень полів таблиці СТУДЕНТИ на цій формі.

8. Використовуючи запит, створити зв'язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 8

1. Створити базу даних ЄВРОПА, що містить дві таблиці з назвами КРАЇНА і МІСТО. Таблиця КРАЇНА містить інформацію про європейські країни, таблиця МІСТО – про міста цих країн. Поле Столиця має логічний тип. Таблиці мають таку структуру:

Таблиця КРАЇНА	**Таблиця МІСТО**
Країна	Країна
Площа_країни	Назва_міста
Населення_країни	Площа
	Населення
	Рік_заснування
	Столиця

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв'язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці КРАЇНА вибрати країни, населення яких менше 35 млн. У відповідь включити всі поля;
   b) з таблиці МІСТО вибрати міста, які засновані до 1500 року, а населення перевищує 1 млн. У відповідь включити поля Країна, Назва_міста, Рік_заснування, Населення;
   c) використовуючи дані з таблиці МІСТО, підрахувати сумарне населення всіх міст України. Результат помістити в поле Все_населення;
   d) знайти всі міста, які були засновані з 1700 по 1950 роки. У відповідь включити поля Країна з таблиці КРАЇНА та поля Назва_міста, Рік_заснування, Столиця з таблиці МІСТО
   e) вилучити з таблиці МІСТО відомості про міста, населення яких не перевищує 250 тисяч.
6. Створити просту форму для додавання даних про нові країни до таблиці КРАЇНА. В форму включити всі поля вказаної таблиці.

7. Створити зв’язану форму, що містить дані про назву, площу і населення країни з таблиці КРАЇНА, а також назву міста і відомості про його її площу, населення і рік заснування з таблиці МІСТО. Забезпечити неможливість модифікації значень полів таблиці КРАЇНА на цій формі. Створити обчислювальне поле для визначення щільності населення кожного міста.
8. Використовуючи запит, створити зв’язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 9

1. Створити базу даних РЕСТОРАН, що містить дві таблиці з назвами ЗАКЛАД і СТРАВА. Таблиця ЗАКЛАД містить інформацію про ресторани та кафе міста, таблиця СТРАВА – про страви, що пропануються в цих закладах. В полі Назва розміщена назва ресторану, поле Тип містить тип страви (закуски, компоти тощо), Вага – вага страви в грамах, Приготування – час приготування в хвилинах. Таблиці мають таку структуру:

Таблиця ЗАКЛАД	**Таблиця СТРАВА**
Назва закладу	Назва закладу
Адреса	Тип
Директор	Назва_страви
Телефон	Вага
	Приготування
	Ціна

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв’язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці СТРАВА вибрати страви, вага яких менше 250 гр. У відповідь включити всі поля;
   b) з таблиці СТРАВА вибрати страви, приготування яких менше 20 хвилин, а ціна не менше 3 гривень. У відповідь включити поля Назва_страви, Вага, Приготування, Ціна;
   c) використовуючи дані з таблиці СТРАВА, знайти найменшу вартість будь-якої страви в ресторанах та кафе міста, час приготування якої менше 15 хвилин. Результат помістити в поле Мінімальна_вартість;

d) знайти всі страви, час приготування яких від 10 до 25 хвилин. У відповідь включити поля Назва закладу і Адреса з таблиці ЗАКЛАД та поля Назва_страви і Приготування з таблиці СТРАВА
e) вилучити з таблиці СТРАВА відомості про страви, час приготування яких більше 45 хвилин.

6. Створити просту форму для додавання даних про нові заклади громадського харчування до таблиці ЗАКЛАД. В форму включити всі поля вказаної таблиці.
7. Створити зв'язану форму, що містить дані про назву, адресу і телефон закладу з таблиці ЗАКЛАД і тип та назву страви, а також її ціну з таблиці СТРАВА. Забезпечити неможливість модифікації значень полів таблиці ЗАКЛАД на цій формі.
8. Використовуючи запит, створити зв'язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

ВАРІАНТ 10

1. Створити базу даних ШКОЛА, що містить дві таблиці з назвами ВЧИТЕЛЬ і КЛАС. Таблиця ВЧИТЕЛЬ містить інформацію про вчителів навчального закладу, таблиця КЛАС – про предмети, які вони викладають. В полі ПІБ розміщені прізвище, ім'я, по батькові вчителя, в полі Стаж – стаж роботи в роках, в полях Предмет і Клас – назва предмету та номер класу, в яких викладає цей вчитель. Таблиці мають таку структуру:

Таблиця ВЧИТЕЛЬ	**Таблиця КЛАС**
Код	Код
ПІБ	Предмет
Адреса	Клас
Дата_народження	
Стаж	
Категорія	

2. Вказати тип кожного поля, визначити ключові поля.
3. Для кожного поля вказати необхідні обмеження на дані.
4. Встановити зв'язок між таблицями.
5. Виконати наступні запити:
   a) з таблиці ВЧИТЕЛЬ вибрати вчителів, стаж роботи яких більше 5 років. У відповідь включити всі поля;

b) з таблиці ВЧИТЕЛЬ вибрати вчителів першої категорії, молодше 40 років. У відповідь включити поля ПІБ, Дата_народження, Стаж, Категорія;
c) використовуючи дані з таблиці ВЧИТЕЛЬ, підрахувати середній стаж вчителів першої категорії. Результат помістити в поле Середній_стаж;
d) знайти всіх вчителів, які викладають фізику в сьомих класах. У відповідь включити поля ПІБ і Стаж з таблиці ВЧИТЕЛЬ та поля Предмет і Клас з таблиці КЛАС
e) вчителям, що мають стаж роботи більше 15 років і другу категорію, змінити її на першу.

6. Створити просту форму для додавання даних про нових вчителів до таблиці ВЧИТЕЛЬ. В форму включити всі поля вказаної таблиці.
7. Створити зв’язану форму, що містить дані про ідентифікаційний код, ім’я, стаж і категорію кожного вчителя з таблиці ВЧИТЕЛЬ і назву предмету і відомості про клас з таблиці КЛАС. Забезпечити неможливість модифікації значень полів таблиці ВЧИТЕЛЬ на цій формі.
8. Використовуючи запит, створити зв’язаний звіт, що містить дані, вказані в завданні 7. Включити у звіт підсумкові поля.

СПИСОК ЛІТЕРАТУРИ

1. Биков В.Ю., Руденко В.Д. Системи управління інформаційними базами даних в освіті. – К.: ІЗМН, 1996.
2. Верлань А.Ф., Коваленко Ф.Е., Валеев Д.Г. Современное состояние и тенденции развития систем управления базами данных. – К., 1994. – 49 с.
3. Вольфенгаген В.Э. и др. Реляционные методы проектирования банков данных. – К.: Вища школа, 1979. – 192 с.
4. Грэй П. Логика, алгебра и базы данных. – М.: Машиностроение, 1989. – 368 с.
5. Жалдак М.І., Рамський Ю.С. Інформатика. – К.: Вища школа, 1991. – 319 с.
6. Змитрович А.И. Базы данных. – Минск: Университетское, 1991. – 271 с.
7. Информатика. Базовый курс / Симонович С.В. и др. –СПб.: Издательство «Питер», 1999. – 640 с.
8. Петухова Н.М. и др. Банки данных. – Л., 1978. – 66 с.
9. Полищук Ю.М., Хон В.Б. Теория автоматизированных банков информации. – М.: Высшая школа, 1989. – 184 с.
10. Праг, Керри Н., Ирвин, Мишель Р. Библия пользователя Access 97.: Пер. с англ. – К.: Диалектика, 1997. – 768 с.
11. Ревунков Г.И. и др. Базы и банки данных и знаний. – М.: Высшая школа, 1992. – 367 с.
12. Редько В.Н., Сергиенко И.В., Стукало А.С. Прикладные программные системы: архитектура, построение, развитие. – К.: Наукова думка, 1992. – 320 с.
13. Сигель Ч. Парадокс – это очень просто. – М.: БорАГ, 1993. – 399 с.
14. Тиори Т., Фрай Дж. Проектирование структур баз данных. – М.: Мир, 1985. Т.1. – 287 с.; Т.2. – 320 с.