# Вступ

Технічний прогрес є безупинним рухом. Розвиток людства не міг би проходити без втілення в їх життя новітніх методів і технологій. В останні десятиліття найбільш стрімко і революційно проходить розвиток новітніх комп’ютерних та інформаційних систем, який веде за собою глобальну автоматизацію. Підприємства, установи та інші об’єкти переходять на автоматизовану роботу, що значно полегшує працю простих людей та веде до зниження затрат на виробництво тих чи інших продуктів чи благ. Автоматизація торкнулась і баз даних працівників різних структур. Саме тому потреба у розробці автоматизованих баз даних є доволі великою.

В майбутньому проекті розглянемо роботу та побудову автоматизованої бази даних для продавця-консультанта магазину з продажу побутової техніки.

Метою даної дипломної роботи є розробка автоматизованої бази даних в середовищі Microsoft Visual FoxPro 7.0.

CУБД FoxPro (Microsoft) має виключно високі швидкісні характеристики. Набір команд і функцій відповідає всім сучасним вимогам до представлення і обробки даних. Надається можливість використовувати максимально зручний інтерфейс. Підтримуються різноманітні багаторівневі меню, робота з комп’ютерним маніпулятором (мишкою) та з вікнами. Дані можуть бути представлені у вигляді таблиці, саме так вони представлені в даній роботі. Наявність у FoxPro мови програмування дозволяє створювати складні системи обробки даних, орієнтовані під конкретно поставлені завдання

Декілька років тому бази даних були орієнтовані насамперед на професійних програмістів, а не звичайних користувачів. Основним споживачем таких систем був військово-промисловий комплекс. З часом бази даних як і всі інформаційні технології починають застосовуватись практично у всіх сферах людської діяльності банківській, акціонерних товариств, приватних компаній. Люди розуміють, що інформація – це гроші. Її втрата або несвоєчасне одержання можуть дорого коштувати. Саме цим можна пояснити настільки бурхливий ріст інформаційних технологій і стрімкий розвиток систем управління базами даних (СУБД).

Перші системи керування базами даних з'явилися в середині шістдесятих і підтримували ієрархічну модель даних, у якій між записами існували відносини предок / нащадок. Через короткий час, були розроблені мережеві бази даних, в основу яких була закладена значно більш складна мережева модель. У кожної з цих моделей були свої плюси і мінуси, що зіграли ключову роль у розвитку реляційної моделі.

У 1970 році стаття наукового співробітника компанії IBM доктора Е.Ф. Кодда про реляційні моделі даних зробила революцію в підході до збереження й обробки інформації. На основі цієї моделі в сімдесяті роки були розроблені перші реляційні бази даних, а в даний час вони розглядаються як стандарт для сучасних комерційних СУБД.

У реляційних базах даних вся інформація зведена в таблиці, рядки і стовпці яких називаються записами і полями відповідно. Ці таблиці одержали назву реляцій, тому модель стала називатися реляційною. Записи в таблицях не повторюються. Їхня унікальність забезпечується первинним ключем, що містить набір полів, які однозначно визначають запис. Для швидкого пошуку інформації в базі даних створюються індекси по одному або декільком полям таблиці. Значення індексів зберігаються в упорядкованому виді і містять посилання на записі таблиці. Для автоматичної підтримки цілісності зв'язаних даних, що знаходяться в різних таблицях, використовуються первинні і зовнішні ключі. Для вибірки даних з декількох зв'язаних таблиць використовуються значення одного або декількох співпадаючих полів. [1]

Сучасне виробництво немислимо без систем різного ступеня складності, що управляють певними процесами. Але будь-якій системі, що управляє, необхідне відповідне інформаційне і програмне забезпечення, інакше вона не зможе продуктивно працювати. Якщо розглядати інформаційне забезпечення (бази даних), то сучасний ринок програмного забезпечення може запропонувати досить великий вибір систем управління базами даних (СУБД), орієнтованих на різних користувачів: від дрібних підприємців до крупних підприємств і корпорацій. Наш вибір FoxPro обумовлений перш за все різносторонністю цієї СУБД, зручністю як для розробника додатків, так і для звичайного користувача. Наявність в ній мови програмування дозволяє створювати складні системи обробки даних, орієнтовані на конкретні завдання і навіть під конкретного користувача. З іншого боку, в ній відбиті і в різній мірі використовуються багато сучасних технологій програмування: ActiveX, COM, SQL, ODBC, OLE, DCOM, API і ISAPI, і багато що інше. При всьому цьому вона зберегла сумісність із старими версіями під DOS, створеними ще фірмою Fox Software. Якщо ще додати, що FoxPro реалізований також в середовищах Macintoch і Unix, то наш вибір стає обґрунтованим.

Завданням даної дипломної роботи є побудова бази даних і виконання над нею заданих операцій. База даних і таблиця були створені за допомогою діалогових вікон, а операції з базою даних виконані за допомогою програм.

У даній дипломній роботі розглянуте питання реалізації реляційної бази даних, що містить дві таблиці. Поставлене завдання реалізації самої бази даних, а також інформаційних запитів до неї в СУБД FoxPro. База даних і таблиця були створені за допомогою внутрішніх візуальних засобів вибраної СУБД, а інформаційні запити оформлені у вигляді окремих програм, що входять в проект Visual FoxPro.

# 1. Інформаційні системи ти засоби їх програмної реалізації

**База даних** – це єдине, централізоване і певним чином впорядковане сховище даних певної предметної області, тобто якоїсь галузі виробничої чи невиробничої сфери, яка займається вирішенням задач певного напрямку. Наприклад, завод, фабрика, магазин, кур’єрська служба. Отримати доступ до певних баз даних за допомогою спеціальних програм, які одержали назву систем управління базами даних (СУБД).

Як приклад бази даних можна навести бібліотеку, картотеку відділення міліції, журнал академгрупи, відеокасети певної камери спостереження та ін.

Бази даних мають декілька рівнів представлення. Найнижчий – це фізичний, на якому дані представляються байтами на певних типах пристроїв для запам'ятовування. Фізичний рівень представлення інформації у базах даних доступний тільки для вузького кола спеціалістів.

Термін «інформація» означає пояснення, викладання, повідомлення. Відомо багато визначень цього поняття, які даються за різних підходів до нього в різних наукових галузях. Наприклад, під інформацією розуміють ті відомості, які зменшують ступінь невизначеності нашого знання про конкретний об’єкт. Кібернетика, для якої інформація є центральним поняттям, визначає його як співвідношення між відомостями (даними) та їх одержувачами. У такому разі під відомостями розуміють будь-які дані, які містять знання відносно будь-кого і будь-чого.

У теорії інформаційних систем обробки даних інформація ототожнюється з будь-якими відомостями (даними), тобто тлумачиться як сукупність відомостей про будь-кого. При кібернетичному підході інформацією є лише нові, корисні, вагомі для користувача відомості, і задача полягає в їх здобутті. Природно, що така інформація має потенційно міститися в згаданих відомостях. В іншому випадку ніякої інформації одержати не вдасться. При підході о інформації з позиції теорії автоматизованої обробки даних задачі надається інший відтінок із «сирої» інформації здобути «готову».

Розглянуті підходи до поняття інформації можна використовувати не лише при аналізі різних об’єктів, а й при дослідженні однієї загальної проблеми, наприклад, управління народним господарством. Необхідно лише чітко визначити, який зміст вкладається в інформацію. Кожна наукова галузь, а також людська практика пов’язані зі «своєю» інформацією. Залежно від того чи іншого тлумачення інформації застосовується відповідний апарат аналізу. [3]

Широке коло користувачів працює з базою даних за допомогою зручних засобів, тобто готової оформленої оболонки, що не викликає труднощів при роботі та не потребує знань формування інформації на фізичному рівні (про це говорилося вище).

Таким чином, в базах даних дані розробляються і зберігаються як єдине ціле в інтересах розв'язування всіх задач предметної області.

Кожна програма вибирає із бази лише ті дані, які потрібні тільки для даної задачі. Наприклад, для нарахування заробітної платні учителям будуть вибрані прізвища учителів, їх кваліфікація, педагогічний стаж, навчальне навантаження, а для програми складання розкладу – прізвища учителів, їх навчальне навантаження та ін.

Відмітимо ще одну важливу перевагу використання баз даних, яка полягає у тому, що забезпечується незалежність прикладних програм від даних. Це означає, що зміни в даних не викликають за собою необхідність змін у програмі. Аналогічно зміни у логіці програми не викликають необхідності зміни механізму доступу до даних. Тут, звичайно, необхідно виключити випадки видалення із бази тих даних, які використовує власна програма. При звичайному ж використанні наборів даних будь-яка зміна в даних вимагає внесення змін у відповідну програму, і навпаки.

Функцію забезпечення незалежності даних від програм реалізує система управління базами даних.

Таким чином, база даних містить інформацію, необхідну для розв'язування цілого комплексу задач даної установи, підприємства та ін. База даних може поповнюватися новими даними, а раніше введені дані можуть змінюватися або зовсім видалятися. При цьому зміни в базі даних не вимагають внесення змін у прикладні програми. [2]

В проекті передбачено можливість зручного редагування та видалення непотрібних даних.

Всяка прикладна програма є відображенням якоїсь частини реального світу і тому містить його формалізований опис у виді даних. Великі масиви даних розміщають, як правило, окремо від коду програми, що виконується, і організують у виді бази даних. Починаючи з 60-х років для роботи з даними стали використовувати особливі програмні комплекси, названі системами керування базами даних (СУБД). Системи керування базами даних відповідають за:

• фізичне розміщення даних і їхніх описів;

• пошук даних;

• підтримка баз даних в актуальному стані;

• захист даних від некоректних відновлень і несанкціонованого доступу;

• обслуговування одночасних запитів до даних від декількох користувачів (прикладних програм).

База даних – це організована структура, призначена для зберігання інформації. У сучасних базах даних зберігаються не тільки дані, але й інформація.

Це твердження легко пояснити, якщо, наприклад, розглянути базу даних обліку товарів на складі. Доступ до цієї бази можуть мати декілька працівників. Крім даних, база містить методи й засоби, що дозволяють кожному зі співробітників оперувати тільки даними, які входять до бази. У результаті взаємодії даних, що втримуються в базі, з методами, доступними співробітникам, утворюється інформація, яку вони споживають і на підставі якої в межах власної компетенції роблять введення й редагування даних.

З поняттям бази даних тісно зв'язане поняття системи керування базою даних. Це комплекс програмних засобів, призначених для створення структури нової бази, наповнення її вмістом, редагування вмісту й візуалізації інформації. Під візуалізацією інформації бази розуміється відбір відображуваних даних відповідно до заданого критерію, їхнє впорядкування, оформлення й наступна видача на пристрої виводу або передача по каналах зв'язку[3].

В світі існує безліч систем керування базами даних. Незважаючи на те що вони можуть по-різному працювати з різними об'єктами й надають користувачеві різні функції й засоби, більшість СКБД опираються на єдиний устояний комплекс основних понять. Це дає нам можливість розглянути одну систему й узагальнити її поняття, прийоми й методи на весь клас СКБД.

Відразу пояснимо, що якщо в базі немає ніяких даних (порожня база), то це однаково повноцінна база даних. Цей факт має методичне значення. Хоча даних у базі й немає, але інформація в ній все-таки є – це структура бази. Вона визначає методи занесення даних і зберігання їх у базі. Найпростіший «некомп'ютерний» варіант бази даних – діловий щоденник, у якому кожному календарному дню виділено по сторінці. Навіть якщо в ньому не записано рядків, він не перестає бути щоденником, оскільки має структуру, що чітко відрізняє його від записних книжок, робочих зошитів й іншої паперової продукції.

Бази даних можуть містити різні об'єкти. Основними об'єктами будь-якої бази даних є її таблиці. Найпростіша база даних має хоча б одну таблицю. Відповідно, структура найпростішої бази даних тотожно дорівнює структурі її таблиці.

Структуру двовимірної таблиці утворять стовпці й рядки. Їхніми аналогами в найпростішій базі даних є поля й записи. Якщо записів у таблиці поки немає, виходить, її структура утворена тільки набором полів. Змінивши склад полів базової таблиці (або їхні властивості), ми змінюємо структуру бази даних й, відповідно, одержуємо нову базу даних[1].

Поля бази даних не просто визначають структуру бази – вони ще визначають групові властивості даних, записуваних в осередки, що належать кожному з полів:

1. Ім'я поля – визначає, як варто звертатися до даних цього поля при автоматичних операціях з базою (за замовчуванням імена полів використовуються як заголовки стовпців таблиць).
2. Тип поля – визначає тип даних, які можуть утримуватися в даному полі.
3. Розмір поля – визначає граничну довжину (у символах) даних, які можуть розміщатися в даному полі.
4. Формат поля – визначає спосіб форматування даних в осередках, що належать полю.
5. Маска введення – визначає форму, в якій вводяться дані в поле (засіб автоматизації введення даних).
6. Підпис – визначає заголовок стовпця таблиці для даного поля (якщо підпис не зазначений, то як заголовок стовпця використовується властивість Ім'я поля).
7. Значення за замовчуванням – те значення, що вводиться в осередки поля автоматично (засіб автоматизації введення даних).
8. Умова на значення – обмеження, використовуване для перевірки правильності введення даних (засіб автоматизації введення, що СКБД, як правило, для даних, що мають числовий тип, грошовий тип або тип дати).
9. Повідомлення про помилку – текстове повідомлення, що видається автоматично при спробі введення в поле помилкових даних.
10. Обов'язкове поле – властивість, що визначає обов'язковість заповнення даного поля при наповненні бази.
11. Порожні рядки – властивість, що дозволяє введення порожніх строкових даних (від властивості «Обов'язкове поле» відрізняється тим, що ставиться не до всіх типів даних, а лише до деяких, наприклад до текстового).
12. Індексоване поле – якщо поле має цю властивість, всі операції, пов'язані з пошуком або сортуванням записів за значенням, що зберігається в даному полі, істотно прискорюються. Крім того, для індексованих полів можна зробити так, що значення в записах будуть перевірятися по цьому полю на наявність повторів, що дозволяє автоматично виключити дублювання даних.

Оскільки в різних полях можуть утримуватися дані різного типу, то й властивості полів можуть розрізнятися залежно від типу даних. Так, наприклад, список вищевказаних властивостей полів відноситься в основному до полів текстового типу. Поля інших типів можуть мати або не мати ці властивості, але можуть додавати до них і свої. Наприклад, для даних, що представляють дійсні числа, важливою властивістю є кількість знаків після десяткової коми. З іншого боку, для полів, використовуваних для зберігання малюнків, звукозаписів, відео кліпів й інших об'єктів OLE, більшість вищевказаних властивостей не мають змісту[6].

Таблиці баз даних, як правило, допускають роботу з набагато більшою кількістю різних типів даних:

1. Текстовий – тип даних, використовуваний для зберігання звичайного неформатованого тексту обмеженого розміру (до 255 символів).
2. Числовий – тип даних для зберігання дійсних чисел.
3. Поле Мемо – спеціальний тип даних для зберігання більших обсягів тексту (до 65 535 символів). Фізично текст не зберігається в полі. Він зберігається в іншому місці бази даних, а в полі зберігається покажчик на нього, але для користувача такий поділ помітний не завжди.
4. Дата/час – тип даних для зберігання календарних дат і поточного часу.
5. Грошовий – тип даних для зберігання грошових сум. Теоретично, для їхнього запису можна було б користуватися й полями числового типу, але для грошових сум є деякі особливості (наприклад, пов'язані із правилами округлення), які роблять більш зручним використання спеціального типу даних, а не настроювання числового типу.
6. Лічильник – спеціальний тип даних для унікальних (не повторюваних у полі) натуральних чисел з автоматичним нарощуванням. Природне використання – для порядкової нумерації записів.
7. Логічний – тип для зберігання логічних даних (можуть приймати тільки два значення, наприклад Так чи Ні).
8. Гіперпосилання – спеціальне поле для зберігання адрес URL Web-об'єктів Інтернету. При щиглику на посиланні автоматично відбувається запуск браузера й відтворення об'єкта в його вікні.
9. Майстер підстановок – це не спеціальний тип даних. Це об'єкт, настроюванням якого можна автоматизувати введення даних у поле так, щоб не вводити їх вручну, а вибирати їх зі списку, що розкривається.

Бази даних – це теж файли, але робота з ними відрізняється від роботи з файлами інших типів, створюваних іншими додатками. Всю роботу з обслуговування файлової структури бере на себе операційна система. Для бази даних пред'являються особливі вимоги з погляду безпеки, тому в них реалізований інший підхід до збереження даних[15].

Бази даних – це особливі структури. Інформація, що в них утримується, дуже часто має суспільну цінність. Нерідко з однією й тією же базою працюють тисячі людей по всій країні. Від інформації, що втримується в деяких базах, може залежати благополуччя великої кількості людей. Тому цілісність вмісту бази не може й не повинна залежати не від конкретних дій якогось користувача, що забув зберегти файли перед вимиканням комп'ютера, не від перебоїв в електромережі.

Проблема безпеки баз даних вирішується тим, що в СКБД для збереження інформації використовується подвійний підхід. У частині операцій, як звичайно, бере участь операційна система комп'ютера, але деякі операції збереження відбуваються в обхід операційної системи.

Методично правильно починати роботу з олівцем й аркушем паперу в руках, не використовуючи комп'ютер. На даному етапі він просто не потрібний. Неоптимальні рішення й прямі помилки, закладені на етапі проектування, згодом дуже важко усуваються, тому цей етап є основним.

Технічне завдання на проектування бази даних повинен надати замовник. Однак для цього він повинен володіти відповідною термінологією й знати, хоча б загалом, технічні можливості основних СКБД. На жаль, на практиці таке положення зустрічається не завжди[1].

З'ясувавши основну частину даних можна приступати до створення структури бази, тобто структури її основних таблиць:

1. Робота починається зі складання основного списку полів – він може нараховувати десятки й навіть сотню позицій.
2. Відповідно до типу даних, розташовуваних у кожнім полі, визначають найбільш підходящий тип для кожного поля.
3. Далі розподіляють поля основного списку по базових таблицях. На першому етапі розподіл роблять по функціональній ознаці. Ціль – забезпечити, щоб введення даних в одну таблицю проводилося, по можливості на одному робочому місці.
4. У кожній з таблиць задається ключове поле. У якості такого вибирають поле, дані в якому повторюватися не можуть. Наприклад, для таблиці даних про книги таким полем може служити інвентарний номер книги. Якщо в таблиці взагалі немає ні яких полів, які можна було б використати як ключові, завжди можна ввести додаткове поле типу Лічильник – воно не може містити повторюваних даних по визначенню.
5. За допомогою олівця й паперу розкреслюють зв'язки між таблицями. Таке креслення називається схемою даних. Існує кілька типів можливих зв'язків між таблицями. Найпоширенішими є зв'язки «один до багатьох» й «один до одного». Зв'язок між таблицями організується на основі загального поля, причому в одній з таблиць воно обов'язково повинне бути ключовим, тобто на стороні «один» повинне виступати ключове поле, що містить унікальні, неповторювані значення. Значення на стороні «багато хто» можуть повторюватися.
6. Розробкою схеми даних закінчується «паперовий» етап роботи над технічною пропозицією, після чого час приступати до безпосереднього створення бази даних.

Варто пам'ятати, що по ходу розробки проекту замовникові неодмінно будуть спадати на думку нові ідеї. На всіх етапах проектування він прагне охопити єдиною системою всі нові й нові підрозділи й служби підприємства. Можливість гнучкого використання його побажань багато в чому визначається кваліфікацією розроблювачів бази даних. Якщо схема даних складена правильно, підключати до бази нові таблиці неважко. Якщо структура бази нераціональна, розроблювач може отримати серйозні труднощі й ввійти в суперечність із замовником. Протиріччя виконавця із замовником завжди свідчать про недостатню кваліфікацію виконавців. Саме по цьому етап попереднього проектування бази даних варто вважати основним. Від його успіху залежить, наскільки база даних стане зручною, і чи будуть із нею працювати користувачі. Якщо відзначається, що користувачі бази «саботують» її експлуатацію й воліють працювати традиційними методами, це говорить не про низьку кваліфікацію користувачів, а про недостатню кваліфікацію розроблювача бази[8].

На цьому етапі завершується розробка бази даних, і на наступному етапі починається її основне проектування. Із цього моменту варто почати роботу із СКБД.

В якості предметної області для розробленого проекту реляційної бази даних було вибрано облік побутової техніки на складі.

# 2. Опис предметної області

## 

## 2.1 Постановка задачі

Вихідні дані задачі являють собою записи заданої структури, що повинні вводитися з клавіатури, а потім виводитися у файл даних на магнітний диск. Отже, однієї з підзадач повинна бути задача створення файлу даних на магнітному диску.

Створений файл даних необхідно переглянути на екрані або вивести на друк у виді таблиці з печаткою заголовка і шапки цієї таблиці. Для цього наступної підзадачею повинна бути задача перегляду файлу даних. Також повинна бути можливість додавання записів у створений файл даних.

Крім того, для діалогу користувача із системою необхідно створити так називане, «Меню».

Предмет складання бази даних – облік замовлень кур’єрської служби. Складемо концептуальну модель представлення реальності в базі даних. Згідно умові, єдиною об'єктною множиною є об'єктна множина «друкована продукція». Кожний з цих об'єктів володіє однаковим по структурі безліччю атрибутів (ознак). Кожний з атрибутів характеризує конкретний об'єкт з якої-небудь сторони: кількість, якість, ціна і т.д.

Таким чином, база даних містить інформацію, необхідну для розв'язування цілого комплексу задач даної установи, підприємства та ін. База даних може поповнюватися новими даними, а раніше введені дані можуть змінюватися або зовсім видалятися. При цьому зміни в базі даних не вимагають внесення змін у прикладні програми. [2]

База даних побутова техніка міститиме такі дані про:

* Назву продукції, що реалізується;
* Код продукції;
* Ціну (за одиницю продукції);
* Ідентифікаційний номер продукції;
* Кількість замовленої продукції;
* Сплачену загальну ціну;

В проекті передбачено можливість зручного редагування та видалення непотрібних даних.

## 

## 2.2 Вибір програмного середовища

При виборі програмного середовища я орієнтувався на такі три причини:

**Причина 1.** При великій кількості окремих файлів, або деякі з них мають занадто багато інформації, що заважає роботі з даними. До того ж працювати з такими об’ємами даних не дозволяють обмеження по пам’яті програми або системи.

**Причина 2.** При використанні даних різними способами: для інформації по конкретним домовленостям (наприклад рахунки-фактури), для залікового аналізу (наприклад, щоквартальні звіти про обсяги продаж) або для прогнозування окремих ситуацій. Тому приходиться розглядати дані з різних сторін, що суттєво заважає створенню єдиної структури представлення даних, що задовольняє всі ваші потреби.

**Причина 3.** Є необхідність в використанні одних і тих же даних кількома спеціалістами. Скажімо, введенням, оновленням та аналізом інформації займаються різні люди. Якщо в електронну таблицю або документ вносити зміни одночасно може тільки одна людина, то з таблицею в базі даних можуть працювати одразу декілька користувачів. При цьому гарантується, що вони завжди мають справу з останніми версіями даних.

Саме тому я і вибрав середовище Microsoft FoxPro.

# 3. Опис створення програми

## 3.1 Проектування баз даних

З розвитком комп'ютерної техніки зросла складність інформаційних систем і обсяги баз даних[11]. У даний час розробка таких систем – це задача для колективів розроблювачів, що вимагає спеціальних методик і інструментів. Розробку інформаційних систем прийнято розбивати на наступні етапи:

• етап аналізу предметної області (зовнішній рівень проектування);

• етап проектування (інфологічний рівень проектування);

• даталогічний етап;

• етап тестування і супроводу.

Розроблювач повинний враховувати процеси, що відбуваються в реальному житті. Тому методика організації вихідних матеріалів проекту повинна дозволяти як можна більш швидке внесення змін у готовий проект. Чималу роль тут грає і виразна документованість проекту.

Проектування бази даних повинне починатися з аналізу предметної області, у результаті якого створюється її опис. Цей опис може виконуватися за допомогою звичайної мови, таблиць, графіків і т. п.

Всяка прикладна програма є відображенням якоїсь частини реального світу і тому містить його формалізований опис у виді даних. Великі масиви даних розміщають, як правило, окремо від коду програми, що виконується, і організують у виді бази даних. Починаючи з 60-х років для роботи з даними стали використовувати особливі програмні комплекси, названі системами керування базами даних (СУБД). Системи керування базами даних відповідають за:

• фізичне розміщення даних і їхніх описів;

• пошук даних;

• підтримка баз даних в актуальному стані;

• захист даних від некоректних відновлень і несанкціонованого доступу;

• обслуговування одночасних запитів до даних від декількох користувачів (прикладних програм).

З розвитком комп'ютерної техніки зросла складність інформаційних систем і обсяги баз даних[11]. У даний час розробка таких систем – це задача для колективів розроблювачів, що вимагає спеціальних методик і інструментів. Розробку інформаційних систем прийнято розбивати на наступні етапи:

• етап аналізу предметної області (зовнішній рівень проектування);

• етап проектування (інфологічний рівень проектування);

• даталогічний етап;

• етап тестування і супроводу.

Розроблювач повинний враховувати процеси, що відбуваються в реальному житті. Тому методика організації вихідних матеріалів проекту повинна дозволяти як можна більш швидке внесення змін у готовий проект. Чималу роль тут грає і виразна документованість проекту.

Проектування бази даних повинне починатися з аналізу предметної області, у результаті якого створюється її опис. Цей опис може виконуватися за допомогою звичайної мови, таблиць, графіків і т. п.

**Зовнішній рівень проектування**

На зовнішньому рівні проектування перш за все потрібно вивчити функціонування об’єкта управління і визначитись, які саме дані доцільно зберігати в базі. Результатом такого проектування є перелік задач, що мають розв’язуватись системою, та перелік атрибутів, що мають зберігатись в БД для успішної автоматизації поставлених задач[2].

Проектування на цьому рівні не виключає елементів надлишковості, неузгодженості і дублювання.

Існує два підходи для проектування на цьому рівні:

– від запиту;

– від предметної області.

При проектуванні від запиту вивчають всі потреби користувачів та створюють відповідну систему. Таке проектування здійснюється швидко і не потребує значних затрат. Але при зміні потреб користувачів доводиться змінювати систему.

При проектуванні від предметної області вивчають всю предметну область та створюють загальну систему. Таке проектування здійснюється довше і вимагає значних затрат, але створена система стійка до вимог користувачів.

На практиці комбінують ці два підходи, тобто вивчають потреби користувачів та розробляють ширшу систему для врахування змін потреб, що можуть відбутись найближчим часом.

**Інфологічний рівень проектування**

На рівні інфологічного проектування будується інформаційна логічна модель предметної області (модель даних). Тут відбувається структуризація даних, в якій усуваються елементи надлишковості, неузгодженості і дублювання та відображення інформаційної особливості об’єкта управління без врахування обмежень конкретної СУБД. Для проектування БД необхідний деякий загальний підхід, який з самого початку гарантує нам надійність зберігання даних і простоту маніпуляції ними. Такий підхід назвемо моделлю даних.

Структура БД визначається покладеної в її основу моделлю даних. Існують різні моделі баз даних, наприклад реляційна модель, графова модель, мережева модель і інші. Найбільш розповсюдженої в даний час є реляційна модель даних.

При реляційному підході дані представляються у вигляді двовимірних таблиць – найбільш природному для людини. Розглянемо основні поняття реляційних баз даних.

Тип даних – це поняття має такий же зміст, як і в мовах програмування. Всі існуючі сучасні бази даних підтримують спеціальні типи даних, призначені для збереження даних цілого типу, дробового з крапкою, що плаває, символів і рядків, календарних дат.

Домен – це потенційна безліч значень простого типу даних, вона має подібність з підтипом даних у деяких мовах програмування. Домен визначається двома елементами – типом даних і логічним вираженням, що застосовується до даних. Якщо результат цього вираження дорівнює значенню «істина», то екземпляр даних належить домену[9].

Відношення – це двовимірна таблиця особливого виду, що складається з заголовка і тіла.

Заголовок – це фіксована безліч атрибутів, кожний з яких визначений на якомусь домені, причому між атрибутами і визначальними доменами існує взаємно однозначна відповідність.

Приведені вище поняття є теоретичними і використовуються при розробці мовних засобів і програмних систем реляційних СУБД. У повсякденній роботі замість них використовуються їхні неформальні еквіваленти:

* відношення – таблиця;
* атрибут – стовпчик або поле;
* кортеж – запис або рядок.

Таким чином, ступінь відношення – це число колонок у таблиці, а кардинальне число – кількість рядків.

Для даного відношення завжди існує набір атрибутів, що однозначно ідентифікують кортеж. Такий набір атрибутів називається ключем.

Ключ повинний задовольняти наступним вимогам:

• повинний бути унікальним;

• повинний бути мінімальним, тобто видалення будь-якого атрибута з ключа веде до порушення унікальності.

На практиці як первинний ключ часто застосовують спеціальний числовий атрибут – автоінкрементне поле, значення якого може генеруватися тригером або спеціальними засобами, визначеними в механізмі СУБД.

Засновник реляційного підходу Дейт установив, що реляційна модель складається з трьох частин:

• структурної;

• маніпуляційної;

• цілісної.

У структурній частині моделі фіксуються відношення, як єдина структура даних, використовувана в реляційної моделі

У маніпуляційній частині фіксуються два базових механізми маніпулювання реляційними базами – реляційна алгебра і реляційне числення.

Під цілісною частиною розуміють якийсь механізм забезпечення цілісності даних. Цілісна частина укладає в собі дві основних вимоги цілісності реляційних баз даних – цілісність сутностей і цілісність по посиланнях[4].

Вимога цілісності сутностей полягає в тому, що будь-який кортеж будь-якого відношення повинний бути відмінним від будь-якого іншого кортежу цього відношення, тобто іншими словами, будь-яке відношення повинне мати первинний ключ. Ця вимога повинна виконуватися, якщо виконуються базові властивості відносин.

Вимога цілісності по посиланнях полягає в тому, що для кожного значення зовнішнього ключа, що з'являється у відношенні, що посилається, у відношенні, на якому веде посилання, повинний найтися кортеж з таким же значенням первинного ключа, або значення зовнішнього ключа повинне бути невизначеним (тобто ні на що не вказувати).

Описані в даному пункті основні поняття не відносяться до якої-небудь конкретної реалізації бази даних, а є загальними для них усіх. Таким чином, ці поняття є основою визначеної загальної моделі, що називається реляційною моделлю даних.

На підставі моделі даних складемо словник даних. Словник даних – це система, в якій зберігаються відомості про об'єкти, їх атрибути, про значення і формати представлення даних. Опишемо призначення і властивості полів реляційної таблиці «товари».

* Найменування товару. Служить первинним ключем, по якому можна дістати доступ до будь-якого рядка таблиці. Тип даних – строковий (Character), довжина – 20 символів. Ширина поля – 20 символів. Можливі значення – назви товарів, що мають відношення до офісу.
* Ціна одиниці товару. Зберігає ціну певного виду товарів. Тип даних – грошовий (Currency) точністю до 2 знаків після коми. Ширина поля – 7 символів. Можливі значення обмежені шириною поля.
* Кількість одиниць товару. Зберігає число одиниць товару, що знаходяться в даний момент на складі. Тип даних – цілий (Integer). Ширина поля – 4 символи. Можливі значення обмежені шириною поля.
* Одиниця вимірювання. Зберігає назву одиниці вимірювання товару. Тип даних – строковий (Character), довжина – 15 символів. Ширина поля – 15 символів. Можливі значення – відповідно до першого поля таблиці.
* Дата надходження. Зберігає число, місяць і рік надходження товару. Тип даних – вираз дати (Date). Ширина поля – 8 символів. Можливі значення записуються у форматі: мм/дд/гггг, де мм – номер місяця (01..12), дд – день (01..31), гггг – номер року.
* Фірмавиробник. Зберігає назву фірми виробника товару. Тип даних – строковий (Character), довжина – 20 символів. Ширина поля – 20 символів. Можливі значення – різні’.
* Постачальник. Зберігає номер партії завозу товару. Тип даних – строковий (Character), довжина – 11 символів. Ширина поля – 11 символів. Можливі значення не обмежені.

При проектуванні реляційної бази даних необхідно вирішити питання про найбільш ефективну структуру даних. Основні цілі які при цьому наслідуються:

* забезпечення швидкого доступу до даних в таблицях
* виключення непотрібне повторення даних, яке може стати причиною помилки при вводі і нераціонального використання дискового простору
* забезпечення цілісність даних таким чином, щоб при зміні одних об’єктів автоматично відбувалися відповідні зміни в зв’язаних з ними об’єктами

Процес зменшення надлишковості інформації в базі даних називається нормалізацією. В теорії нормалізації баз даних розроблені досить формалізовані підходи до розбиття даних, які володіють складною структурою.

Теорія нормалізації оперує з п’ятьма нормальними формами таблиць. Ці форми призначені для зменшення надлишкової інформації від першої до п’ятої нормальної форми. Тому кожна наступна форма повинна задовольняти умовам попередньої і деяким додатковим умовам. При практичному проектуванні баз даних четверта і п’ята форми зазвичай не використовуються.

**Перша нормальна форма таблиці**

Таблиця в першій нормальній формі повинна задовольняти такі умови:

* Таблиця не повинна мати записи що повторюються
* В таблиці повинні бути відсутніми групи полів що повторюються
* Рядки повинні бути невпорядковані
* Стовпчики повинні бути не впорядковані

Для задоволення умови 1 кожна таблиця повинна мати унікальний індекс. Умова 2 поступає видалення груп що повторюються.

**Друга нормальна форма таблиці**

Про таблицю кажуть що вона знаходиться в другій нормальній формі, якщо:

* Вона задовольняє вимогам першої нормальної форми
* Любе не ключове поле однозначно ідентифікується повним набором ключових полів

З приведеного вище означення слідує, що поняття другої нормальної форми застосовуване тільки до таблиць, які мають складовий індекс.

**Третя нормальна форма таблиці**

Про таблицю кажуть, що вона знаходиться в третій нормальній формі, якщо вона задовольняє вимогам другої нормальної форми.

Жодне з не ключових полів таблиці не ідентифікується з допомогою іншого не ключового поля.

Зведення таблиці до третьої нормальної форми має на увазі розділення таблиці з ціллю розміщення в окрему таблицю (або декілька таблиць) стовпчиків, які не залежать від повного ключа. В результаті такого розбиття кожне з не ключових полів повинне стати незалежним від якого небуть іншого не ключового поля.

Структура бази містить 12 інформаційних об’єктів різних типів, що дає можливість відображати повну інформацію як про саму службу, так і про продукцію, яка реалізується.

Коди продукції та міститимуться в полі **серійнийном** в таблицях замовників та продукції;

Назва товару, що реалізується міститиметься в полі **Товар**, а ціна одиниці даного продукту (в гривнях) в полі **ціна**;

Назва підприємства чи установи, що виробляє продукцію міститься в полі **Виробник** таблиці замовників;

Кількість замовленої продукції(штук) відображена в полі **кількпродаж**;

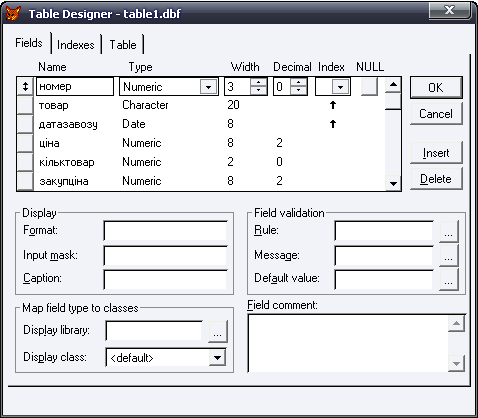
Вище описані дані зручно вводяться та редагуються у відповідних формах. Крім того користувач може переглянути вже введені дані в загальному вигляді.

Середовище Microsoft Visual FoxPro 8.0 дає програмісту можливість виконувати поставлені завдання як самостійно, так і за допомогою великої кількості «помічників».

**3.2 Створення таблиць**

Дані в майбутню базу завантажуватимуться з 2 основних таблиць (**Table1.dbf, Table2.dbf**). Розглянемо структуру кожної з них.

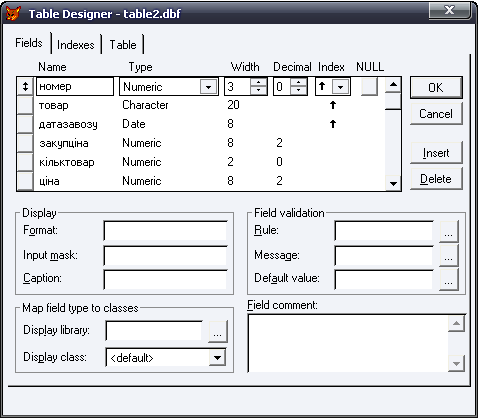
Структура таблиці **Table1.dbf**



Як видно з структура таблиці **Table1.dbf** має вигляд:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Ширина поля** | **Індекс** | **Знаків після коми** | **Значення поля** |
| Товар | Character | 30 | + |  | Назва товару |
| Виробник | Character | 20 | + |  | Фірма виробник техніки |
| Параметри | Character | 10 |  |  | Розміри або параметри техніки |
| Номер | Numenic | 3 | + |  | Табельний номер |
| Кільктовар | Numenic | 2 | + | 2 | Кількість товару |
| Ціна | Numenic |  |  |  | Ціна техніки |
| Закупці на | Numenic |  |  | 2 | Закупочна ціна техніки |
| Дата завозу |  | 8 | + |  | Дата завозу товару |

Структура таблиці **Table2.dbf**



Як видно з структура таблиці **Table2.dbf** має вигляд:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Ім’я поля** | **Тип даних** | **Ширина поля** | **Індекс** | **Знаків після коми** | **Значення поля** |
| Товар | Character | 30 | + |  | Назва товару |
| Виробник | Character | 20 | + |  | Фірма виробник техніки |
| Параметри | Character | 10 |  |  | Розміри або параметри техніки |
| Номер | Numenic | 3 | + |  | Табельний номер |
| Кільктовар | Numenic | 2 | + | 2 | Кількість товару |
| Ціна | Numenic |  |  |  | Ціна техніки |
| Закупці на | Numenic |  |  | 2 | Закупочна ціна техніки |
| Дата завозу |  | 8 | + |  | Дата завозу товару |

У Visual FoxPro7.0. вся інформація зберігається в базі даних, що складається з таблиць, відносин між таблицями, індексів, тригерів і збережених процедур. Кожна таблиця має унікальне ім'я і зберігається в окремому файлі, найменування якого збігається з ім'ям таблиці. Створений файл має розширення DBF.

Кожна створювана таблиця може мати зв'язані з нею індекси, використовувані для упорядкування даних і швидкого пошуку необхідних записів, причому одна таблиця може мати кілька індексів.

Для збереження значень полів типу Memo і General застосовуються окремі файли. Memo-полю чи таблиць містять текстову інформацію, а полючи типу General використовуються, як правило, для збереження двійкової інформації і даних інших додатків, що працюють у середовищі Windows.

У Visual FoxPro реалізовані тригери, що дозволяють централізовано обробляти події, що виникають при будь-яких змінах у базі даних. Також можна створювати збережені процедури, що є частиною бази даних і можуть використовуватися при описі таблиць, для перевірки введених даних, визначення значення за замовчуванням і т. п.

Надзвичайно зручним і корисним засобом доступу до бази даних є представлення даних. Представлення даних дозволяють поєднувати дані таблиць і відображати них у більш зручному виді. Ви можете вибрати тільки цікавлячі вас поля таблиць, об'єднати кілька полів в одне поле, обчислити підсумкові значення і задати нові імена полів таблиці. Як правило, кількість представлень у базі даних набагато перевершує кількість таблиць. В міру експлуатації бази даних їхня кількість безперервно росте. У багатьох інформаційних системах доступ до даних, включаючи перегляд, додавання і редагування, здійснюється тільки за допомогою представлень даних. Цей підхід дозволяє здійснити гнучке керування доступом до інформації. При використанні представлень для вибірки даних у формах, звітах, при створенні запитів і в програмах застосовуються ті ж правила, що і для таблиць. Редагування даних, включених у представлення, можливо тільки за певних умов. Наприклад, у тому випадку, якщо воно створено на основі тільки однієї таблиці.

Для відображення і редагування даних використовуються форми, звіти, запити і програми. При створенні форм, звітів і запитів застосовуються конструктори. Тому ці компоненти часто називають конструкторськими об'єктами. Форми і звіти є складеними об'єктами, тому що вони складаються з більш дрібних об'єктів (таких як полючи, кнопки, діаграми, рамки, OLE-компоненти і т. п.), що називаються об'єктами інтерфейсу.

Форми використовуються для перегляду або введення даних у таблиці. Дані можна вводити безпосередньо в таблиці, але використання форми є більш швидким і більш ефективним способом уведення. Форма містить деякі або всі поля таблиць, у які ви вводите інформацію. Для створення форм ви можете використовувати майстер створення форм або конструктор форм. Майстер форм містить цілий ряд шаблонів, що визначають співвідношення між таблицями, що поміщаються у форму, вид відображення даних і порядок розміщення полів. Для створення складних форм застосовується конструктор форм.

Звіти використовуються для печатки, що утримується в базі дані інформації. Прикладами звітів є прайс-лист товарів, список покупців, оборотна складська відомість. Як правило, звіти створюються в тому випадку, якщо інформацію необхідно передавати кому-небудь у друкованому виді. Для створення звітів у Visual FoxPro, як і для форм, використовуються майстер і конструктор звітів. За допомогою майстра звітів ви можете швидко створити власний звіт на основі наявних шаблонів. Застосування конструктора звітів дозволяє створювати звіти довільної складності, включаючи багаторівневе угруповання даних і розміщення полів, що обчислюються.

Запити є засобом вибірки даних з однієї або декількох таблиць. У Visual FoxPro для створення запиту ви можете використовувати як конструктор запитів, так і спеціалізована мова Structured Query Language (SQL). Результати виконання запиту можуть відображатися у формі, виводитися у виді звітів і діаграм або зберігатися в зазначеній вами таблиці.

Програми, написані мовою Visual FoxPro, є об’єктно-орієнтованими. За допомогою них ви обробляєте події у формі, створюєте об'єкти, здійснюєте різні обчислення, керуєте базою даних. Для зручності роботи ви можете об'єднати програми в бібліотеки.

Для створення форм у Visual FoxPro можна використовувати не тільки базові класи, але і створювати власні. Наприклад, ви можете визначити клас форм, у якому заданий визначений колір тла і стандартний набір кнопок для керування даними. Щоб стандартизувати розробку, корисно мати один або кілька користувацьких класів для кожного базового класу. Класи, створені в Visual FoxPro, зберігаються в бібліотеках класів.

Для об'єднання компонентів створюваного додатка використовується проект, у який включаються всі перераховані компоненти. Використання проекту спрощує розробку додатка і його супровід.

Кожен компонент зберігається в окремому файлі, причому імена файлів, що містять основні компоненти, ви задаєте самостійно, а найменування файлів, що містять об'єкти, зв'язані з таблицею, збігаються з ім'ям таблиці. У залежності від типу об'єкта, що утримується в ньому, Visual FoxPro автоматично привласнює кожному файлові розширення, що допомагає в ідентифікації об'єкта. [1]

При розробці системи всі форми продукції та замовників були зв’язані між собою по індексу. Усі елементи зв’язані з базою (відповідні поля при зміні записів завантажуються синхронно в усі поля).

Кнопки на формах автоматично пов’язані із записами. Розглянемо зокрема подію при натискуванні на кнопку «Наступний запис» форми table1. Як бачимо, виконується команда 1. buttonset1.cmdNext.click що присвоює елементам, які відображають інформацію, наступний запис (для кожного поля свій)

## 3.3 Складання програмних запитів до бази даних

Приступимо до програмної реалізації бази даних. Взагалі слід зазначити, що в Visual FoxPro існують 2 шляхи реалізації майже будь-якого завдання: візуальний і програмний. У першому випадку необхідні дії виконуються за допомогою команд меню самого FoxPro і спеціалізованих майстрів. У другому – за допомогою вбудованої мови програмування, успадкованого ще від ранніх версій. У всіх випадках ми користуватимемося другим способом, окрім самого створення бази даних. Річ у тому, що на першому етапі необхідно створити так званий проект Visual FoxPro, до складу якого надалі входитимуть всі без виключення використовувані нами компоненти. Для більшої наочності бажано створювати проект «вручну». Крім того, існують 2 види таблиць FoxPro: вільні і зв'язані, тобто що входять до складу баз даних. Наша таблиця відноситься до другого типа, тоді як команда CREATE TABLE створює вільну таблицю. З урахуванням цього, створення бази даних проведемо візуально, а вся решта маніпуляцій – програмно.

## 3.4 Пошук записів за допомогою циклів WHILE і SCAN

Для послідовного пошуку найпершого запису бази даних, що задовольняє заданій FOR-умові і до тих пір, поки дотримується WHILE-умова (якщо є), застосовується команда LOCATE. Для продовження пошуку, початого командою LOCATE, застосовується команда CONTINUE.

Реалізація ітераційних циклів, тобто циклів з наперед відомою умовою їх закінчення і невідомим числом повторів, виконується наступною інструкцією:

DO WHILE <условие>

<команды>

ENDDO

Для послідовного перегляду бази даних, пошуку всіх записів, що задовольняють умовам і виконання над знайденими записами яких-небудь операцій, служить команда SCAN.

SCAN [<границы>] [FOR <условие>]

<команды>

ENDSCAN

**3.5 Формування звітів**

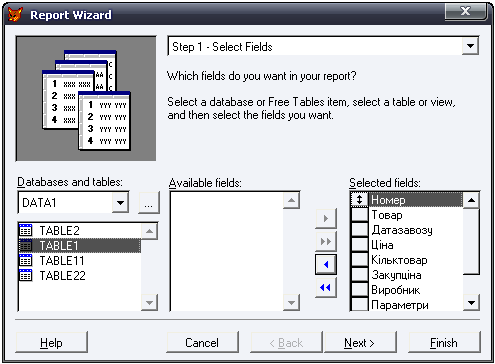
Формування звітів для такого проекту є вкрай необхідним, адже саме в звітах виводиться повна інформація по проекту. Крім того звіти є можливість роздрукувати і тим самим завершити повний цикл розробки бази даних – від проектування до виведення кінцевої сукупної інформації.

Для створення звітів в середовищі розроблений дуже зручний майстер звітів. Спочатку він надає користувачу вибір: звіт з однієї таблиці чи декількох таблиць. Після цього користувач вибирає таблицю(таблиці), дані з якої відображатимуться у звіті. Після добавлень полів, які виведуться у звіті можна вибрати поле по якому сортуватимуться результати у звіті. Далі вибираємо бланк для звіту, вид сторінки звіту (книжна чи альбомна), індексовані поля. Після цього зберігши все раніше введене ми отримуємо файл звіту (розширення для файлів звітів – **\*.frx**).

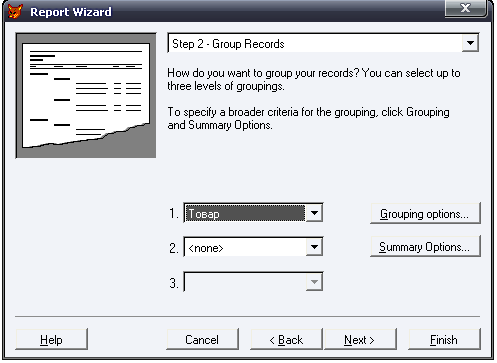
В даному проекті створимо два звіти по двох основних таблицях.

Перший звіт створимо по таблиці комп’ютерної техніки Table1**.dbf**.

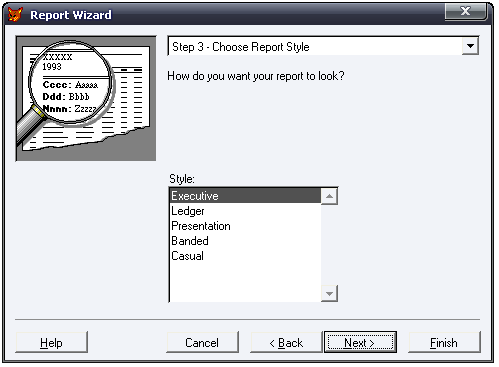
Виберемо відповідну таблицю і добавимо поля, які відображатимуться в звіті.



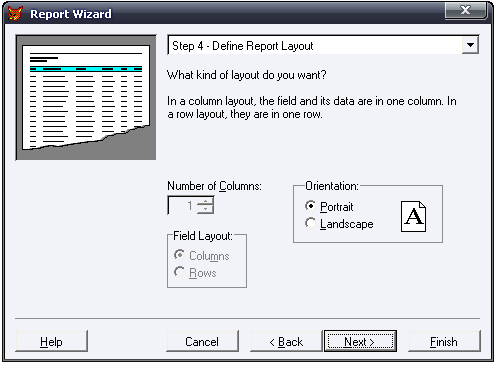
Вибираємо поле, по якому сортуватимуться всі дані в звіті (в нашому випадку це поле Товар)



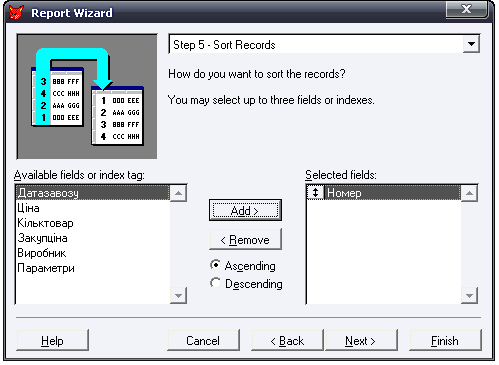
Вибираємо вид відображення звіту



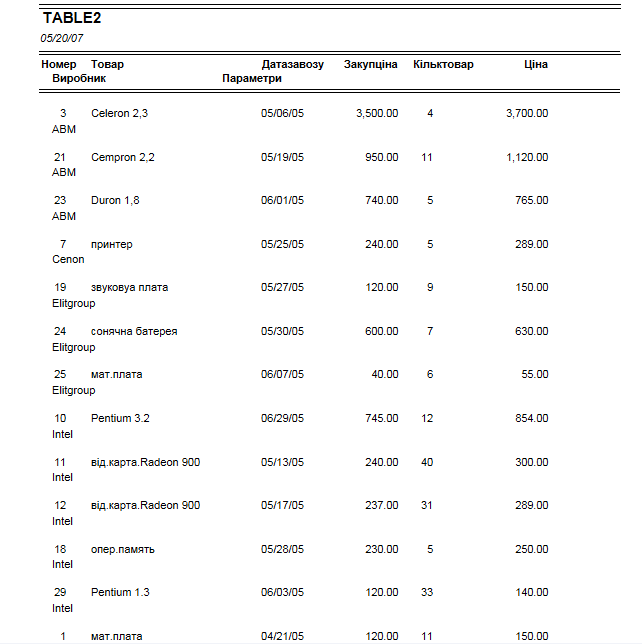
Вибираємо формат сторінки звіту (в нас – альбомна)



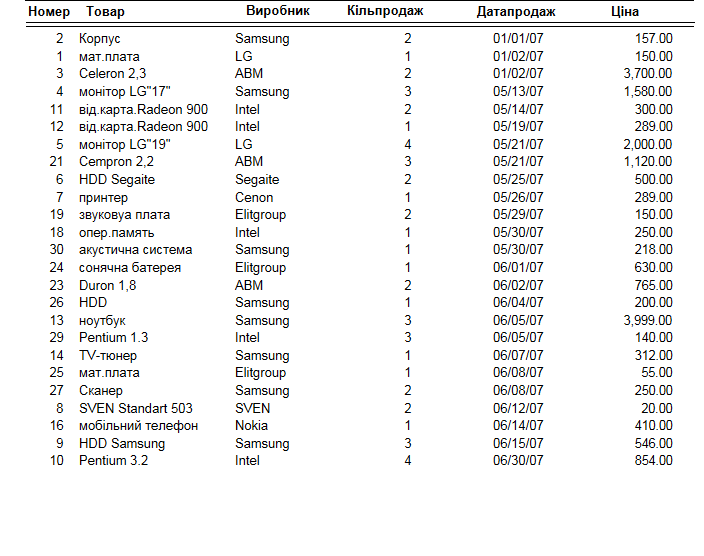
Встановлюємо індексовані поля



Після чого, зберігши і відкоригувавши деякі мітки в звіті отримуємо повністю придатний для друку документ



Аналогічно формуємо звіти для продукції, що реалізується

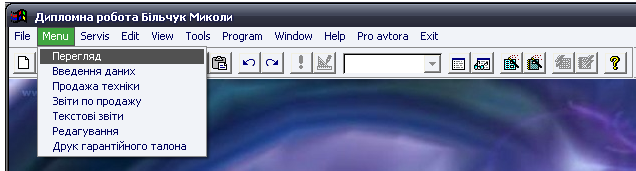


## Робота з програмою

Головне меню складається з таких пунктів. Розглянемо детальніше кожен з пунктів і події, що він викликає.



Пункт меню File – містить стандартний набор процедур відповідних цьому пункту.

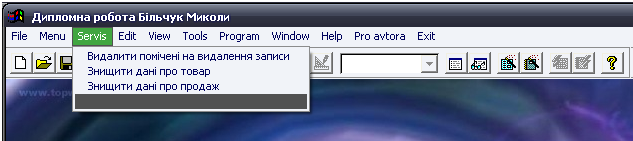


Пункт меню «Menu» відкриває наявні форму проекту, з яких користувач має доступ у всі розділи проекту. При натисканні даної кнопки виконується команда переходу до форми:

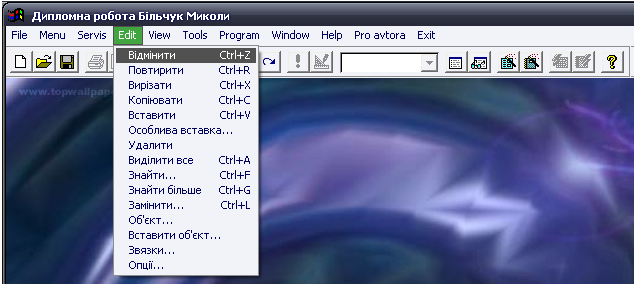
do form…

Іншим варіантом виконання цієї команди є комбінація клавіш Ctrl-M.

Пункт меню Servis – містить сервісні функції роботи з календарним планом, калькулятором та видаленням записів.



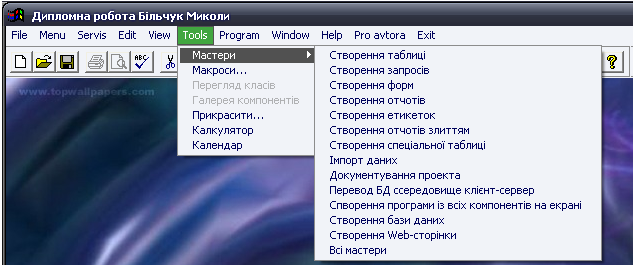
* Edit – застосовується для роботи не тільки з текстом, але й іншими об’єктами;



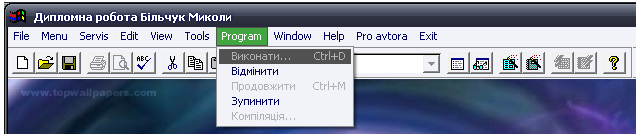
* View – містить команду «Панель інструментів»



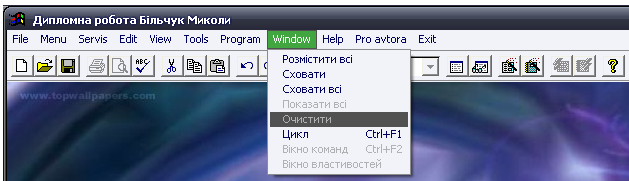
* Tools – використання різноманітних інструментів;



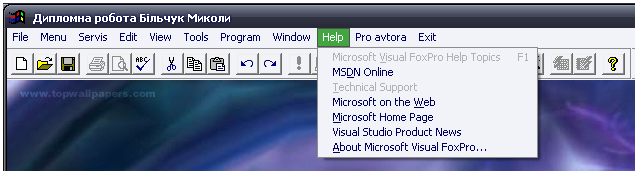
* Program – використовується для роботи з програмою;



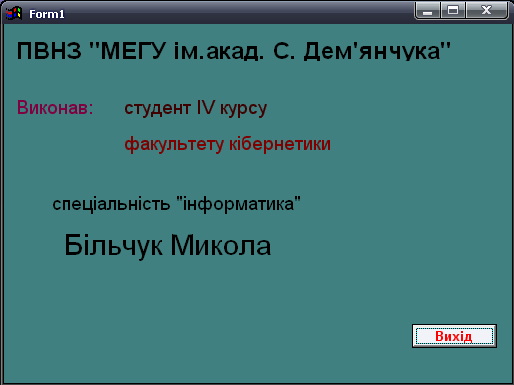
* Window – використовується для роботи з вікнами;



* Help – використовується для отримання допомоги;

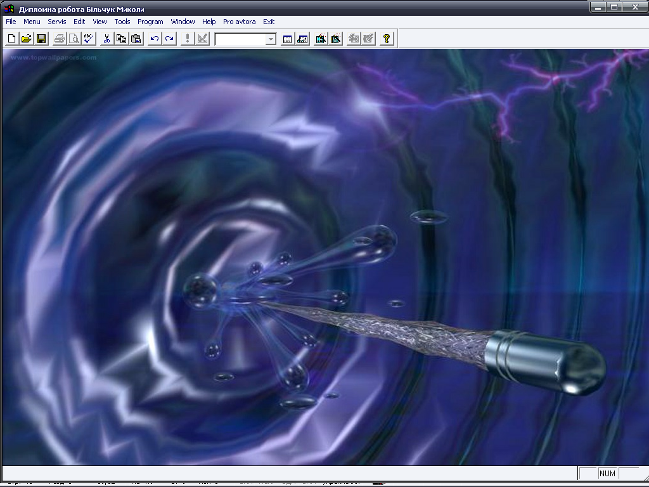


* Pro avtor – виводить вікно про автора програми;



* Exit – здійснює вихід у системне меню.

При запуску проекту запускається головне меню з якого користувач вибирає відповідний розділ проекту.



Код програми **program1.prg** має вигляд:

Set date german

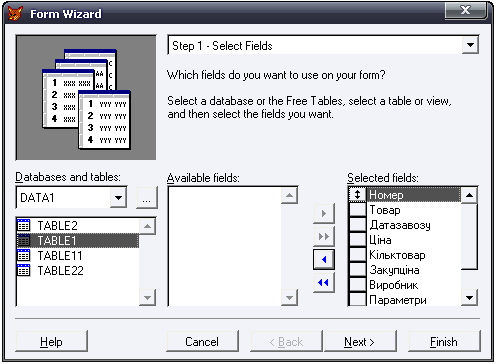
Menu1.mnx

Коди інших процедур наведені в додатку.

Для організації виведення та редагування даних будемо застосовувати форми та звіти. Для формування форм будемо використовувати майстри, що дозволить зекономити час та зменшити кількість помилок.

Форми будемо розробляти за допомогою майстрів форм, що дозволяє значно спростити роботу та виключити помилки.

Розробка форми за допомогою майстра включає в себе декілька кроків. Перш за все середовище FoxPro робить запит який тип форми створювати: на формі можна відображати дані з однієї таблиці, або з двох зв’язаних таблиць. Для форм, які відображаються дані з двох таблиць буде два кроки, щоб вказати таблиці з даними.



Далі майстер пропонує вибрати поле, за яким сортувати дані при виводі, також відображаються проіндексовані поля таблиці.

Залишилось змінити заголовки текстових полів, замінивши назви, згенеровані FoxPro на потрібні нам. Для всіх форм встановимо такі загальні властивості:

AutoCenter-true – при виведенні, форма буде знаходитись посередині екрана.

BorderStyle-1 (fixed size) забороняє зміну розміру форми.

MaxButton-false – робимо недоступною кнопку максимізації вікна.

Даний проект складається з 19 форм. Одні з них створені за допомогою майстра форм, а інші створені звичайним методом.

Розглянемо основні форми:

Форма **1.scx** створена для перегляду, сортування та пошуку даних.

Форма була створена звичайним способом. сучасні та інтуїтивно зрозумілі користувачу.

Форма **1.scx** створена для відображення наявних в базі даних товарів побутової техніки.

Форма **Table1.scx** являється формою, що служить для добавлення та зміни даних по продукції яку реалізує дана служба. Користувач в кожному новому записі може добавити інформацію про назву продукції, її код та ціну за одиницю продукції, вводити новий товар та редагувати існуючий.

Форма створена за допомогою майстра форм.

Форма **Table1.scx** служить для відображення бази даних наявної продукції, що реалізується.

Форма **avtor.scx** – це форма яка містить основну інформацію про автора програми.

Visual FoxPro складається з окремих компонентів, що використовуються для збереження інформації, її відображення і редагування.

# 

# Висновки

У курсовій роботі були розглянуті прийоми проектування і реалізації реляційних баз даних і таблиць в СУБД Visual FoxPro 6.0. Була спроектована структура реляційної таблиці, в неї були внесені дані за допомогою спеціальних запитів. Операції над даними таблиці були виконані програмним шляхом за допомогою створення автономних модулів \*.prg, що входять до складу проекту Visual FoxPro.

Автоматизація та інформатизація торкнулася всіх сфер людської діяльності. Завдання покладені на автоматизацію – полегшення умов праці звичайних людей. Автоматизація баз даних значно полегшує роботу відповідних підрозділів, робить її швидшою, якіснішою та ефективнішою.

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

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

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

База даних орієнтована на інтегровані запити, а не на одну програму, як у випадку файлового підходу, і використовується для інформаційних потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

**Список використаної літератури**

1. Попов А.А. Створення додатків для FoxPro 2.5/2.6 в DOS і WINDOWS. – М.: Видавництво «Калашников і До», 1997. – 660 с.: илл.
2. FoxPro. Language Refrence. – Microsoft Corp., 1994.
3. Пінтер Ліс. Розробка додатків в Microsoft FoxPro 2.5. – М.: ТОО Эдель, 1995.
4. Дейт К. Руководство по реляційній СУБД DB2. – М.: Фінанси і статистика, 1988.
5. Вейскас.Д, В26 Эффективная работа с Мicrosoft Access 97 – Спб: ЗАО «Издательство Питер», 1999. – 976 с.:ил.
6. Кауфельд. Дж, К45 FoxPro для «чайников». – К.: «Диалекика», 1995. – 264 с., ил.
7. Microsoft Excel для Windows 95. Шаг за шагом: Практ. Пособ\Пер. с англ. – М: Издательство ЭКОМ, 1997. – 432 с.:ил.
8. Селиджтаун.М, В26 «FoxPro 2.5 Практическое пособие» – М: изд. «Москва-Пресс», 1994. – 296 с.:ил.; 3-е Издание.
9. Лемашко Е.В., Романчуков В.Г. Программирование в системе команд СУБД семейства Fox: учебное пособие / ГАУ, М., 1998.
10. Компьютерный практикум. Программирование в среде Турбо-Паскаль и СУБД типа Fox. Методические указания к выполнению курсового проекта. /Сост.: О.Н. Леонова, И.А. Несмеянов; ГАУ, М., 1998.
11. Антифеев Дм.Д. Современные средства построения корпоративных систем поддержки принятия управленческих решений «Терн», М., 2001
12. Рогач І. Ф., Сендзюк М.А., Антонюк В.А. Інформаційні системи у фінансово-кредитних установах: Нанч. посібник. – 2-ге вид.,
13. Савчук Т.О. Організація баз даних і знань. Вінниця: ВДТУ, 2000 р.
14. Кельдер Т.Л. Системи обробки економічної інформації. Курс лекцій. // Електронна версія – http://www.zsu.zp.ua/lab/mathdep/mme/ІV/soeі/іndex.htm
15. Степанов Ю.Л. Разработка приложений баз данных для СУБД Sybase SQL Anywhere. Санкт-Петербургский филиал Военного университета ПВО. Электронная версия – http://www.cіtforum.elcat.kg/database/sql\_any/іndex.shtml