Міністерство освіти і науки України
Рівненський державний гуманітарний університет
Кафедра інформаційних технологій та моделювання

Кваліфікаційна робота
за освітнім ступенем «магістр»

на тему: Інформаційна система оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів.

Виконав: магістр 2 курсу
групи М-КН-21
спеціальності 122 «Комп’ютерні науки»
Борейчук Віктор Васильович
Керівник:
доц. Шевцова Наталія Вікторівна

Рівне – 2023

РЕФЕРАТ

Метою кваліфікаційної роботи на тему «Інформаційна система оптимального підбору витратних матеріалів та розрахунок вартості меблів» є розробка інформаційної системи для невеликих і середніх підприємств, які займаються виготовленням мебельної продукції, що вирішує проблему оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів. Система обрахунку меблевої продукції на сучасному ринку відіграє важливу роль для виготовлення та реалізації меблів. Від ефективності обрахунку значною мірою залежить термін виготовлення продукції, також збільшуються оборотні кошти, відповідно збільшується реалізація продукції та заробітна плата робітників.

Об’єкт дослідження – автоматизація технологічних процесів на виробництві меблів. Предмет дослідження – інформаційна система оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів.

Розроблена в кваліфікаційній роботі програма дозволяє в мінімальні терміни обрахувати вартість кухонних секцій, змінюючи їхні габаритні розміри, також до загальної вартості враховується ціна фасаду, стільниці, різної фурнітури, яка використовується при виготовленні кухні, і яка відіграє важливу роль в загальній вартості кухні. Програма має можливість вирахувати, за допомогою спеціальної формули, розміри кожної заготовки які використовуються при складанні секції, також за допомогою звітів роздрукувати ці розміри для створення розкрою листка ДСП в спеціальній програмі.

Система призначена для невеликих і середніх підприємств, які займаються виготовленням меблевої продукції, а також для магазинів які реалізують цю продукцію. Програма дозволяє вести базу даних матеріалів, фурнітури, аксесуарів, і головне клієнтів з детальним описом про замовлення (який матеріал використався, колір, товщина, різновид фурнітури, вартість фасадів і т.д.).

Кваліфікаційна робота складається зі змісту, вступу, чотирьох розділів, висновків та списку використаних літературних джерел із 13 найменувань. До роботи додається реферат українською та англійською мовами. Загальний обсяг роботи складає 61 сторінку.

Ключові слова: інформаційна система, бази даних, автоматизація обчислень, виробництво меблів.

ABSTRACT

The purpose of the qualification work on the topic " Information system of optimal selection of consumables and calculation of the cost of furniture" is the development of an information system for small and medium-sized enterprises engaged in the manufacture of furniture products, which solves the problem of optimal selection of consumables and calculation of the cost of furniture products. The accounting system for furniture products on the modern market plays an important role in the manufacture and sale of furniture. The production period of products depends to a large extent on the efficiency of the calculation, working capital also increases, the sales of products and wages of workers increase accordingly.

The object of research is the automation of technological processes in the production of furniture. The subject of the study is an information system for optimal selection of consumables and calculation of the cost of furniture products.

The program developed in the qualification work allows you to calculate the cost of kitchen sections in the shortest possible time by changing their overall dimensions, and the price of the facade, table top, and various accessories used in the manufacture of the kitchen, which plays an important role in the total cost of the kitchen, is also taken into account in the total cost. The program has the ability to calculate, with the help of a special formula, the dimensions of each workpiece that are used in the assembly of the section, as well as with the help of reports to print these dimensions to create a cut-out of a chipboard sheet in a special program.

The system is intended for small and medium-sized enterprises that manufacture furniture products, as well as for stores that sell these products. The program allows you to maintain a database of materials, fittings, accessories, and most importantly, customers with a detailed description of the order (what material was used, color, thickness, type of fittings, cost of facades, etc.).

The qualification work consists of a table of contents, an introduction, four chapters, conclusions and a list of used literary sources from 13 items. An abstract in Ukrainian and English is attached to the work. The total volume of work is 61 pages.

Keywords: information system, databases, computer automation, furniture production.

ЗМІСТ

ВСТУП

Актуальність теми. Проєктувати вироби з деревини, особливо меблі, нині можна не тільки на папері, а й на екрані комп'ютера за допомогою спеціалізованих програм, які наочно демонструють кожну деталь майбутнього меблевого виробу. З огляду на те, що у виробництві меблів основною сировиною являються плиткові матеріали, то найважливішого значення набуває їх раціональний розкрій. Технологічний процес розкрою плиткових матеріалів на усіх меблевих виробництвах значною мірою визначає собівартість продукції та раціональне використання дорогої сировини.

Розкрій – це операція отримання із плит заготовок потрібних розмірів і кількості з максимально можливим корисним виходом. Широке застосування плит у меблевій промисловості, зробило цю операцію однією із тих, що визначають ефективність усього виробництва. Так, наприклад, витрати на матеріали у собівартості будь-якого елемента модульного типу складають від 65 до 70%. У той же час відходи, що утворюються від розкрою плит, досягають 12%. Враховуючи, що вартість матеріалу в собівартості виробництва меблів висока, зниження витрати матеріалів і підвищення виходу панелей у процесі розкрою плит є надзвичайно важливою проблемою.

Завданням кваліфікаційної роботи є розробка інформаційної системи підрахунку вартості меблевої продукції, яка здатна вирішувати наступні задачі:

1. Візуальна підтримка вибору стандартних секцій, матеріалів – в програмі зберігається вся інформація за замовленням: про клієнта, деталі розробки секцій, вартість доставки, установки, аванс та залишок до оплати. Система забезпечує трасування між замовниками, тобто дозволяє відстежити весь список замовлень які були прийняті: від вибору секцій до розмірів конкретної заготовки.

2. Зворотний зв'язок із замовником. Замовник самостійно може підрахувати вартість кухні яка його зацікавила. Потрібно лише орієнтуватися у видах матеріалів, фурнітури та аксесуарів.

3. Гнучка методологія розробки мебелі дозволяє успішно вести розробку як однотипних, так і багатосекційних кухонь . Система реалізує фундамент для будь-якої гнучкої методології ітераційно-інкрементного процесу розробки, дозволяє планувати склад фурнітури і контролювати їх використання.

4. Можна управляти процесом розрахунку вартості меблевої продукції змінюючи коефіцієнти розходу матеріалів, нарахування вартості робіт, комп’ютерну розробку, змінюючи при цьому саму систему обрахунку. На сьогоднішній день це відіграє важливу роль в існуванні фірми-виробника.

5. Прозорість процесу. Вся інформація по процесу розробки доступна на одному екрані, у зв'язку з чим істотно спрощується контроль процесу розробки , замовник також може контролювати процес розробки або навіть приймати в ньому участь в частині управління пріоритетами реалізації або виправлення помилок.

6. Централізоване сховище документів, проєктів, даних. Всі учасники дістають доступ до найсвіжіших матеріалів, видів фурнітури. Не потрібно буде їх вибирати по каталогам, всі дані знаходяться в універсальній базі даних, яка попередньо оновлюється працівниками.

Метою дипломної роботи є розробка інформаційної системи для невеликих і середніх підприємств, які займаються виготовленням мебельної продукції, що вирішує проблему оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів.

Об’єкт дослідження – автоматизація технологічних процесів на виробництві меблів.

Предмет дослідження – інформаційна система оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів.

Апробація результатів дослідження. Результати дослідження доповідалися на звітній науковій конференції здобувачів освіти, аспірантів, викладачів РДГУ в травні 2023р.

Структура роботи. Кваліфікаційна робота складається зі змісту, вступу, чотирьох розділів, висновків та списку використаних літературних джерел, додатків. До роботи додається реферат українською та англійською мовами.

Загальний обсяг роботи складає 61 сторінку. Робота містить 11 рисунків та 14 лістингів з фрагментами коду програми. Список використаних літературних джерел складає 13 найменування.

У першому розділі проаналізовано аналогічні комерційні програми на ринку та технології розробки інформаційних систем. Другий розділ містить опис використаної архітектури типу «клієнт-сервер». У третьому розділі висвітлено особливості програмної реалізації інформаційної системи, включаючи проєктування бази даних, огляд компонент та бібліотек, використаних при розробці ІС, інтерфейс програми. У четвертому розділі оцінено ефективність інформаційної системи.

РОЗДІЛ 1
АНАЛІЗ РИНКУ ТА ТЕХНОЛОГІЇ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1. Огляд існуючих програм - аналогів

На сьогоднішній день існують потужні комерційні програми для створення різноманітної мебелі, дизайну тощо. Такі як, наприклад, від компаній Базис-центр, ІНТЕАР Лтд і Ecru. Це програми, які містять в собі потужні набори інструментів для створення мебелі, візуалізації замовлення, підрахунку вартості. З безкоштовних аналогів існують тільки демо-версії цих продуктів з обмеженою функціональністю. Ці програми не зовсім доступні за своєю вартістю, зокрема програма Woody компанії ІНТЕАР Лтд, Web-інтерфейс накладає деякі складності для підрахунку вартості мебелі, система не має зручного комплектування кухонних секцій.

1.1.1. Місце Woody в сімействі програм, пропонованих науковою фірмою ІНТЕАР

Наукова фірма ІНТЕАР заснована в 1992 році. Вона займається розробкою і реалізацією програмного забезпечення для комп'ютерного проєктування. Відмітна особливість програм, пропонованих ІНТЕАР, – це використання технології інтерактивного графічного просторового проєктування. Основний момент цієї технології полягає в тому, що шлях до робочої документації починається з тривимірної моделі, що створюється і редагованої в тривимірному просторі [6].

У 1996 році Наукова фірма ІнтеАр випустила на ринок систему комп'ютерного проєктування InteAr 3.0, а потім в 1997 році - InteAr 4.O. Ці системи орієнтовані на архітекторів і дизайнерів інтер'єрів і меблів. Вони призначені для проєктування котеджів, квартир, інтер'єрів і екстер'єру житлових і суспільних будівель, м'якою і інших меблів, сантехніки і об'єктів дизайну найрізноманітнішого функціонального призначення. При цьому

забезпечується отримання фотореалістичних зображень модельованих об'єктів.

У лютому 1999 року була запропонована широкому кругу користувачів система Woody 1.1 – спеціальний інструмент для конструктивного проєктування корпусних меблів з листових матеріалів (ДСП, ДВП, скло і т.д.). Система оперує елементами моделі, що володіють чіткою прикладною орієнтацією на конструювання корпусних меблів, завдяки чому забезпечує автоматизований підбір конструктивно-технологічних рішень і автоматичне формування проєктної документації, включаючи: комплект креслень, специфікації деталей і витрати матеріалів [8].

Проєктування спочатку відбувається на наочних зображеннях тривимірної моделі конструйованого виробу. При цьому конструктор створює не абстрактні геометричні примітиви (куб, сфера, паралелепіпед), а конкретні деталі меблів (полиці, дверці) і розміщує фурнітуру. Деталі створюються з цілком певних реальних матеріалів, описаних в обширній базі даних.

Логічні зв'язки між прикладними примітивами, з яких конструюється модель, дозволяють моделювати з'єднання деталей і розміщувати фурнітуру в тривимірному просторі. Це дозволяє контролювати коректність з'єднань і сприяє безпроблемній збірці.

Дизайнер відразу ж бачить, як виглядатиме майбутній виріб, і може показати його зображення замовникові.

В кінці 1999 року запропонована версія Woody 1.20 і програма розкрою листових матеріалів Sawyer. За період з 2000-го по 2001-й рік випущені Woody 1.30 - 1.50. В кінці 2002-го – Woody 2.O.

Нові версії Woody успадкували функціональне призначення і позитивні властивості попередніх версій. Додатково вони дозволяють моделювати фасади з фрезеруванням, засклені і фільонки дверці, стільниці, кромки фігурного профілю і фігурне фрезерування торців деталей. Система тепер містить вбудовані засоби для поповнення і редагування бази даних

матеріалів і кріплень. Починаючи з версії 1.50, Woody дозволяє підключати призначені для користувача інтерфейси, які розширюють прикладну функціональність системи. Це дає можливість спростити процес проєктування розсувних систем і шаф-купе різних конструкцій.

Woody 2.0 надає можливість створювати і сполучати похилі деталі, довільно редагувати контур деталі, формувати проєкт з декількох виробів, випускати складальні креслення і зображення.

Система розкрою Sawyer забезпечена могутнім оптимізуючим алгоритмом. Вона забезпечує автоматичне формування карт розкрою з урахуванням орієнтації текстури, технологічних відступів, чистового різу, ширину розпилу, характеристик відходів. Враховується технологія використання розпилювального устаткування. Показуються лінії резу і послідовність розпилювання. Sawyer підтримує склад листових матеріалів, що забезпечує раціональне використання залишків. Опційно доступний склад того, що комплектують, зокрема, фурнітури [7].

Системи Woody, Sawyer і InteAr можуть обмінюватися даними один з одним, забезпечуючи комплексне рішення завдань проєктування від конструювання окремого виробу корпусних меблів, до отримання карт розкрою і вражаючих ілюстрацій, що демонструють виріб в інтер'єрі замовника.

Крім того, InteAr може послужити як тривимірний редактор для створення і редагування тривимірних моделей фурнітури, засклених і фільонок дверець, фігурних профілів для кромки і фрезерування торців деталей, якими можна поповнювати базу даних Woody. На базі InteAr розроблений спеціальний полегшений варіант тривимірного редактора – 3DM. Їм опційно може доповнюватися комплект постачання Woody для самостійного вирішення користувачем завдань поповнення бази даних моделями візуалізації.

Зв'язки Woody + Sawyer виключають ручне введення розмірів заготовок деталей при формуванні завдання на розкрій, забезпечує

автоматичну передачу інформації про тип матеріалу і напрямок текстури.

Використовуючи комплекс Woody + Sawyer, користувач може отримати цілий ряд різних документів (див. рис. 1.1).

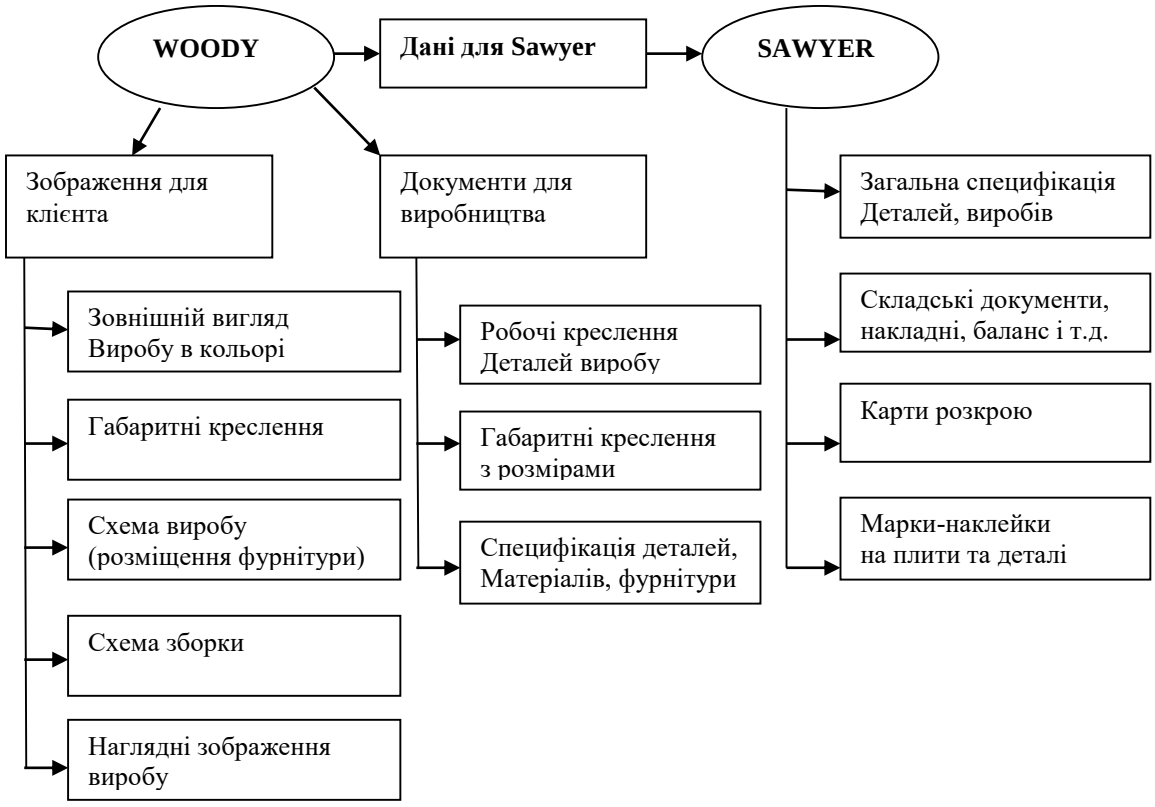


Рис. 1.1. Вихідні документи

Так, за допомогою Woody користувач може отримати зображення для клієнта і документи для виробництва. Зовнішній вигляд виробу виходить шляхом роздруку його зображення безпосередньо з Woody або за допомогою експорту зображень у файли популярних растрових форматів (BMP, JPEG, TGA). Зображення можуть бути отримані в різних режимах, зокрема, у вигляді каркасної схеми, на якій видно розташування фурнітури усередині виробу. Управляючи видимістю окремих деталей виробу, користувач може роздрукувати (або експортувати в растрові файли) серію зображень, що ілюструють послідовність збірки виробу. Зображення виробу можуть бути автоматично забезпечені винесеннями з нумерацією деталей.

Woody генерує робочі креслення і специфікації, дозволяє їх виводити безпосередньо на друкуючий пристрій або експортувати у файли для включення в документи, що готуються в рамках інших систем.

Sawyer на підставі завдання на розкрій, що включає декілька виробів, готує креслення карт розкрою, звідні специфікації витрати матеріалів за завданням, наклейки для маркировки на плити та деталі. З Sawyer можна також роздрукувати накладні і відомості підведення балансу, що відображають поточний стан складу.

1.1.2. Модуль Базис-Салон дизайн інтер'єру. Основні можливості.

Модуль Базис-Салон призначений для експлуатації в меблевих салонах. Він дозволяє:

Формувати прайс-листи меблів всіх виробників, з якими працює меблевий салон, групуючи вироби по функціональному, конструктивному або іншим ознакам.

Створювати моделі приміщення замовника з урахуванням розташування вікон, дверей, арок і інших особливостей конкретного приміщення, досягаючи високим ступенем відповідності комп'ютерній моделі і реальності.

Розставляти вибрані моделі меблів в приміщенні, переміщати їх, комбінувати різні варіанти і поєднання, змінювати колірну гамму, використовувати декоративні елементи і виконувати багато інших дизайнерських операцій, добиваючись відповідності вимог і побажань покупця можливостям продавця.

Отримувати точну вартість замовлення на будь-якому етапі роботи з клієнтом.

Формувати повний комплект необхідних документів, включаючи реалістичну «картинку» приміщення клієнта з майбутніми меблями.

Відстежувати проходження замовлення від моменту підписання договору до моменту відвантаження виробів з можливістю поетапної оплати, отримання поточної інформації про стан замовлення і т.д.

Оперативно передавати замовлення безпосередньо у виробництво, виключаючи які-небудь проміжні дії.

Структурно модуль Базис-Салон складається з трьох компонентів [6]:

1. блок формування прайс-листів (електронних каталогів) меблів;
2. блок моделювання інтер'єру і розстановки меблів;
3. блок формування виробничого завдання.

В даний час вимоги меблевого ринку істотно змістилися у бік якості і нестандартності виробів: клієнт вимагає якісні меблі, що враховують особливості його житла і відповідну його естетичним запитам, причому у нього з'явився широкий вибір виробників меблів. Це змінює умови роботи меблевих салонів, на які лягає чи не основна відповідальність за «утримання» клієнта, своєчасне і якісне виконання замовлення. Істотну допомогу в рішенні цих складних завдань може надати модуль Базис-Салон.

Для формування електронних каталогів меблів передбачено два режими. В тому випадку, якщо для автоматизації проєктування і виробництва меблів на підприємстві вже використовується система Базис-Конструктор-мебляр, формування каталогів відбувається на основі конструкторсько-технологічної інформації практично в автоматичному режимі. Інакше досить скористатися можливостями модуля Базис-Мебляр, що трохи збільшить трудомісткість наповнення інформацією створюваного каталога [6].

У блоці моделювання інтер'єру на основі використовуваних прайс-листів і розмірів приміщення відбувається спільна робота дизайнера або продавця-консультанта і замовника над дізайн-проєктом.

З пропонованого асортименту виробів вибираються вподобані, вони встановлюються в потрібні місця або вішаються на стіну, підбираються матеріали корпусів і фасадів, приміщення, що найбільш гармонують з інтер'єром.

При необхідності замовлення може доповнюватися стандартними купувальними елементами: миттям, витяжками, газовими або електричними плитами і іншою побутовою технікою і предметами інтер'єру. Високий ступінь реалістичності і оперативний розрахунок вартості дозволяють

підібрати найбільш відповідний варіант замовлення як з естетичної, так і з фінансової точки зору.

Під час роботи над дізайн-проєктом блок формування виробничого завдання автоматично готує всю необхідну інформацію для передачі замовлення у виробництво або відвантаження готових виробів з складу. Подібний підхід дозволяє практично повністю виключити суб'єктивні помилки і значно скоротити час виконання замовлення.

Інтерфейс модуля Базис-Салон реалізований в традиційної для всіх модулів системи Базис-Конструктор-мебляр простій і зручній формі. Він дозволяє однаково ефективно працювати як продавцеві-консультантові, що починає, так і досвідченому дизайнерові інтер'єрів.

Модуль Базис-Салон орієнтований на дві групи учасників меблевого ринку:

- виробники меблів, що мають власну торгову мережу;
- меблеві салони, що реалізовують вироби багатьох виробників.

У першому випадку модуль Базис-Салон органічно вбудовується в єдиний ланцюжок «проєктування - виробництво - реалізація меблів». Замовлення, прийняте від клієнта, автоматично передається в інші модулі системи:

А) модуль Базис-Кошторис для розрахунку потрібної кількості матеріалів і що комплектують;

Б) модуль Базис-Розкрій для формування карт розкроу листових і погонних матеріалів;

В) модуль БАЗИС-ЧПУ для формування програм, що управляють, для фрезерно-присадних верстатів і оброблювальних центрів.

У другому випадку після затвердження замовлення клієнтом формуються документи на відвантаження необхідних упаковок з складу, а у разі відсутності яких-небудь позицій - на придбання їх у виробника.

У обох випадках використання модуля Базис-Салон дозволяє «утримати» потенційного клієнта за рахунок творчого підходу до задоволення його запитів і істотного скорочення часу обробки замовлення.

1.1.3. Дизайнерська програма PRO100 компанії ECRU

Ця програма призначена для моделювання меблів і оформлення приміщень цими меблями. Програма дозволяє створювати об'ємні моделі будь-яких меблів і елементів оформлення інтер'єрів, розташовувати їх в заздалегідь створеному приміщенні, переглядати створене в декількох режимах відображення, у тому числі і фотореалістичному, друкувати і експортувати в графічний файл створені сцени, вести прайс-листи, проводити розрахунки за ціною створюваних проєктів, витратою матеріалів, друкувати звіти [11].

Програма PRO100 застосовується на всіх тих етапах процесу виробництва меблів, на яких споживач хоче усучаснити свою роботу, спираючись на досягнення комп'ютерної техніки.

Отже, програма PRO100 може застосовуватися для проєктування меблів «з нуля», для побудови власної бібліотеки, для планування постачання у виробництві, для аранжування інтер'єрів або ж, нарешті, для сприяння в процесі безпосереднього продажу - на кожному з цих етапів доступна негайна візуалізація, різні типи видів, оцінка і рапорти. Завдяки цьому дана програма витримує випробування як в невеликих, одноосібних або таких, що складаються з декількох чоловік фірмах і салонах продажу, так і на великих фабриках, що мають мережі власних магазинів і великі виробничі центри.

Простота обслуговування, швидкість дії, а також постійна можливість введення змін в проєкті, відмінно полегшує життя виробникам і продавцям меблів.

1.2. Аналіз основного завдання проєкту та вибір методів розв’язку завдань

Перед створенням інформаційної системи оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів було вивчено питання доцільності створення цього ПЗ. Проведено огляд та аналіз існуючих програм, визначено, який повинен бути продукт, розглянуто основні методи вирішення завдань.

Основною проблемою при обрахунку вартості кухонних секцій полягає точний розрахунок витрачених матеріалів, підрахунок кількості та вартості фурнітури яка використовується в даному замовленні. Підрахунок кріплення, що дозволяє побудувати кухонні секції, займає великий проміжок часу в підрахунку всього продукту. Тому що кожен елемент кріплення має свої властивості при його розташуванні та залежить від виду матеріалів з якого виготовляється виріб.

Крім ефективності процесу підрахунку велике значення для замовника і продавця має зворотний зв'язок із замовником і прозорість процесу, що дозволяє отримати точну інформацію про вибрані секції, види та типи матеріалів (колір, товщина ДСП, малюнок фасаду, профіль фрезерування торців фасаду, різноманітність ручок, ножов і т.д.), а також можливість на кожному етапі підрахунку змінити дані відповідно до побажань замовника та його платоспроможності.

Програма не повинна мати обмежень по ресурсах.

Має бути передбачена можливість передачі і зберігання файлів, проєктів, текстових файлів та малюнків.

Система повинна мати інтуїтивно зрозумілий графічний інтерфейс для перегляду вибраних секцій та встановлення їх розмірів.

Програма передбачає всі основні моменти, необхідні для підрахування вартості: від вибору нових стандартних секцій до встановлення розмірів нестандартних, від фільтрування необхідних даних про фурнітуру та матеріали до редагування самої бази даних. Таким чином, передбачена

можливість створювати проєкт із фрагментів, тобто можна самостійно створити нові види секцій. Дані структуруються так, що можливо передбачити все необхідне, що пов’язане з секціями. Для передачі та зберігання інформації використовується база даних створена за допомогою EMS SQL Manager, яка зберігає всі дані які використовуються програмою.

1.2.1. Трьохрівнева система

Якщо, наприклад, у фірми є декілька офісів по всій країні, і в кожному з них – парк комп’ютерів з 10-30 штук, то вирішити проблему постійного оновлення (внесення змін, дадавання даних і т.д.) корпоративних програм можна наступним чином. Самий простий вихід - використовувати трьохрівневу систему: клієнт, сервер логіки (так звана «бізнес-логіка») і сервер додатку. В такій системі вся логіка зібрана в сервері додатків. Якщо щось змінилося в базі даних або в логіці обробки даних, достатньо обновити його, і всі клієнти будуть працювати по-новому без якихось патчів [2].

Перевага такої системи ще й у тому, що на клієнтських машинах не потрібно тримати драйверів доступу до якихось баз. Клієнти повинні лише знати, де знаходиться сервер додатків, вміти до нього підключитися і правильно відобразити дані.

Для прикладу, уявімо собі класичну задачу – поява нової версії бази даних або перехід на базу якісно більш нового рівня. Ну от не вистачає нам уже можливостей MySQL, захотілося перейти на Oracle. Для цього переставляється сервер баз даних, змінюється сервер додатків на підключення до нової бази – і клієнти готові до роботи. Їх обновляти не треба.

Але саме цікаве те, що клієнтська програма може бути яка завгодно. Можна написати сценарії, які дозволяють працювати з сервером додатків прямо з браузера. В такому випадку із базою зможуть працювати користувачі на будь-якій платформі (Windows, Linux і т.д.).

Трьохрівнева система показана на рисунку 1.2.

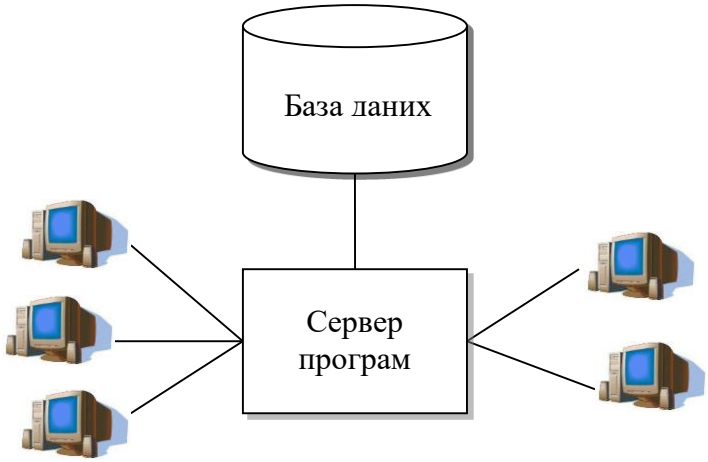


Рис. 1.2. Трьохрівнева система.

Не дивлячись на наявність сервера додатків, немає сенсу закидати в нього всю логіку обробки даних. Якщо використовується потужна база даних, яка підтримує зберігаючі процедури і функції, то краще перекласти частину логіки на сервер бази. В такому випадку внесені в зберігаючий код зміни вступають в силу моментально і не треба навіть оновлювати сервер додатків.

Якщо в мережі не так уже й багато комп’ютерів (не більше 20) і сервер достатньо потужний, то можна сервер додатків і базу даних розмістити на одному фізичному сервері. В такому випадку обмін даними між сервером додатків і базою буде відбуватися всередині одного комп’ютера, а не по мережі, що може суттєво понизити навантаження на мережеве обладнання.

Припустимо, сервер додатків і база даних знаходяться на різних серверах. Результати запитів будуть спочатку йти через комутатор від бази даних до сервера додатків, а потім через той же комутатор до комп’ютерів клієнтів. Таким чином, по мережі двічі пролітають одні й ті ж дані. Щоб цього уникнути, доцільно об’єднати в одному фізичному сервері логіку і дані.

Отже, якщо пишемо базу, з якою буде працювати одночасно лише одна людина, то однозначний вибір – локальна база. Якщо ж із базою будуть працювати хоча б дві людини, то не потрібно вигадувати мережеві з’єднання, а краще скористатися технологією клієнт-сервер. Вона позбавляє

мережу від зайвого трафіку, більш надійна при багатокористувацькій роботі і дає максимальну кількість можливостей.

Якщо кількість користувачів катастрофічно збільшується і з'являються проблеми з оновленням системи, то найкращий вибір – перехід на трьохрівневу систему. Це дещо складніше в розробці, але набагато краще під час супроводу.

1.2.2. Етапи та основні правила створення програмного забезпечення.

Процес проєктування та створення програмної системи проходив наступні етапи [12]:

Специфікація вимог:

1. Формулювання задач та існуючих методів дослідження.
2. Огляд літератури та існуючих систем щодо тематики проєкту.

Аналіз:

3. Аналіз основного завдання.
4. Аналіз окремих проєктних рішень.

Проєктування:

5. Побудова структурної схеми мети.
6. Вибір оптимальних методів розв'язку задач.
7. Розробка програмних модулів, оформлення інтерфейсу.
8. Наскрізний аналіз проєкту з метою виявлення помилок.

Реалізація:

9. Кодування і відлагодження програмних модулів.
10. Оформлення програмної документації.

Тестування і верифікація:

11. Тестування програмного комплексу, усунення знайдених помилок.
12. Аналіз ступеня виконання задач, оцінка ефективності створеної системи.

Перераховані вище етапи є життєвим циклом програмного забезпечення. Він є ітеративним, тобто допускає багатократне повторення своїх етапів. В ході розробки можуть виникнути проблеми, які будуть вимагати змін вимог до системи; під час реалізації може виникнути необхідність переглянути результати, отримані під час розробки; під час тестування можуть бути виявлені помилки і так далі.

Розглянемо детальніше деякі аспекти програмування згідно хронології життєвого циклу програмного продукту.

При створенні проєкту намагалися дотримуватись наступних правил щодо проєктування програм.

✓ Чітке визначення поставленої задачі (ще до етапа реалізації). Виконано повне і по можливості точне визначення вимог до системного комплексу.

2. Вибір алгоритму. Було переглянуто і проаналізовано декілька варіантів вирішення поставлених завдань і обрано той, який би найкраще забезпечував необхідні рішення.

3. Вибір мови програмування. Серед універсальних мов високого рівня безперечним лідером є мова C/C++. До безумовних переваг останніх належать: наявність операторів, що забезпечують базові програмні елементи; можливість програмування на низькому рівні з доступом до адрес оперативної пам’яті; великі бібліотеки службових підпрограм та класів. З іншого боку, вони мають і ряд недоліків, до яких в першу чергу слід віднести наявність синтаксичних неоднозначностей, які обмежують можливості компілятора щодо контролю правильності програми.

Але оскільки, використовуючи лише мову C/C++, не можна було максимально точно вирішити існуючі проблеми щодо реалізації необхідної функціональності програмної системи, то використовувались також спеціалізовані мови розробника ПЗ, зокрема мова розширеної розмітки XML (головним чином, можливості бібліотеки libxml2).

4. Створення універсального коду програм. Тобто реалізації програм таким чином, щоб можна було їх компілювати під інші платформи, відмінні від Windows.

5. Використання бібліотечних підпрограм. Використання бібліотек стандартних підпрограм значно спрощує створення великих програмних проєктів.

У проєкті використана система управління об'єктно-орієнтовними базами даних PostgreSQL і C-бібліотека libpq для доступу до бази даних. Також Libxml2 - вільно ліцензована бібліотека мови C для обробки XML.

Використання бібліотечних підпрограм збільшує надійність і зменшує складність програмування.

6. Встановлення мети проєкту. При створенні програмного проєкту неможливо досягти найкращих критеріїв якості одночасно по всіх показниках. Тому слід визначити головну мету проєкту, що й було зроблено перед етапом реалізації.

До основних цілей належать:

1) обмежений термін виконання;
2) зручність і простота використання;
3) ефективність (по швидкодії).

7. Забезпечення правильності програми. Для забезпечення цієї вимоги використовувались два підходи:

- ретельне тестування (повноти тестування практично неможливо досягти в загальному випадку);
- створення проєкту, в якому помилки одразу зведені до мінімуму.

1.2.3. Дерево цілей та реалізація програми.

Після огляду існуючих систем управління процесом розробки ПЗ та аналізу основних задач, були розглянуті методи їх реалізації. Було побудовано структурну ієрархію системного комплексу, яка наведена у

додатку Б. Створено початкове дерево головних задач, яке має наступний вигляд (див. рисунок 1.3).

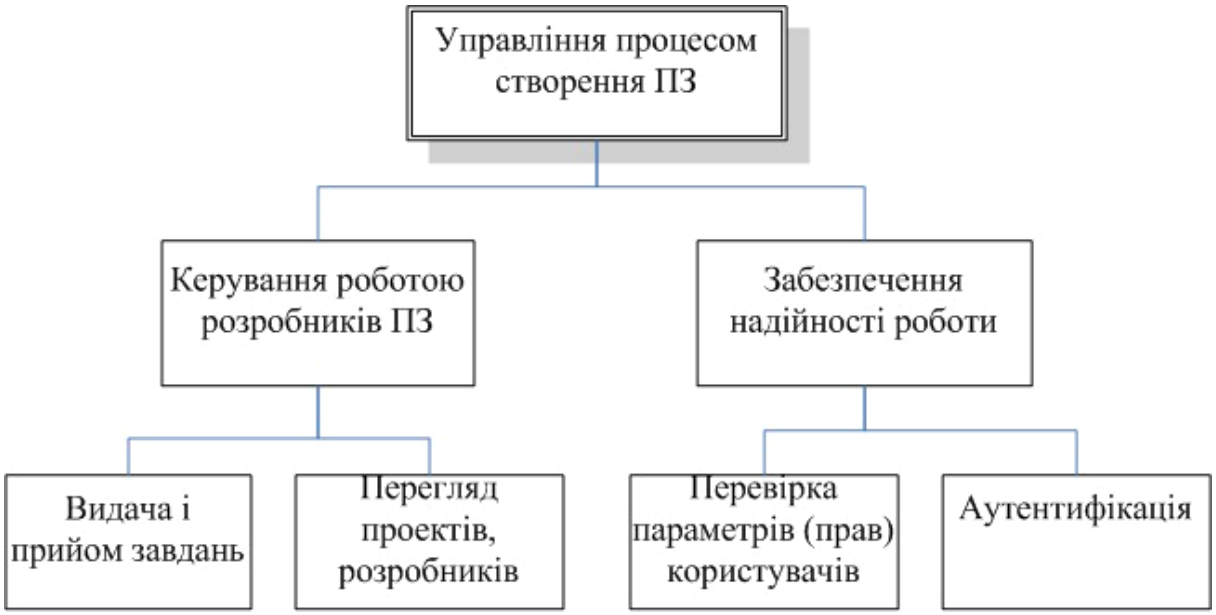


Рис. 1.3. Дерево основних задач «Інформаційної системи»

Для надійної та зручної роботи з базою даних вирішено використовувати трьохрівневу архітектуру організації системи. Загальне зображення цієї архітектури, що має безпосереднє відношення до проєкту, можна побачити на рисунку 1.4. Так, база даних міститься на сервері баз даних, в ролі якого виступає PostgreSQL. Саме ця система управління об’єктно-орієнтовними базами даних обрана тому, що вона може вільно поширюватися і має потужний інструментарій для підтримки роботи з БД. Також PostgreSQL сумісний з запитами SQL.

Рис.1.4. Трьохрівнева організація «Інформаційної системи»

Сервер додатків (додаток Д) звертається до БД за допомогою бібліотеки pqLib, а із клієнтськими програмами він «спілкується» по протоколу TCP/IP з допомогою повідомлень у форматі XML. Алгоритм роботи сервера додатків наведений у додатку Г. Програма-клієнт, механізм роботи якої показано в алгоритмі додатку В, звертається до серверу додатків, який у свою чергу бере дані з БД і відправляє відповідь клієнту. Відповідно до отриманої відповіді динамічно будується інтерфейс форм. Відповіді клієнтові містять також правила побудови наступних можливих запитів. Так реалізована побудова динамічного інтерфейсу користувача, клієнтського сокета і запитів.

Розглянемо детальніше алгоритм роботи програми-клієнта. Після запуску програми відбувається спроба під’єднання до сервера додатків. Якщо спроба вдала, то від сервера приходить відповідь про те, що з’єднання відбулося. Далі відсилається запит на отримання меню форми. Якщо запит виконаний, то отримуємо відповідь з інформацією щодо динамічної побудови цього меню та побудови форми запиту отримання переліку об*єктів одного типу. Функція побудови меню має наступний вигляд.

Лістинг 1.1

```
//Build menu
    while (mnode != NULL) {
      string mname = (char*)xmlGetProp(mnode, (const xmlChar*)"name");
      TMenuItem *NewItem = new TMenuItem(mainMenu);
      mainMenu->Items->Add(NewItem);
      NewItem->Caption = mname.c_str();
      NewItem->Name = mname.c_str();
      string mType = (char*)xmlGetProp(mnode,(const xmlChar*)"rtype");
      if (mType == "Get") {
        cMenuGet * cmg = new
cMenuGet(mname,xdoc,Socket,(char*)xmlGetProp(mnode,(const
xmlChar*)"request"));
```

```
        getList.push_back(cmg);
        TMenuItem *NewItemG = new TMenuItem(mainMenu);
        NewItem->Add(NewItemG);
        NewItemG->Caption = (char*)xmlGetProp(mnode,(const
xmlChar*)"request");
        NewItemG->Name = (mname+"_Get").c_str();
        NewItemG->OnClick = cmg->menuClick;
       }
      mnode = mnode->next;
    }
```

Нижче подані структури і мапи, в яких зберігається інформація, необхідна для побудови форми запиту отримання переліку об’єктів одного типу. Інформація містить в собі перелік полів фільтрації та сортування.

Лістинг 1.2

```
typedef struct _scbData {
      string id; //id selected fields
      string fname; //description. View in CB
 }stSCBData;
 typedef struct _filterData {
      string fname; //fname in XML
      string descr; //description in XML
      map<string,string> mapFlt; //[desct]=id
 }stFilterData;
 map<string, stFilterData*,less<string>,allocator<stFilterData*> > mapFilter;
//[fname]=str
 map<string, stFilterData*,less<string>,allocator<stFilterData*> > mapSortBy;
//[fname]=str
```

Після побудови меню форми користувач може отримати інформацію, яка його цікавить (про проєкти чи розробників), вибравши певний пункт меню. Після вибору певного пункту створюється клас cViewEdit (наведений

нижче), який по номеру ID необхідного об’єкта отримує необхідну інформацію з сервера. Форми перегляду, додавання і редагування користувачів містяться у додатку Ж, а інформація про проєкти – у додатку К.

Лістинг 1.3

```
/*class cViewEdit*/
/*Змінні для зберігання інформації, яка виводиться на екран, а також назви запитів, заговка форми і т.п.*/
typedef struct _viewLine {
      string nField, //name of field
            id,     //id record
            descr,  //description
            val;    //value field
    }stViewLine;
    xmlDocPtr inDoc;
    string viewName;
    string viewBlockName, addBlockName, editBlockName;
    string addNameRequest, editNameRequest;
    string menuName;//user,role...
    // [id record]=data record
    map<string,list<stViewLine*, allocator<stViewLine*> > > mapViewLines;
    map<int,string,less<int>,allocator<string> > mapSG_ID;
/*блоки змінних для зберігання інформації, необхідної для побудови форм для запитів додавання і редагування*/
//add block
    typedef struct _scbData {
      string id; //id selected fields
      string fname; //description. View in CB
    }stSCBData;
    typedef struct _selectData {
      string fname; //fname in XML
```

```
    string descr; //description in XML
    map<string,string> mapSlct; //[desct]=id
  }stSelectData;
  map<string,string> mapSlctVal;
  map<string, stSelectData*,less<string>,allocator<stSelectData*> >
mapSelect; //[fname]=str
  //end add block
  //edit block
  typedef struct _seiData {
    string description;
    map<string,string > mapSct; //[desct]=id
  }stEIData;
  map<string, stEIData*,less<string>,allocator<stEIData*> > mapEdit;
  //[fname]=str
  map<string,string> mapEditData; //[fmane]=fval
  //end edit block
```

Тоді користувач прогладає отриману інформацію, редагує чи/і додає необхідні дані. Таким чином, весь процес обміну інформацією із сервером відбувається з допомогою запитів.

РОЗДІЛ 2
МЕРЕЖЕВЕ ПРОГРАМУВАННЯ

2.1. Архітектура клієнт-сервер

Мережеве програмування – це обширна область з великим вибором різних технологій для охочих встановити зв'язок між декілька машинами. Серед них такі прості, як послідовна лінія зв'язку, і такі складні, як системна мережева архітектура (SNA) компанії IBM. На сьогодні протоколи TCP/IP – основна технологія побудови мереж [8]. Це обумовлено розвитком Internet і найпоширенішого додатку – Всесвітньої павутини (World Wide Web). Проте і до появи Web TCP/IP був поширеним методом створення мереж. Це відкритий стандарт, і на його основі можна об'єднувати машини різних виробників. До кінця 90-х років TCP/IP завоював лідируюче положення серед мережевих технологій і воно зберігається і досі.

Хоча постійно говориться про клієнтів і сервери, не завжди очевидно, яку роль відіграє конкретна програма. Іноді програми є рівноправними учасниками обміну інформацією, не можна однозначно стверджувати, що одна програма обслуговує іншу. Проте у випадку з TCP/IP відмінність більш чітка. Сервер прослуховує порт, щоб знайти вхідні TCP-з'єднання або UDP-датаграми від одного або декількох клієнтів. З другого боку, можна сказати, що клієнт - це той, хто починає діалог першим.

Розглянуто три типові випадки архітектури клієнт-сервер, показані на рисунку 2.1 [13].

В першому випадку клієнт і сервер працюють на одній машині (рис.2.l а). Це найпростіша конфігурація, оскільки немає фізичної мережі. Дані, що посилаються, передаються стеку TCP/IP, але не поміщаються у вихідну чергу мережевого пристрою, а закільцьовуються системою і повертаються назад в стек, але вже як прийняті дані.

Рис. 2.1. Типові приклади архітектури клієнт-сервер

На етапі розробки таке розміщення клієнта і серверу дає певні переваги, навіть якщо в реальності вони працюватимуть на різних машинах. По-перше, простіше оцінити продуктивність обох програм, оскільки мережеві затримки виключаються. По-друге, цей метод створює ідеальне лабораторне середовище, в якому пакети не пропадають, не затримуються і завжди приходять в правильному порядку.

Примітка. Принаймні, майже завжди. Навіть в цьому середовищі можна створити таке навантаження, що UDP-датаграми пропадатимуть.

І, нарешті, розробку вести простіше і зручніше, коли можна все відлагоджувати на одній машині. Зрозуміло, навіть в умовах промислової експлуатації цілком можливо, що клієнт і сервер працюватимуть на одному комп'ютері.

На другому прикладі конфігурації (рис. 2.1.б) клієнт і сервер працюють на різних машинах, але в межах однієї локальної мережі. Тут має місце реальна мережа, але умови все ж таки близькі до ідеальних. Пакети рідко втрачаються і практично завжди приходять в правильному порядку. Така

ситуація дуже часто зустрічається на практиці. Причому деякі додатки призначені для роботи тільки в такому середовищі. Типовий приклад - сервер друку. В невеликій локальній мережі може бути тільки один такий сервер, обслуговуючий декілька машин. Одна машина (або мережеве програмне забезпечення на базі TCP/IP, вбудоване в принтер) виступає в ролі серверу, який приймає запити на друк від клієнтів на інших машинах і ставить їх в чергу до принтера [13].

У третьому прикладі (рис. 2.1.в) клієнт і сервер працюють на різних комп'ютерах, зв'язаних глобальною мережею. Цією мережею може бути Internet або корпоративна Intranet, але головне – додатки вже не знаходяться усередині однієї локальної мережі, так що на шляху IP-датаграм є, принаймні, один маршрутизатор. Таке оточення може бути більш «ворожим», ніж по-перше двох випадках. У міру зростання трафіку в глобальній мережі починають переповнюватися черги, в яких маршрутизатор тимчасово береже пакети, що поступають, поки не відправить їх адресату. А коли в черзі більше немає місця, маршрутизатор відкидає пакети. В результаті клієнт повинен передавати пакети повторно, що приводить до появи дублікатів і доставки пакетів в неправильному порядку. Ці проблеми виникають досить часто [13].

2.2. Модель клієнт-сервер

Більшість мережевих додатків написана таким чином, що з одного боку присутній клієнт, а з іншого – сервер. При цьому сервер надає певні сервіси клієнтам.

Можна розділити сервери на два класи: послідовні (iterative) і конкурентні (concurrent). Процес функціонування послідовного серверу продемонстровано на блок схемі (рис.2.2.).

Рис. 2.2. Алгоритм роботи послідовного серверу

В процесі виконання кроку П2 може виникнути проблема. Вона полягає в тому, що в цей час ніякі інші клієнти не можуть обслуговуватись.

Конкурентний сервер, з іншого боку, працює таким чином (рис. 2.3.).

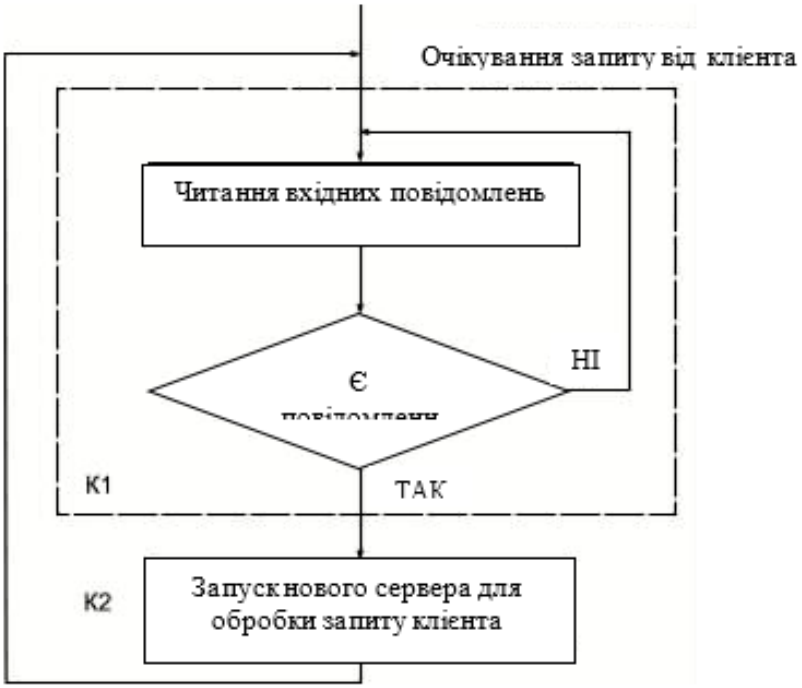


Рис. 2.3. Алгоритм роботи конкурентного серверу

Запуск нового серверу на кроці К2 для обробки запиту клієнта може виглядати як створення нового процесу, задачі, залежно від того яка операційна система лежить в основі цього серверу. Новий сервер обробляє поступивший запит клієнта повністю. Після закінчення сервер знищується.

Перевага конкурентного серверу полягає в тому, що він просто запускає інші сервери для обробки запитів від клієнтів. В подібному випадку кожний клієнт має власний сервер. Передбачається, що операційна система підтримує багатозадачність і обслуговування декількох клієнтів одночасно [12].

Ми розділили саме сервери, а не клієнтів, спеціально, тому що в звичайних умовах клієнт не може сказати, з яким сервером, послідовним або конкуруючим, він спілкується.

В загальному випадку, сервери TCP – конкурентні, а сервери UDP – послідовні. Проте з цього правила можуть бути виключення.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Проєктування бази даних

База даних – сукупність даних, що використовується при функціонуванні ІС, організована за певними правилами, які передбачають загальні принципи опису, зберігання і маніпулювання даними і незалежна від прикладних програм (ГОСТ 24.003-84). Система управління базами даних – це сукупність програм і мовних засобів, які призначені для управління даними в базі даних і забезпечують взаємодію її з прикладними програмами (ГОСТ 20886- 85).

Основні принципи створення БД: цілісність, вірогідність, контроль, захист від несанкціонованого доступу , єдність і гнучкість, стандартизація та уніфікація, адаптивність, мінімізація введення і виведення інформації (однократність введення інформації, принцип введення-виведення тільки змін) [1].

Вимоги до інформаційного забезпечення (ГОСТ 24.104-85 "Автоматизовані системи управління. Загальні вимоги") такі [1]:

1. Інформаційне забезпечення має бути достатнім для виконання всіх функцій ІС, які автоматизуються.

2. Для кодування інформації, яка використовується тільки в цій ІС, має бути застосовані класифікатори , які є у замовника ІС.

3. Для кодування в ІС вихідної інформації, яка використовується на вищому рівні, мають бути використані класифікатори цього рівня, крім спеціально обумовлених випадків.

4. Інформаційне забезпечення ІС має бути суміщене з інформаційним забезпеченням систем, які взаємодіють з нею, за змістом, системою кодування, методами адресації, форматами даних і формами подання інформації, яка отримується і видається інформаційною системою.

5. Форми документів, які створюються інформаційною системою, мають відповідати вимогам стандартів УСД чи нормативно-технічним документам замовника ІС.

6. Форми документів і відеокадрів, які вводяться чи коригуються через термінали ІС, мають бути погодженні з відповідними технічними характеристиками терміналів.

7. Сукупність інформаційних масивів ІС має бути організована у вигляді бази даних на машинних носіях.

8. Форми подання вихідної інформації ІС мають бути узгоджені із замовником (користувачем) системи.

9. Терміни і скорочення, які застосовуються у вихідних повідомленнях, мають бути загальноприйнятими в цій предметній області й погоджені із замовником системи.

10. У ІС мають бути передбачені необхідні заходи щодо контролю і оновлення даних в інформаційних масивах ІС, оновлення масивів після відмови будь-яких технічних засобів ІС, а також контролю ідентичності однойменної інформації в базах даних.

Можуть створюватись також самостійні інформаційні засоби і вироби для конкретного користувача.

Ефективне функціонування інформаційної системи об'єкта можливе лише при відповідній організації інформаційної бази – сукупності впорядкованої інформації, яка використовується при функціонуванні ІС і поділяється на зовнішньо- і внутрішньомашинну бази (ГОСТ 34.003-90) [2]. Зовнішньомашинна інформаційна база – частина інформаційної бази, яка являє собою сукупність повідомлень, сигналів і документів, призначених для безпосереднього сприйняття людиною без застосування засобів обчислювальної техніки. Внутрішньомашинна інформаційна база - частина інформаційної бази, що використовується в ІС на носіях даних.

Описуючи організацію інформаційної бази (РД 50-34.698-90), потрібно дати опис логічної і структурної бази даних [1].

Документ складається з двох частин:

1) опис внутрішньомашинної інформаційної бази;

2) опис зовнішньомашинної інформаційної бази.

Кожна частина складається з таких розділів:

1) логічна структура;

2) фізична структура (для зовнішньомашинної інф. бази);

3) організація ведення інформаційної бази.

При описі структури внутрішньомашинної інформаційної бази наводять перелік баз даних і масивів та логічні зв'язки між ними. Для масиву інформації вказують логічну структуру даних.

Описуючи структуру зовнішньомашинної інформаційної бази, наводять перелік документів та інших інформаційних повідомлень, використання яких передбачено в системі, із зазначенням автоматизованих функцій, при реалізації яких формується чи використовується цей документ.

Взаємозв'язок користувача з базою даних зображено на рис.3.1.

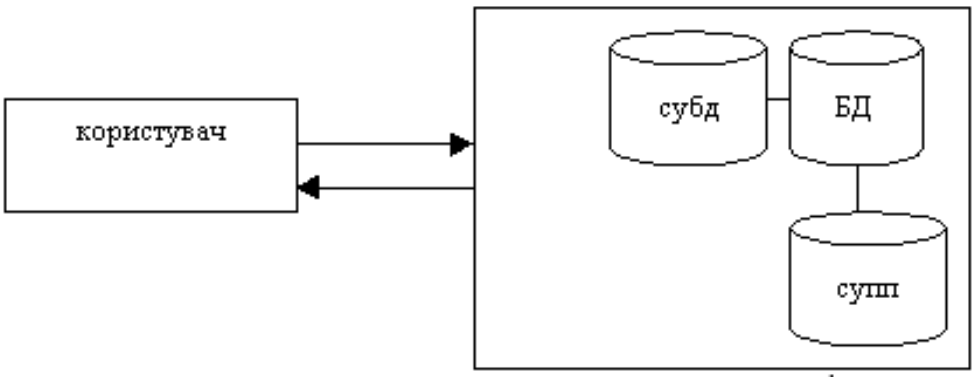


Рис. 3.1. Взаємозв'язок користувача з базою даних

Основною задачею є визначення потрібної кількості баз даних і оптимального розподілу інформації між ними з урахуванням того, що економічний об'єкт – це динамічна система, яка перебуває в постійному розвитку. Використовуючи пріоритет виробничих функцій, необхідно побудувати таку базу даних. Розглянувши поняття "Модель виробу", виділяємо дві оболонки: внутрішня являє конструкторську документацію, зовнішня - технологічну і управлінську інформацію (рис. 3.2).

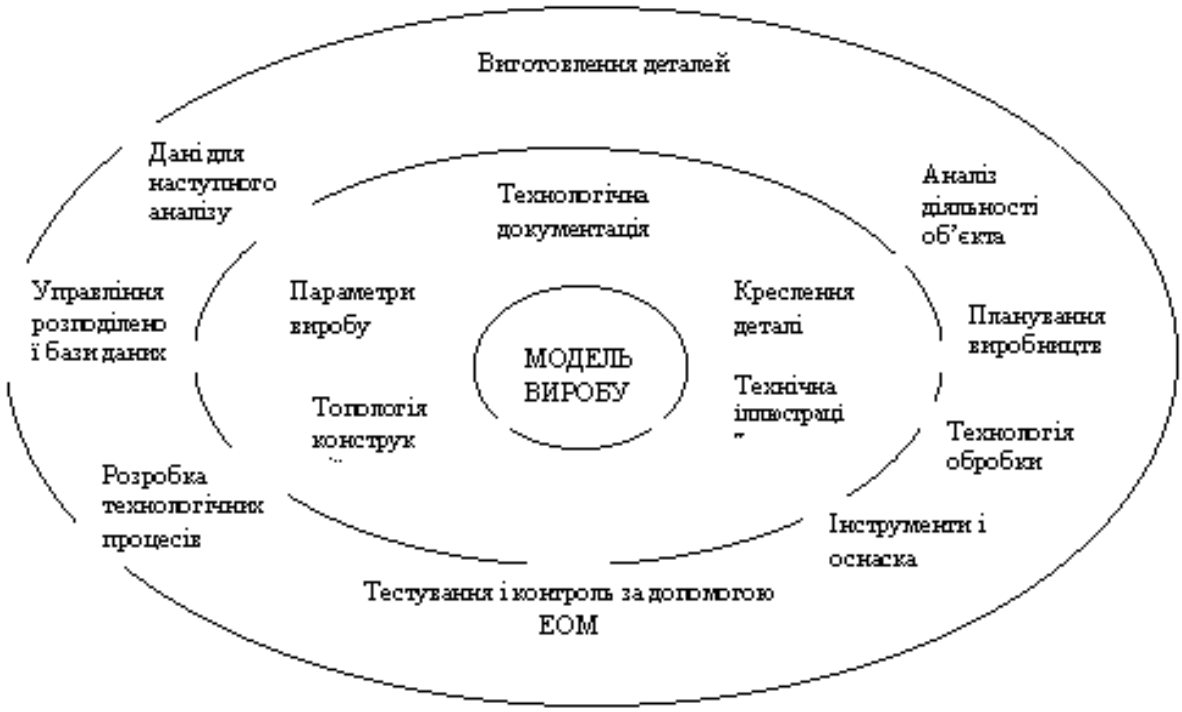


Рис. 3.2. Структура бази даних "Модель виробу"

Однак виникла така проблема: визначити, чи потрібна одна база даних, чи кілька локальних, або взаємозв'язана розподілена база даних, локальні файли чи їх комбінації і т.п. При цьому враховується інформація, що використовується для реалізації багатьох функцій, особливо в оперативному режимі, активна інформація, тобто така, що використовується багаторазово.

Розроблена в кваліфікаційній роботі інформаційна система оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів містить набір повязаних БД для різних функцій:

- створення нового замовлення (архітектура БД наведена у додатку А). Вона містить інформацію про замовника, доставку та установку ;
- вибір кухонних секцій (архітектура БД наведена у додатку Б). Таблиці БД містять інформацію про розміри, будову меблевого виробу, обрану фурнітуру;
- вибір параметрів матеріалу (архітектура БД наведена у додатку В).

3.2. Компоненти СУБД використані при розробці ІС

Компонент TQuery, як і компонент TTable, володіє всіма властивостями компоненту TDataSet. Як і у випадку з компонентом TTable, компонент TDataSource управляє взаємодією між компонентами Data Controls і компонентом TQuery. Звичайний додаток має один компонент DataSource для кожного компоненту TQuery [4].

Найчастіше використовуються наступні властивості компоненту TQuery [4]:

- Active – указує, відкритий (true) або закритий (false) даний запит;
- Eof, Bof – ці властивості приймають значення true, коли покажчик поточного запису розташований на останній або відповідно першому рядку набору даних, виконання запиту, що є результатом;
- DatabaseName – ім'я каталога або псевдонім (alias) видаленої БД, до якої здійснюється запит;
- DataSource – указує джерело даних для запитів, що параметризуються (тобто запитів з параметрами, значення яких заздалегідь невідоме);
- Fields – ця властивість доступна тільки під час виконання (run-time only) і використовується для читання або модифікації поля, визначуваного по порядковому номеру;
- Params – містить параметри для запиту, що параметризується, як SomeNo в наступному прикладі:

  Select * from Orders where CustNo=:SomeNo;

- SQL – рядковий масив, що містить текст оператора запиту SQL.

В створеній інформаційній системі використано компонент TQuery на окремій формі (рис.3.3), для зручності побудови коду програми.

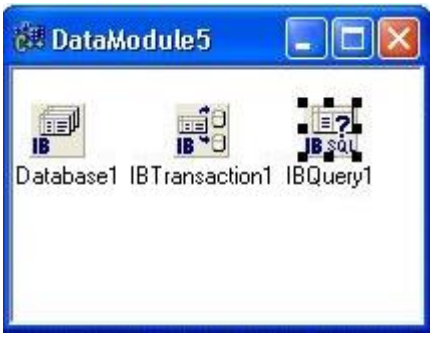


Рис. 3.3. Форма 5 з компонентом TQuery

Відзначимо, що мова запитів SQL (Structured Query Language), традиційно вживаний при роботі з серверними СУБД, може бути використана і при роботі з таблицями формату dBase і Paradox. Не вдаючись до докладного опису синтаксису цієї мови, відзначимо одну її особливість. SQL - мова непроцедурна. На ньому можна написати, що потрібно отримати в результаті запиту, але не можна написати, як це зробити, тобто не можна описати саму процедуру виконання запиту. Річ у тому, що реалізація виконання тих або інших операторів SQL серверами баз даних може бути різна, і в більшості випадків нецікава клієнтському застосуванню, що створюється за допомогою C++ Builder. У разі таблиць dBase або Paradox реалізацію SQL бере на себе бібліотека Borland Database Engine [1].

Компонент TQuery дозволяє використовувати операторів SQL для того, щоб визначати або створювати набори даних, які можна відобразити на екрані, вставляти, видаляти і редагувати рядки.

RequestLive - якщо ця властивість має значення true і синтаксис запиту такий, що його результат може модифікуватися, користувач може редагувати дані із збереженням їх в базі даних. Якщо RequestLive має значення false, результат запиту повертається в змозі read-only.

Найчастіше використовується метод EXECSQL компоненту TQuery. EXECSQL - виконує SQL-запит, що міститься у властивості SQL, якщо запит не повертає дані. Слід вживати цей метод при вставці, редагуванні або видаленні даних. При виконанні ж оператора SELECT (вибір даних) слід

використовувати метод Open. Наступний приклад показує застосування методу EXECSQL.

Лістинг 3.1

```
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add("Select * from
SECTION_PROPERTY where SECTION_TYPE_ID = 1");
DataModule5->IBQuery1->Open();
ComboBox29->Clear();
DataModule5->IBQuery1->First();
```

Запис DataModule5 виконує звернення до «форми 5» на якій розміщений компонент IBQuery1. Цим запитом ми заповнюємо параметри стандартних секцій на головній формі (див. додаток Д), тобто вказуємо поле таблиці для об’єкту ComboBox29, при натисканні на цей об’єкт та вибору коду верхніх секцій відповідно будуть відображатись габаритні розміри в полях Edit1, Edit2, Edit31. Таким самим методом вказуємо параметри і для нижніх секцій.

Лістинг 3.2

```
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add("Select * from
SECTION_PROPERTY where SECTION_TYPE_ID = 2");
DataModule5->IBQuery1->Open();
ComboBox32->Clear();
DataModule5->IBQuery1->First();
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add("Select * from
SECTION_PROPERTY where CODE = "+ComboBox29->Text);
DataModule5->IBQuery1->Open();
```

```
Edit1->Clear();
Edit31->Clear();
Edit2->Clear();
Edit1->Text = DataModule5->IBQuery1->
FieldByName("WIDTH")->AsString;
Edit2->Text = DataModule5->IBQuery1->
FieldByName("HEIGHT")->AsString;
Edit31->Text = DataModule5->IBQuery1->
FieldByName("DEPTH")->AsString;
```

Open – відкриває компонент TQuery. Він еквівалентний привласненню властивості Active значення true. Використовується, якщо результатом запиту є набір даних (такі запити зазвичай починаються з оператора SELECT). Приклад використання методу Open:

```
Query1->Open();
```

Close – закриває компонент TQuery. Виклик Close еквівалентний привласненню властивості Active значення false. Приклад використання методу Close:

```
Query1->Close();
```

Компоненти TQuery володіють великою різноманітністю методів, успадкованих від TDataSet. Найчастіше використовуються наступні методи:

- First, Last, Next, Prior –переміщають покажчик поточного запису на перший, останній, наступний і попередній записи відповідно, наприклад:
- MoveBy –переміщає покажчик на певну кількість рядків.
- Insert, Edit, Delete, Append, Post, Cancel – дозволяють модифікувати результат запиту.
- FreeBookmark, GetBookmark, GotoBookmark – дозволяють створювати закладки (маркіровані рядки) в запиті і потім повернутися до такого рядка пізніше.

Наприклад, метод Insert дозволяє вносити до результату запиту рядка, як в лістингу 3.3.

Лістинг 3.3

```
DataModule5->IBTransaction1->StartTransaction();
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add("Insert into
CONCRET_SECTION_LINK (SECTION_PROPERTY_CODE,
WIDTH, HEIGHT, DEPTH, ORDER_ID, SECTION_TYPE_ID)
values ('"+ComboBox29->Text+"','"+Edit1->Text+"','"+
Edit2->Text+"','"+Edit31->Text+"','"+Edit8->Text+"',1)");
DataModule5->IBQuery1->ExecSQL();
DataModule5->IBTransaction1->Commit();
```

Цим запитом вносимо дані про вибрані параметри секцій: код секції, ширина, висота, глибина, номер замовлення, і параметри вибраної секції (кількість кріплень, ручок, ножок, та різної фурнітури).

Метод Post підтверджує операції Insert, Update або Delete, здійснюючи реальну фізичну зміну в базі даних. Метод Cancel відміняє незавершені операції Insert, Delete, Edit або Append.

На формі в ІС також відображаються зображення вибраних нами секцій. Ця процедура здійснюється за допомогою об’єктів Image, їх розмісщено по 10 штук в кожному ScrollBox, і для кожного Image використовуємо аналогічний код:

Лістинг 3.4

```
{   Image5->Picture->Assign(jp);
    Image5->Invalidate();
    Label63->Caption = ComboBox29->Text;   }
```

Дані про зображення попередньо внесені при опису процедури натискання ComboBox29 та ComboBox32.

Лістинг 3.5

```
TMemoryStream *pMS = new TMemoryStream;
((TBlobField*)DataModule5->IBQuery1->
FieldByName("PICTURE"))->SaveToStream(pMS);
TJPEGImage *jp = new TJPEGImage();
pMS->Position = 0;
jp->LoadFromStream(pMS);
```

3.3. Доступний інтерфейс програми

Як і в будь-якій складній програмі бажано використовувати підказку, для того щоб користувачу було легко орієнтуватись в роботі з програмою. В ІС вона розміщена внизу головної форми – StatusBar1. Після або перед натисканням будь-якої кнопки відображається підказка про натупну або недопустиму дію програми, наприклад:

```
StatusBar1->SimpleText =
" Натисніть Далі щоб перейти до вибору матеріалів. ".
```

При не введенні всіх габаритних розмірів вибраної секції програма зчитує інформацію в полях Edit1, Edit2, Edit31, Edit3, Edit4, Edit26 у вигляді тексту та при наявності не заповненого поля видасть повідомлення "Оберіть параметри секцій." Це виконується за допомогою коду:

Лістинг 3.6

```
if   ((ComboBox29->Text == "") || (Edit1->Text == "") ||
      (Edit2->Text == "") || (Edit31->Text == ""))
{
     ShowMessage("Оберіть параметри секцій.");
}
```

На наступній вкладці головної форми відбувається вибір матеріалів з якого буде проводитись складання меблевої продукції. Щоб ці дані відображалися на формі за допомогою компонентів ComboBox, вказано з яких таблиць потрібно виводити ці дані. Пишемо код вибору поля таблиці Materials відповідно до кожного ComboBox. Наприклад для вибору товщини

ДСП плити, що цілком вагомо впливає на вартість продукції, використовуємо ComboBox1 (див. лістинг 3.7).

Лістинг 3.7

```
//заносимо дані про плиту ДСП
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add ("Select * from
MATERIALS where PROP_MATERIAL_LINK_ID = 2");
DataModule5->IBQuery1->Open();
ComboBox1->Clear();
DataModule5->IBQuery1->First();
while(!DataModule5->IBQuery1->Eof)
{
   ComboBox1->Items->Add(DataModule5->IBQuery1->
   FieldByName("PROPERTY_VALUE")->AsString);
   DataModule5->IBQuery1->Next();
}
DataModule5->IBQuery1->First();
```

В написанні коду програми я використовував підказки які розміщені перед початком основного коду (коментарі). Це відіграє важливу роль для орієнтації в великому наборі команд та процедур, також при написанні оновлених версій програми можна легко знайти код який нам потрібно вдосконалити.

При підрахунку вартості кухонної секції в програмі використовується унікальна формула обрахунку розмірів деталей з яких складається дана секція. Це – кришка, бочок, дно, полка, планка, цоколь і т.д. В цю формулу підставляються лише 3 вибрані габаритні розміри виробу, ширина – Width, висота – Height, глибина – Depth. Якщо формулу описати словами вона буде мати наступний вигляд:

Для обрахунку площі бочка на верхні секції:

2 *(Height * (Depth – 16 – 3)),

де 2 – це кількість бочків (в будь-якій секції їх 2, лівий, правий), 16 – товщина фасаду, 3 – товщина задньої стінки ДВП.

Площа полки:

(Width – (2 * X)) * Depth – 16 – 3 – 40,

де X – товщина ДСП плити з якої виготовляється меблева продукція (вона в нас вказується при виборі матеріалів та заноситься в таблицю CONCRET_SECTION_LINK бази даних), 40 – стандартний відступ полки від внутрішньої сторони фасаду.

Площа цоколя:

Width * 95,

де 95 – стандартна висота зйомного цоколя.

Підрахувавши площу всіх деталей ми її сумуємо та множими на вартість матеріалу. Отримаємо вартість використаного матеріалу. Якщо в замовленні не одна секція тоді ця формула послідовно підставляється до кожної секції обрахувавши таким чином весь розхід матеріалу. Мовою програмування та SQL запитів алгоритм обчислень буде виглядати так:

3. Спочатку відкриваємо дані про вибрані секції замовником

Лістинг 3.8

```
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add("Select
CONCRET_SECTION_LINK.WIDTH,
CONCRET_SECTION_LINK.HEIGHT,
CONCRET_SECTION_LINK.DEPTH,
SECTION_PROPERTY.polka_count from CONCRET_SECTION_LINK,
SECTION_PROPERTY where
(CONCRET_SECTION_LINK.ORDER_ID = "+Edit8->Text+")
and (CONCRET_SECTION_LINK.SECTION_TYPE_ID = 2)");
DataModule5->IBQuery1->SQL->
```

```
Add("(SECTION_PROPERTY.SECTION_TYPE_ID = 2) and
(SECTION_PROPERTY.CODE=
CONCRET_SECTION_LINK.SECTION_PROPERTY_CODE)");
DataModule5->IBQuery1->Open();
DataModule5->IBQuery1->First();
double dsp2=0, dvp2=0, kromka_d=0;
while(!DataModule5->IBQuery1->Eof)
```

4. Запускаємо формулу для підрахунку розходу матеріалу,

Лістинг 3.9

```
double bochok22=0, polka2=0, tsokol2=0, planka22=0, dno2=0;
bochok22 = 2*((DataModule5->IBQuery1->
FieldByName("DEPTH")->AsInteger)-40)*
((DataModule5->IBQuery1->FieldByName("HEIGHT")->
AsInteger)-100-StrToInt(ComboBox1->Text)-28);
polka2 = ((DataModule5->IBQuery1->FieldByName("WIDTH")-
>AsInteger)-(StrToInt(ComboBox1->Text)*2))*
((DataModule5->IBQuery1->FieldByName("DEPTH")->
AsInteger)-90)*(DataModule5->IBQuery1->FieldByName
("polka_count")->AsInteger);
  if (CheckBox14->Checked == true)
  {   tsokol2 = 95*(DataModule5->IBQuery1->
      FieldByName("WIDTH")->AsInteger);   }
planka22 = 2*(((DataModule5->IBQuery1->
FieldByName("WIDTH")->AsInteger)-
(StrToInt(ComboBox1->Text)*2))*(100));
dno2 = ((DataModule5->IBQuery1->FieldByName("DEPTH")-
>AsInteger)-40)*(DataModule5->IBQuery1->
FieldByName("WIDTH")->AsInteger);
dsp2 = dsp2 + RoundTo(((bochok22+polka2+
tsokol2+planka22+dno2)/1000000), -2);
```

```
dvp2 = dvp2 + RoundTo(((((DataModule5->IBQuery1->
FieldByName("HEIGHT")->AsFloat)-100-28-4)*
((DataModule5->IBQuery1->FieldByName("WIDTH")->
AsFloat)-4))/1000000), -2);
```

5. Всі отримані дані ми ділимо на 1000000, тому що розрахунок розмірів проводиться в міліметрах, а площа ДСП в метрах квадратних. Отримавши загальну площу ДСП плити множимо її на вартість, яку беремо з таблиці MATERIALS:

Лістинг 3.10

```
double vartistMat2=0;
vartistMat2 = (3.2*dsp2*(DataModule5->
IBQuery1->FieldByName("PRICE")->AsFloat));
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add
("Select PROPERTY_VALUE from MATERIALS where ID = 32");
DataModule5->IBQuery1->Open();
DataModule5->IBQuery1->First();
```

6. Після натисканні на кнопку Далі інформацію про вартість матеріалів виводимо на форму за допомогою Edit5:

```
Edit5->Text = FloatToStr(RoundTo(vartistMat2,0) +
RoundTo(vartistMat1,0) + RoundTo(vartistKrom,0));
```

Подібними формулами обраховуємо вартість фасадів та кромки. Всі обчислення сумуємо до Edit5. На формі також є поля для встановлення вартості доступних послуг: доставки, установки, комп’ютерного дизайну та виїзду на заміри (рис. 3.4).

Рис. 3.4.

При введенні даних у ці поля програма автоматично додає їх до Edit5, тобто до загальної вартості. При закінченні всіх обчислень програма заносить всі результати в базу даних прив’язуючи їх до номеру замовлення:

Лістинг 3.11

```
DataModule5->IBTransaction1->StartTransaction();
DataModule5->IBQuery1->Close();
DataModule5->IBQuery1->SQL->Clear();
DataModule5->IBQuery1->SQL->Add
("Update ORDERS set VARTIST="+Edit5->Text);
DataModule5->IBQuery1->SQL->Add(",AVANS="+Edit6->Text);
DataModule5->IBQuery1->SQL->Add(",ZALUSHOK="+Edit7->Text);
DataModule5->IBQuery1->SQL->Add
(",ZAMIRU_PRICE="+Edit29->Text);
DataModule5->IBQuery1->SQL->Add
(",INSTALL_PRICE="+Edit28->Text);
DataModule5->IBQuery1->SQL->Add
(",DESIGN_PRICE="+Edit30->Text);
DataModule5->IBQuery1->SQL->Add
(",VARTIST_MATERIALS="+Label22->Caption);
DataModule5->IBQuery1->SQL->Add(" where ID = "+Edit8->Text);
DataModule5->IBQuery1->ExecSQL();
DataModule5->IBTransaction1->Commit();
```

Наступним кроком виконуємо вибір фурнітури. Кріплення за допомогою якого здійснюється зборка секції використовується стандартне, незалежно від виду матеріалу. Кріплення розміщується так щоб ззовні не було помітно яким чином з’єднана дана деталь (додаток З). Наприклад конфірмати використовуються для з’єднання бочків з дном та кришкою, вставляються збоку та на видимих ділянках секції прикриваються заглушками. Задня стінка ДВП кріпиться шурупами 3х15, яких зазвичай не помітно. Кришки або стільниці нижніх секцій кріпляться мініфіксами, які врізаються в бочки зсередини та прикриваються заглушками ззовні. Всі ці кріплення пораховані відповідно до типів секцій та внесені в базу даних в таблицю SECTION_PROPERTY. В даній таблиці знаходиться вся інформація про секції, та редагується вона через форму 7.

Отже при виборі коду секції ми відразу вибрали певну кількість кріплення та маючи ціну – вираховуємо вартість конфірматів, шурупів, мініфіксів і т.д.

При переході на вкладку «Фурнітура» з бази даних заносяться дані про завіси, на яких будуть триматись фасади, направляючі, які використовуються для шухляд, ножки, ручки, ліфти. Замовник маючи уявлення про численні види меблевої фурнітури може сам впливати на загальну вартість всього замовлення, тому що на сьогоднішній день існує багато чисельна різноманітність фурнітури. Вся вона створена для різних побажань замовника.

Наприклад, звичайні направляючі виготовляються з тонкого металу пофарбованого недорогою фарбою та мають пластмасові ролики, які з часом зношуються. Направляючі марки Blum виготовляються з високоякісного металу, містять металеві ролики, а головне в них розміщений спеціальний механізм – це може бути сенсорний механізм, механізм з доводкою, механізм з електроприводом.

РОЗДІЛ 4
ОЦІНКА ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

В наш час відкрилися небачені у минулому столітті можливості для впровадження автоматизації виробничих процесів. Підвищення ефективності виробництва і управління в різних галузях людської діяльності значною мірою визначилось інтенсивністю розвитку програмних методів керування, використанням технічних засобів для обробки та зберігання інформації.

Інформаційна система є сукупністю організаційних, технічних, програмних і інформаційних засобів, з'єднаних в єдину систему з метою збору, зберігання, обробки і видачі необхідної інформації, призначена для виконання заданих функцій.

Мета створення інформаційних систем – у обмежено короткі терміни створити систему обробки і зберігання даних, яка має задані споживчі якості. До них належать: функціональна повнота, своєчасність, функціональна надійність та економічна ефективність.

Функціональна повнота – це властивість інформаційної системи, яка характеризує рівень автоматизації управлінських робіт.

Своєчасність є властивістю інформаційної системи, що характеризує можливість одержання апаратом керівництва необхідної інформації.

Функціональна надійність – це властивість інформаційної системи виконувати свої функції з обробки даних.

Економічна ефективність інформаційної системи виявляється в поліпшенні економічних результатів функціонування об’єкта внаслідок впровадження системи.

Інформаційну систему створюють у випадках, коли потрібно вдосконалити діючу методику й техніку розв’язання задач, впровадити нові задачі чи організувати нові обчислювальні центри.

Саме вдосконалення процесу управління створенням ПЗ стало основною метою реалізації інформаційної системи оптимального підбору витратних матеріалів та розрахунку вартості меблевих виробів. Будь-який замовник хоче одержати якісну систему (проєкт) за мінімальний час і з мінімальним вкладенням грошових коштів, що вимагає від розробника застосування ефективної методики організації процесу розробки.

Якщо проєкт настільки малий, що всю інформацію по ньому можна тримати в своїй голові, то навряд чи створювана система підвищить ефективність управління проєктом.

При створенні «Інформаційної системи» були дотримані наступні принципи: системності, розвитку (відкритості), сумісності, стандартизації та ефективності.

Принцип системності: при декомпозиції встановлені такі зв’язки між структурними елементами системи, які забезпечують цілісність системи та її взаємодію з іншими системами.

Принцип розвитку: виходячи з перспектив розвитку об’єкта автоматизації система створена з урахуванням можливості поповнення та оновлення та оновлення функцій і складу системи, не порушуючи її функціонування. Під об’єктом автоматизації розуміється процес управління створенням ПЗ.

Принцип сумісності: при розробці системи реалізовані інформаційні інтерфейси, завдяки яким вона може взаємодіяти з іншими системами за встановленими правилами.

Принцип стандартизації (уніфікації): при створенні системи раціонально використовуються уніфіковані елементи, проєктні рішення, пакети прикладних програм, компоненти.

Принцип ефективності: досягнення раціонального співвідношення між затратами і цільовими ефектами, включаючи кінцеві результати, одержані завдяки автоматизації.

Ефективність – це степінь сумарності результатів з витратами. Оптимізація – це покращення характеристик програмної системи чи просто програми. Тобто швидкоруч написана програма, яка довго, хоч і правильно працює, очевидно, неефективна. Вона підлягає оптимізації. З іншого боку, ефективна програма нікому не потрібна, якщо вона не забезпечує правильних результатів.

Отже, перший етап програмування – створення правильної програми, і лише другий – її оптимізація. Але перед тим, як починати покращувати ефективність програми, слід перевірити, наскільки це покращення буде корисним, і точно визначити місце, яке слід переробити.

Справа у тому, що існує правило 20/80: 20% об’єктного коду (тексту програми) виконується 80% часу роботи всієї програми. Деякі програми наукових обчислень дають навіть співвідношення 3/90.

Ця невелика частина програми, виконання якої займає більшу частину часу роботи програми, називається критичною областю. Саме критичну область і слід оптимізувати.

У вартості процеса програмування переважну частину складає вартість людської праці, тому при визначенні можливого покращення треба оцінювати обсяг роботи, необхідної для досягнення цього покращення.

Багато засобів, які підвищують ефективність програми, не погіршують її зручність для читання, отже, мають використовуватись завжди. В усіх інших випадках слід дотримуватись правила: зручність для читання програми важливіша за її ефективність.

Оптимізація некритичних областей програми майже завжди буде марною витратою часу.

ВИСНОВКИ

Інформаційна система розроблена в кваліфікаційній роботі дозволяє в мінімальні терміни обрахувати вартість кухонних секцій, змінюючи їхні габаритні розміри, також до загальної вартості враховується ціна фасаду, який на сьогоднішній день має дуже великий різновид, ціна стільниці, різної фурнітури, яка використовується при виготовленні кухні, і яка відіграє важливу роль в загальній вартості кухні. Програма забезпечує функції роботи з базами даних матеріалів, фурнітури, аксесуарів, і головне клієнтів з детальним описом про замовлення (який матеріал використався, колір, товщина, різновид фурнітури, вартість фасадів і т.д.).

Інформаційна система призначена для невеликих і середніх підприємств, які займаються виготовленням меблевої продукції, а також для магазинів які реалізують цю продукцію. Вона буде особливо корисною для розробників проєктів мебелі в поєднанні з іншими спеціалізованими програмами.

Розроблена у даній дипломній роботі інформаційна система має сучасну архітектуру методів роботи з БД. Розглянуті основні можливості програмування баз даних в Visual C++, зокрема використовуючи ODBC та DAO, а також можливості візуального конструювання баз даних. Обмін даними між сервером додатків та клієнтським додатком ґрунтується на протоколі XML. Здійснено спробу створити клієнтську частину, яка динамічно формує користувацький інтерфейс незалежно від структури БД. Побудовано формат структури даних, прийнятної для користувацького інтерфейсу динамічної побудови.

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ІС – інформаційна система

SQL – Structured Query Language – структурована мова запитів

XML (eXtensible Markup Language) – мова розширеної розмітки

СУБД – система управління базами даних

ПЗ – програмне забезпечення

БД – база даних

ЕОМ – електронно-обчислювальна машина

ПК – персональний комп’ютер

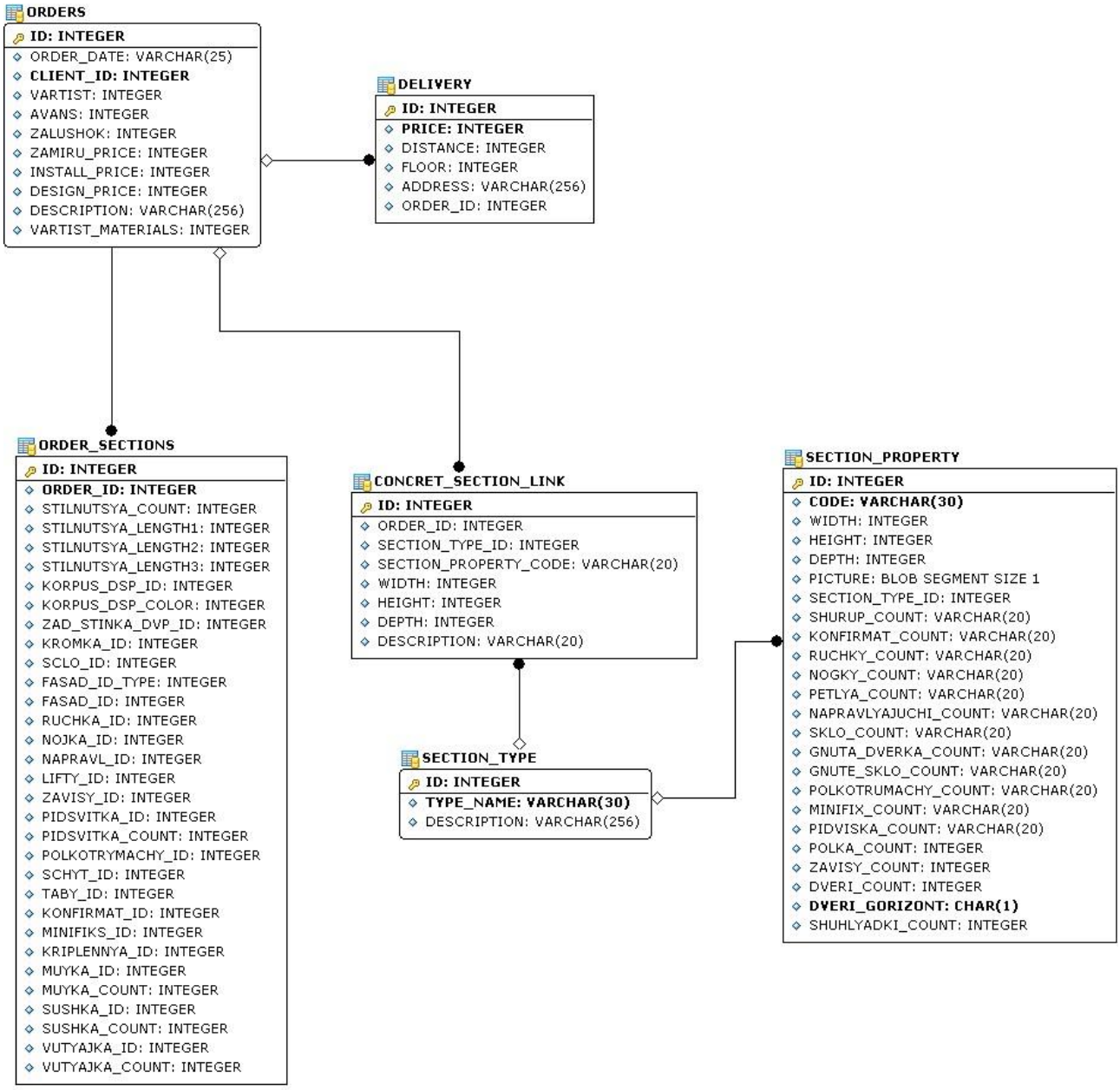
СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Анісімов А.В. Кулябко П.П. Інформаційні системи та бази даних: навчальний посібник для студентів факультету комп'ютерних наук та кібернетики. Київ, 2017. 110 с.
2. Гомонай-Стрижко М.В., Якімцов В.В. Інформаційні системи та технології на підприємстві: конспект лекцій. Львів: НЛТУ, 2014. 200 с.
3. Добровольська Л. О., Черевко О. О. Інформаційні системи в промисловості : навчальний посібник. Маріуполь : ПДТУ, 2014. 238 с.
4. Савчук Т.О. Організація баз даних і знань. Вінниця: ВДТУ, 2017 р. 150с.
5. Морзе Н.В., Піх О.З. Інформаційні системи : навчальний посібник. Івано-Франківськ: «ЛілеяНВ», 2015. 384 с
6. WinABePC. Програмне забезпечення для майстрів. URL: http://www.1c.ru/vendors/winavers/page2_01_10_01_.html (дата звернення 18.05.2023)
7. Вікіпедія.CVS. URL: http://ru.wikipedia.org/wiki/CVS (дата звернення 18.05.2023)
8. Технологія Клієнт-Сервер. URL: http://www.optim.ru/ (дата звернення 18.05.2023)
9. Dotmarketing. URL: http://dotmarketing.org/pages/ (дата звернення 18.05.2023)
10. Meblevyk. URL: http://www.meblevyk.info/content/detail/373/print
11. Koljmya. URL: http://koljmya.at.ua/news/pro_100_v4_42/2010-03-17-194 (дата звернення 18.05.2023)
12. Коновалов В.С., Радоуцький К.Є. Сучасні принципи і методи проектування програмного забезпечення: Конспект лекцій. Ч. 2. Харків: УкрДАЗТ, 2015. 109 с.
13. Соммервилл Иан. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. М.: Издательский дом "Вильямс", 2002. 624 с.

ДОДАТКИ

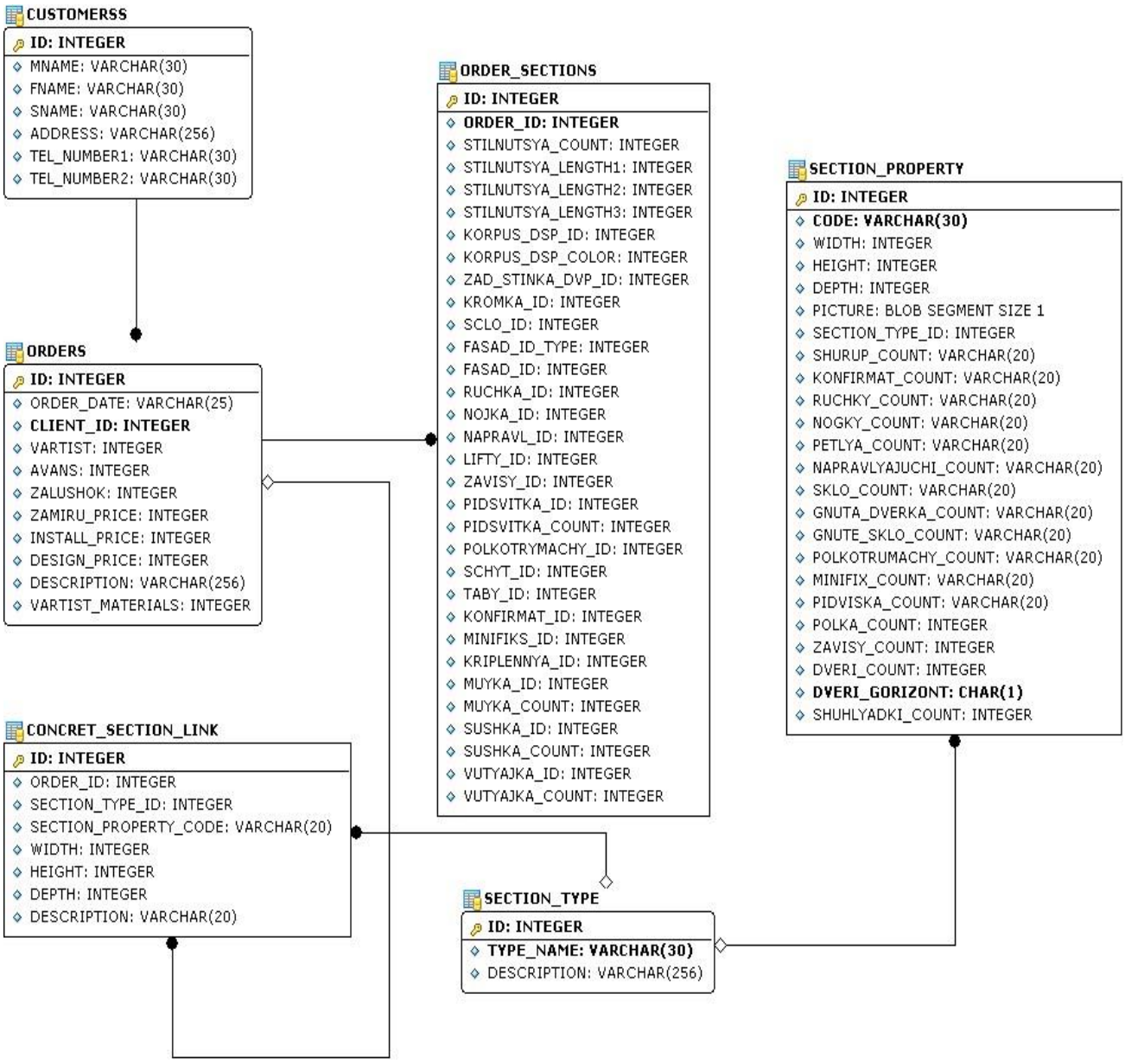
Додаток А

Архітектура бази даних – створення нового замовлення



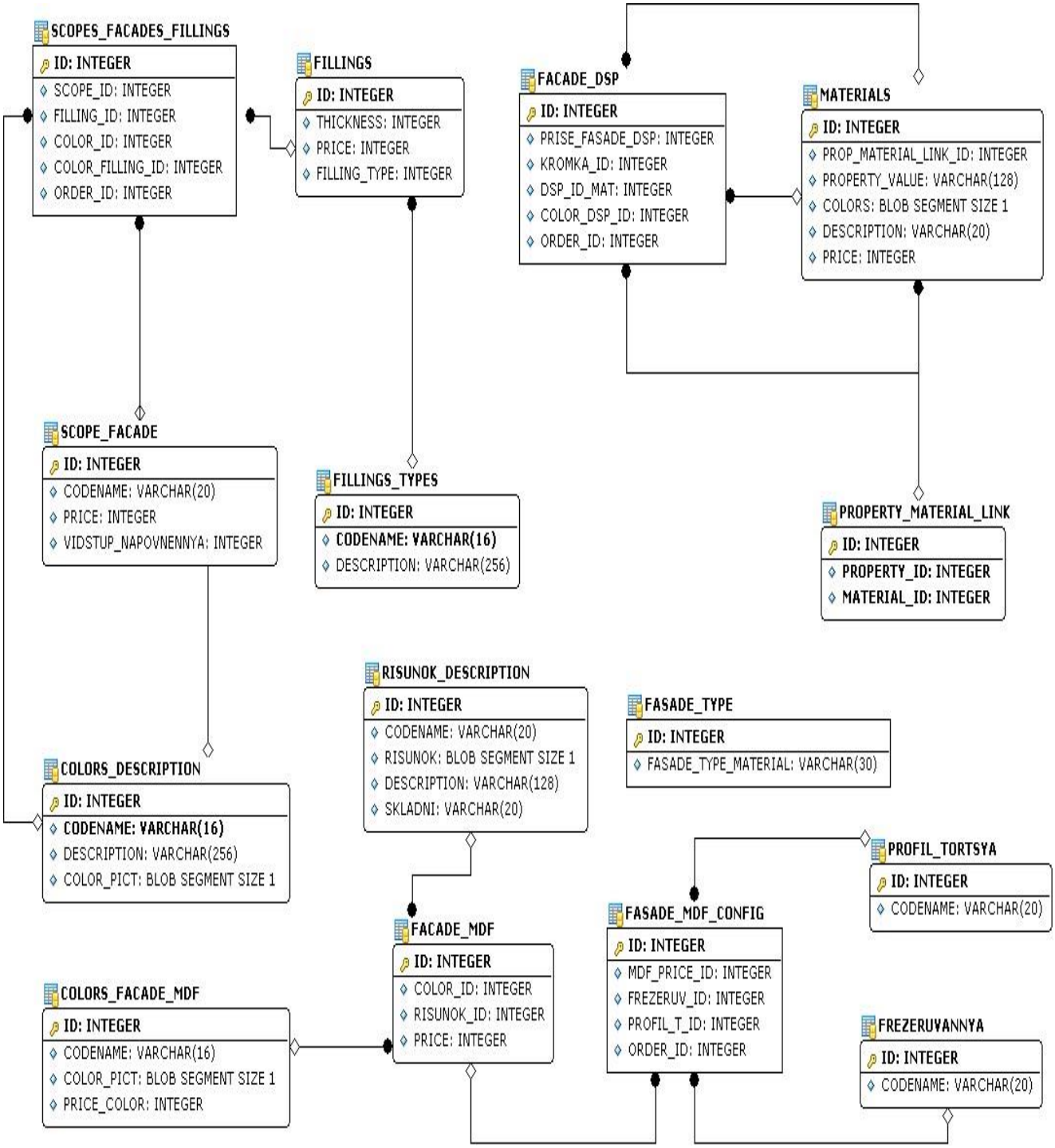
Додаток Б

Архітектура бази даних – вибір кухонних секцій



Додаток В

Архітектура бази даних – вибір параметрів матеріалу

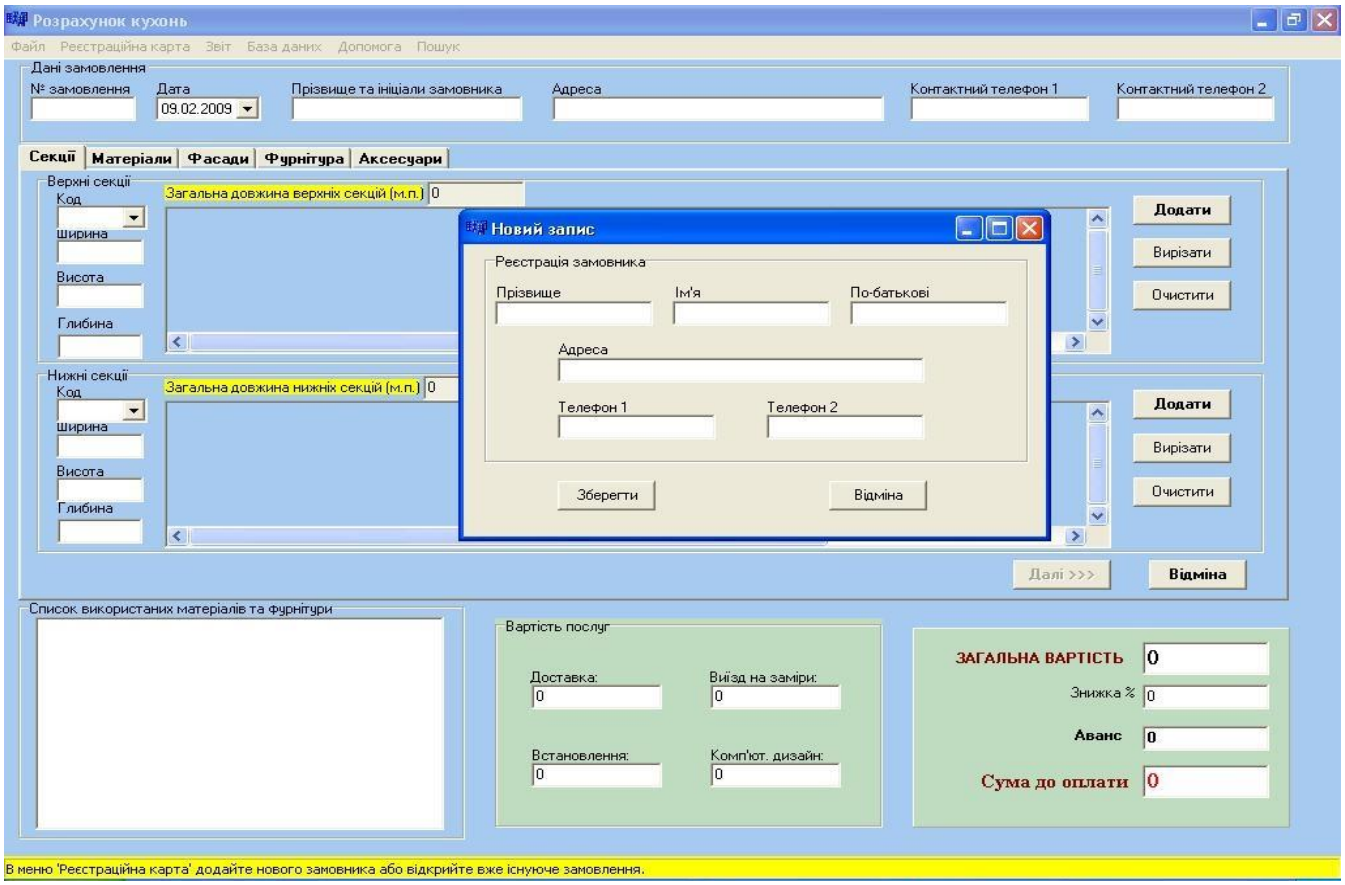


Додаток Г

Підключення до сервера бази даних

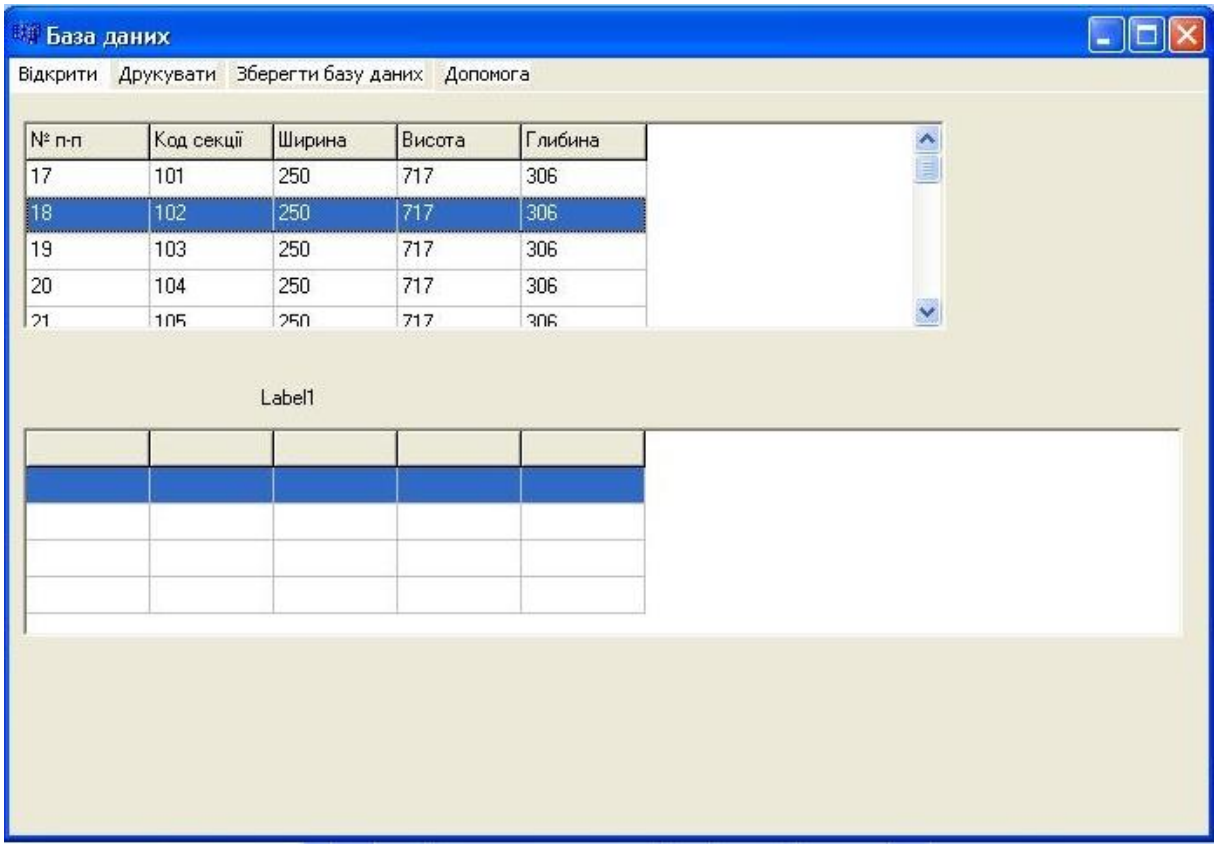
Додаток Д

Головна форма програми



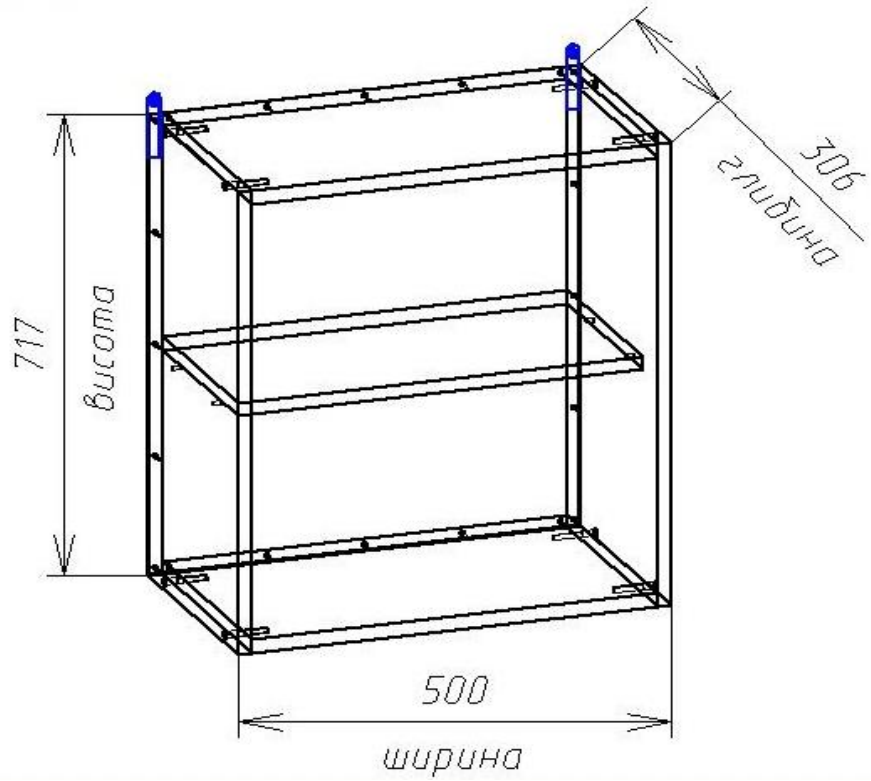
Додаток Ж

Редагування бази даних



Додаток З

Стандартні розміри секцій



Додаток К

Зображення стандартної секції з МДФ фасадом