Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

_____Інститут електроенергетики_____
(інститут)
_____факультет інформаційних технологій_____
(факультет)
Кафедра__інформаційних систем та технологій та комп'ютерної інженерії___
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА

кваліфікаційної роботи ступеня ____бакалавра___
(бакалавра, спеціаліста, магістра)

Студента_____Проценко Кирило Віталійовича_____
(ПІБ)
академічної групи_____126-17-1_____
(шифр)
спеціальності _____126 « Інформаційні системи та технології»____
(код і назва спеціальності)
за освітньо-професійною програмою
_____«Інформаційні системи та технології»_____

(офіційна назва)
на тему____Розробка інформаційної системи підприємства "ІСТА"_____

(назва за наказом ректора)

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	Проф.Гнатушенко В.В			
розділів:				

Рецензент	доц. Хом'як Т.В			

Нормоконтролер	проф. Коротенко Г.М..			

ДНІПРО

2021

ЗАТВЕРДЖЕНО:
завідувач кафедри
___інформаційних технологій___
___та комп'ютерної інженерії____
(повна назва)
_____ __Гнатушенко В.В.__
(підпис) (прізвище, ініціали)

«_____»_____ 2021 року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня ____бакалавр _____
(бакалавра, спеціаліста, магістра)

студенту _____Проценко К.В._____ **академічної групи** ____126-17-1___
(прізвище та ініціали) (шифр)

спеціальності ___126 « Інформаційні системи та технології »_____

за освітньою-професійною програмою _____

_____« Інформаційні системи та технології»_____
на тему____Розробка інформаційної системи підприємства "ІСТА"_____

затверджену наказом ректора НТУ «Дніпровська політехніка» від 07.06.2021р. № 317-с

Розділ	**Зміст**	**Термін виконання**
1 Аналіз програмних засобів для проектування систем	На основі матеріалів з відповідних джерел та даних проаналізувати засоби для проектування систем	18.03.2021 р.
Розділ 2. Проектна частина	На основі матеріалів виробничої практики і інших науково-технічних джерел проаналізувати методи і засоби по створенню ІС та створити відповідну розробку	23.06.2021 р.

Завдання видано _____ _____
(підпис керівника) (прізвище, ініціали)

Дата видачі _____

Дата подання до екзаменаційної комісії ___23.06.2021 р._____

Прийнято до виконання _____ _____Проценко К.В_____
(підпис студента) (прізвище, ініціали)

ДНІПРО

2021

РЕФЕРАТ

Пояснювальна записка містить 75 сторінок, 38 малюнків, 11 джерел

Об’єкт розробки: Проектування інформаційної системи ІСТА

Мета дипломної роботи: створення бази даних для інформаційної системи ІСТА

У вступі вказана мета дипломної роботи, обгрунтована її актуальність, конкретизовані завдання, які необхідно вирішити.

У першому розділі пояснювальної записки наведено опис характеристик та використання програм BPwin та ERwin. Характеристика та використання SQL Server 2008 R2 в яку входять ключи, визначення атрибутів, нормалізація, цілісність даних. Характеристика програмного забезпечення Delphi 7 та постановка задачі.

У другому розділі написано структуру інформаційної системи

Практична значимість даного проекту полягає в можливості його використання для підвищення рівня інформаційної системи ІСТА

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ПІДПРИЄМСТВА «ІСТА», БАЗА ДАНИХ, МОДЕЛІ ДАНИХ, BPWIN, ERWIN, SQL SERVER 2008 R2 , DELPHI 7, POWERSHELL

ABSTRACT

The explanatory note contains 75 pages, 38 figures, 11 sources

Object of development: Design of the ISTA information system

The purpose of the thesis: to create a database for the information system ISTA

The introduction indicates the purpose of the thesis, substantiates its relevance, specifies the tasks to be solved.

The first section of the explanatory note describes the characteristics and use of BPwin and ERwin. Characteristics and use of SQL Server 2008 R2 which includes keys, attribute definition, normalization, data integrity. Characteristics of Delphi 7 software and problem statement.

The second section describes the structure of the information system

The practical significance of this project lies in the possibility of its use to increase the level of the ISTA information system

INFORMATION SYSTEM FOR “ISTA ENTERPRISE”, DATABASE, DATA MODELS, BPWIN, ERWIN, SQL SERVER 2008 R2, DELPHI 7, POWERSHELL

ЗМІСТ

ВСТУП 8

РОЗДІЛ 1 9

АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМ 9

1.1 Характеристика та використання AllFusion Process Modeler (BPwin) 9

1.2 Характеристика та використання AllFusion Data Modeler (ERwin). 15

1.3.Характеристика SQL Server 2008 R2 23

1.3.1 Визначення атрибутів 23

1.3.2 Ключі 24

1.3.3 Нормалізація 25

1.3.4 Цілісність даних 28

1.4 Характеристика програмного забезпечення Delphi 7 30

1.5 Постановка задачі 36

РОЗДІЛ 2 37

РОЗРОБКА МОДЕЛІ ДАНИХ 37

2.1 Сценарії діяльності підприємства. 37

2.2 Розробка діаграм IDEF0 і DFD. 39

2.2.1 Побудова діаграми IDEF0 43

2.2.2.Методологія діаграми DFD 44

2.3 Розробка моделі даних ERwin 50

2.3.1. Створення логічної моделі даних 50

2.3.2 Фізична модель даних 53

2.4 Ініціалізація SQL - скрипта 54

РОЗДІЛ 3 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ 56

3.1. Характеристики серверу 56

3.2. Призначення і умови застосування вибранного серверу 57

3.3 Підключення та створення БД на SQL сервері 60

4.1 Розробка додатка клієнта та тестування системи 66

ВИСНОВКИ 73
СПИСОК ПОСИЛАНЬ 74
ДОДАТОК А 76
ДОДАТОК Б 77
ДОДАТОК В 78
ДОДАТОК Г 79
ДОДАТОК Д 86
ДОДАТОК Е 109
ДОДАТОК Є 110

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ІС – інформаційна система

BPwin – AllFusion Process Modeler

Erwin – AllFusion Data Modeler

СУБД – Система управління базами даних

ВСТУП

Найбільшу частину даних, представлених в сучасних системах організації діяльності підприємства, в даний час складає аналіз цієї діяльності.

Останнім часом для цілей аналізу підприємств все більше поширення отримує засіб моделювання в Bpwin та ERwin компанії Computer Associates для представлення бізнес-процесів. Даний BPwin - засіб допомагає чітко документувати важливі аспекти будь-яких бізнес-процесів, в першу чергу для платформи Windows. BPwin дуже гнучкий і розширюваний, що дозволяє справлятися з моделюванням складних бізнес-процесів.

Моделювання в Erwin проводиться від логічної моделі до фізичної моделі системи даних.

Клієнтська частина додатка може створюватися як за допомогою спеціалізованих програмних середовищ, що поставляються спільно з сервером, так і за допомогою звичайних засобів розробки програм, таких як Delphi, який надає досконалий інструментарій для розробників програмного забезпечення і баз даних. За допомогою потужного інтегрованого середовища розробки, візуального редактора інтерфейсу, обширного набору компонентів стало можливим створювати програмні задачі набагато швидше і ефективніше.

Метою роботи є проектування інформаційної системи підприємства у середовищі Bpwin, Erwin + SQL Server та розробка додатку клієнта для кінцевого користувача, за допомогою програмного продукту Delphi.

РОЗДІЛ 1
АНАЛІЗ ПРОГРАМНИХ ЗАСОБІВ ДЛЯ ПРОЕКТУВАННЯ СИСТЕМ

1.1 Характеристика та використання AllFusion Process Modeler (BPwin)

На початкових етапах створення ІС необхідно зрозуміти, як працює організація, яку потрібно автоматизувати.Така модель повинна містити дані предметної області, отже, вона повинна містити в собі знання всіх учасників бізнес-процесів організації[1]

Найбільш зручною мовою моделювання бізнес-процесів IDEF0, запропонований більше 20 років тому Дугласом Россом (SoftTech, Inc.) і називався спочатку SADT - Structured Analysis and Design Technique. На початку 70-х років збройні сили США застосували підмножина SADT щодо моделювання процесів, для реалізації проектів в рамках програми ICAM (Integrated Computer-Aided Manufacturing). Надалі це підмножина SADT було прийнято в якості федерального стандарту США під найменуванням IDEF0.

В IDEF0 система представляється як сукупність взаємодіючих робіт або функцій. Така функціональна орієнтація є принциповою - функції системи аналізуються незалежно від об'єктів, якими вони оперують. Це дозволяє більш чітко змоделювати логіку і взаємодію процесів організації. Під моделлю в IDEF0 розуміють опис системи (текстове і графічне), яке має дати відповідь на деякі наперед визначені запитання.

Модельована система розглядається як довільна підмножина підприємства. Довільне тому, що, по-перше, ми самі визначаємо, чи буде якийсь об'єкт компонентом системи, або ми будемо його розглядати як зовнішній вплив, і, по-друге, воно залежить від точки зору на систему. Система має межу, яка відділяє її від решти. Взаємодія системи з навколишнім світом описується як вхід (щось, що переробляється системою), вихід (результат діяльності системи), управління (стратегії та процедури, під

управлінням яких проводиться робота) і механізм (ресурси, необхідні для проведення роботи). Перебуваючи під управлінням, система перетворює входи у виходи, використовуючи механізми.

Процес моделювання будь-якої системи в IDEF0 починається з визначення контексту, тобто найбільш абстрактного рівня опису системи в цілому. У контекст входить визначення суб'єкта моделювання, цілі і точки зору на модель.

Під суб'єктом розуміється сама система, при цьому необхідно точно встановити, що входить в систему, а що лежить за її межами, іншими словами, ми повинні визначити, що ми будемо надалі розглядати як компоненти системи, а що як зовнішній вплив. На визначення суб'єкта системи буде істотно впливати позиція, з якої розглядається система, і мета моделювання - питання, на які побудована модель повинна дати відповідь. Іншими словами, спочатку необхідно визначити область (Scope) моделювання.

Опис галузі як системи в цілому, так і її компонентів є основою побудови моделі. Хоча передбачається, що протягом моделювання область може коригуватися, вона повинна бути сформульована спочатку, оскільки саме область визначає напрямок моделювання й коли мусить бути закінчена модель. При формулюванні області необхідно враховувати два компоненти - широту і глибину. Широта передбачає визначення меж моделі, ми визначаємо, що буде розглядатися всередині системи, а що зовні. Глибина визначає, на якому рівні деталізації модель є завершеною. При визначенні глибини системи необхідно не забувати про обмеження часу - трудомісткість побудови моделі зростає в геометричній прогресії від глибини декомпозиції. Після визначення меж моделі передбачається, що нові об'єкти не повинні вноситися в модельовану систему; оскільки всі об'єкти моделі взаємопов'язані, внесення нового об'єкта може бути не просто арифметичної добавкою, але в змозі змінити існуючі взаємозв'язки. Внесення таких змін в

готову модель є, як правило, дуже трудомістким процесом (так звана проблема "плаваючої області").

Формулювання мети дозволяє команді аналітиків сфокусувати зусилля в потрібному напрямку. Прикладами формулювання цілі можуть бути наступні твердження: "Ідентифікувати і визначити поточні проблеми, зробити можливим аналіз потенційних поліпшень", "Ідентифікувати ролі і відповідальності службовців для написання посадових інструкцій", "Описати функціональність підприємства з метою написання специфікацій інформаційної системи" і т. д.
Точка зору (Viewpoint). Хоча при побудові моделі враховуються думки різних людей, модель повинна будуватися з єдиної точки зору. Точку зору можна представити як погляд людини, яка бачить систему в потрібному для моделювання аспекті. Точка зору повинна відповідати мети моделювання.

Очевидно, що опис роботи підприємства з точки зору фінансиста і технолога буде виглядати зовсім по-різному, тому протягом моделювання важливо залишатися на вибраній точці зору. Як правило, вибирається точка зору людини, відповідального за модельовану роботу в цілому. Часто при виборі точки зору на модель важливо задокументувати додаткові альтернативні точки зору. Для цієї мети зазвичай використовують діаграми FEO (For Exposition Only), які будуть описані в подальшому.

IDEF0[2] - це модель, яка передбачає наявність чітко сформульованої мети, єдиного суб'єкта моделювання однієї точки зору. Для внесення області, мети і точки зору моделі IDEF0 в BPwin слід вибрати пункт меню Edit/Model Properties, що викликає діалог Model Properties. В закладці Purpose слід внести мета і точку зору, а в закладку Definition - визначення моделі і опис галузі. Нижче представлений приклад декомпозиції на рис.1.1.

В закладці Status того ж діалогу можна описати статус моделі (чорновий варіант, робочий, остаточний і т. д.), час створення і останнього редагування (відстежується в подальшому автоматично за системну дату).

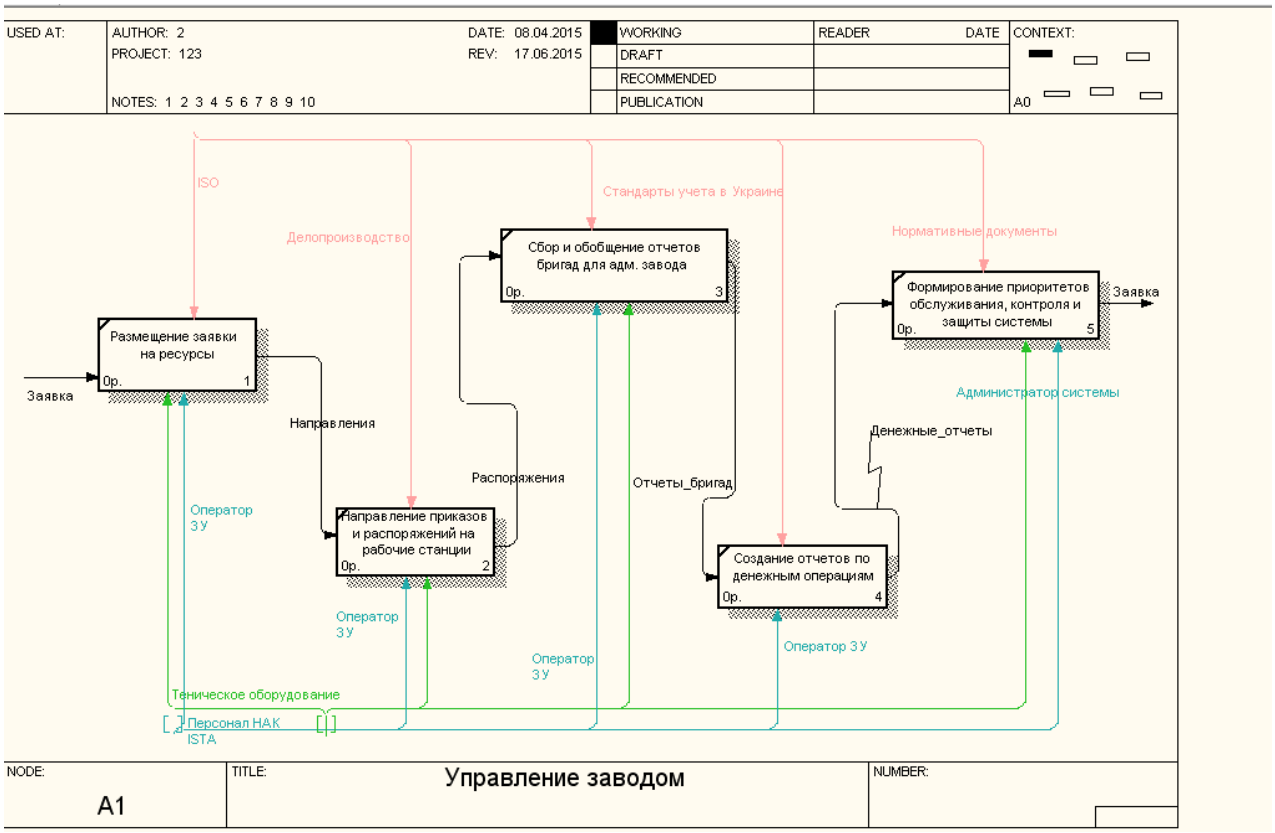


Рис.1.1 – Приклад діаграми IDEF0

В закладці Source описуються джерела інформації для побудови моделі (наприклад, "Опитування експертів предметної області та аналіз документації"). Закладка General служить для внесення імені проекту та моделі, імені та ініціалів автора і тимчасових рамок моделі AS-IS і TO-BE.[1]

Технологія проектування ІС передбачає спочатку створення моделі AS-IS, її аналіз і поліпшення бізнес-процесів, тобто створення моделі TO-BE, і тільки на основі моделі TO-BE будується модель даних, прототип і потім остаточний варіант ІС. Побудова системи на основі моделі AS-IS призводить до автоматизації підприємства за принципом "все залишити як є, тільки щоб комп'ютери стояли", тобто ІС автоматизує недосконалі бізнес-процеси і дублює, а не замінює існуючий документообіг. У результаті впровадження і експлуатація такої системи призводить лише до додаткових витрат на закупівлю обладнання, створення програмного забезпечення та супровід того й іншого.

Іноді поточна AS-IS і майбутня TO-BE моделі різняться дуже сильно, так що перехід від початкового до кінцевого стану стає неочевидним. В цьому випадку необхідна третя модель, що описує процес переходу від початкового до кінцевого стану системи, оскільки такий перехід - це теж бізнес-процес.

Результат опису моделі можна отримати в звіті Model Report. Діалог налаштування звіту по моделі викликається з пункту меню Report/Report Model.

У діалозі налаштування слід вибрати необхідні поля, при цьому автоматично відображається черговість виведення інформації в звіт. В реальних схемах до кожної роботи може підходити і від кожної може відходити близько десятка стрілок. Якщо діаграма містить 6-8 робіт, то вона може містити 30-40 стрілок, причому вони можуть зливатися, розгалужуватися і припинятися. Такі діаграми можуть стати дуже погано читаються. В IDEF0 існують угоди з малювання діаграм, які покликані полегшити читання та експертизу моделі. Деякі з цих правил BPwin підтримує автоматично, виконання інших слід забезпечити вручну.

Прямокутники робіт повинні розташовуватися по діагоналі з верхнього лівого у правий нижній кут (порядок домінування). Далі можна додати нові роботи або змінити розташування існуючих, але порушувати діагональне розташування робіт по можливості не слід. Порядок домінування підкреслює взаємозв'язок робіт, дозволяє мінімізувати вигини і перетину стрілок.

Слід максимально збільшувати відстань між вхідними або вихідними стрілками на одній грані роботи. Якщо включити опцію Line Drawing: Automatically space arrows на закладці Layout діалогу Model Properties (меню Edit/Model Properties), BPwin буде розташовувати стрілки належним чином автоматично. Слід максимально збільшити відстань між роботами, поворотами і перетинами стрілок.

Якщо дві стрілки проходять паралельно (починаються з однієї і тієї ж межі однієї роботи і закінчуються на одній і тій же грані іншої роботи), то по можливості слід їх об'єднати і назвати єдиним терміном.

Циклічні зворотні зв'язки слід малювати тільки в разі крайньої необхідності, коли підкреслюють значення повторно використовуваного об'єкта. Прийнято зображати такі зв'язки на діаграмі декомпозиції. BPwin не дозволяє створити циклічно-зворотній зв'язок за один прийом. Якщо все ж необхідно зобразити такий зв'язок, слід спочатку створити звичайну зв'язок по входу, потім розгалужувати стрілку, направити нову гілку назад до входу роботи-джерела і, нарешті, видалити стару гілку стрілки виходу.

Інтуїтивно-зрозумілий графічний інтерфейс, який швидко і легко освоюється, що дозволяє зосередитися на аналізі самої предметної області, не відволікаючись на вивчення інструментальних засобів. Інтерактивне виділення об'єктів забезпечує постійний візуальний зворотний зв'язок при побудові моделі. BPwin підтримує посилальну цілісність, не допускаючи визначення некоректних зв'язків і гарантуючи несуперечливість відносин між об'єктами при моделюванні.

Автоматизація процесу проектування. BPwin автоматизує багато завдань, зазвичай пов'язані з побудовою моделей процесів, забезпечуючи семантичну точність, необхідну для гарантії правильних і узгоджених результатів. Підсвічування об'єктів спрощує побудову моделі, виключаючи часто зустрічаються помилки моделювання.

Властивості, що визначаються користувачем. Ви можете налаштувати BPwin для збору інформації, істотною для вашого бізнесу. Ця інформація одразу ж стає доступною через генератор звітів BPwin і може бути експортована в інші програми, наприклад, Microsoft Word і Excel.

Контекстні діаграми для опису меж системи, області дії, призначення об'єктів. Ієрархічна структура графіків, що полегшує послідовне уточнення елементів моделі. Декомпозиційні діаграми для опису особливостей

взаємодії різних процесів. BPwin підтримує автоматичну настройку розмірів діаграм і можливість зміни масштабу зображення моделей.

Організаційні діаграми. Організаційні структури впливають на визначення і виконання бізнес-процесів. BPwin підтримує явне визначення ролей, а це визначає та впорядковує задачі або роботи, складові бізнес-процеси. Ґрунтуючись на ролях, визначених користувачем, BPwin формує організаційні діаграми.

Технології моделювання. BPwin забезпечує спільне й повторне використання технологій моделювання бізнес-процесів (IDEF0), потоків робіт (IDEF3) і потоків даних (DFD).

Функціонально-вартісний аналіз BPwin повністю підтримує методи розрахунку собівартості за обсягом господарської діяльності і оптимізований для аналізу процесів. Розвинені засоби підготовки звітів і двонаправлений інтерфейс зі спеціалізованим інструментарієм, полегшує реалізацію корпоративної стратегії на основі управління господарською діяльністю.

Власний генератор звітів. Report Template Builder (RTB) - це новий генератор звітів, загальний для ERwin і BPwin, створює різноманітні звіти та Web-сторінки. Ви можете визначати шаблони звітів, застосовуючи їх до будь-яким своїм моделям. Підхід "визначити одного разу - застосовувати повторно і всюди" дозволяє організації швидко створювати і просувати стандарти звітності. RTB підтримує безліч форматів, включаючи RTF, HTML, XLS (Excel) і звичайний текст.[2]

1.2 Характеристика та використання AllFusion Data Modeler (ERwin).

Для проектування інформаційних систем спочатку використовувалася програма AllFusion Process Modeler(BPwin). AllFusion Process Modeler(ERwin) в рамках одного інструмента забезпечує функціональність моделювання бізнес-процесів, потоків процесів і потоків даних, що враховує потреби як бізнес-аналітиків, так і технологів. На відміну від простих

додатків для створення діаграм, Erwin[3] надає доступ до деталізації, що прив'язує операції моделювання бізнес-процесів до потоків даних або до потоків робіт. Крім того, кожний об'єкт моделі може мати свої власні властивості, включаючи певні користувачем, і відносяться до особливостей конкретного бізнесу. Erwin також може прив'язувати конкретні операції до різних ролей або підрозділам самої організації, надаючи вам повне розуміння не тільки того, що зроблено, але й ким зроблено, і коли.

Зазвичай розробка моделі бази даних складається з двох етапів: складання логічної моделі і створення на її основі фізичної моделі. ERwin повністю підтримує такий процес, він має два представлення моделі: логічний (logical) і фізичний (physical). Таким чином, розробник може будувати логічну модель бази даних, не замислюючись над деталями фізичної реалізації, тобто приділяючи основну увагу вимогам до інформації та бізнес-процесів, які буде підтримувати майбутня база даних. ERwin має дуже зручний інтерфейс, що дозволяє представити базу даних в самих різних аспектах. Наприклад, ERwin має такі засоби візуалізації як "збережене уявлення" (stored display) і "предметна область" (subject area). Збережені подання дозволяють мати кілька варіантів представлення моделі, в кожному з яких можуть бути підкреслені певні деталі, які викликали б перенасичення моделі, якщо б вони були поміщені на одному поданні. Предметні області допомагають відокремити окремі фрагменти моделі, які відносяться лише до певної області, з числа тих, що охоплює інформаційна модель. Інтерфейс середовища розробки ERwin представлений на рисунку 1.2.

ERwin має потужні засоби візуалізації моделі, такі, як використання різних шрифтів, кольорів і відображення моделі на різних рівнях, наприклад, на рівні опису суті, на рівні первинних ключів сутності і т. д. Ці кошти ERwin значно допомагають при презентації моделі в колі розробників системи або стороннім особам.

Можливість використання моделі ERwin одночасно для логічного і фізичного представлення даних дозволяє по закінченні роботи отримати

повністю документовану модель.ERwin, як і інструмент моделювання бізнес-процесів BPwin, інтегрований з генератором звітів фірми CA/Logic Works - RPTwin. Цей засіб дозволяє отримувати докладні звіти по моделі, висвітлюючи різні ракурси та аспекти. Як вже говорилося, ERwin є не тільки інструментом для дизайну баз даних, він також підтримує автоматичну генерацію спроектованої визначеної на фізичному рівні структури даних.

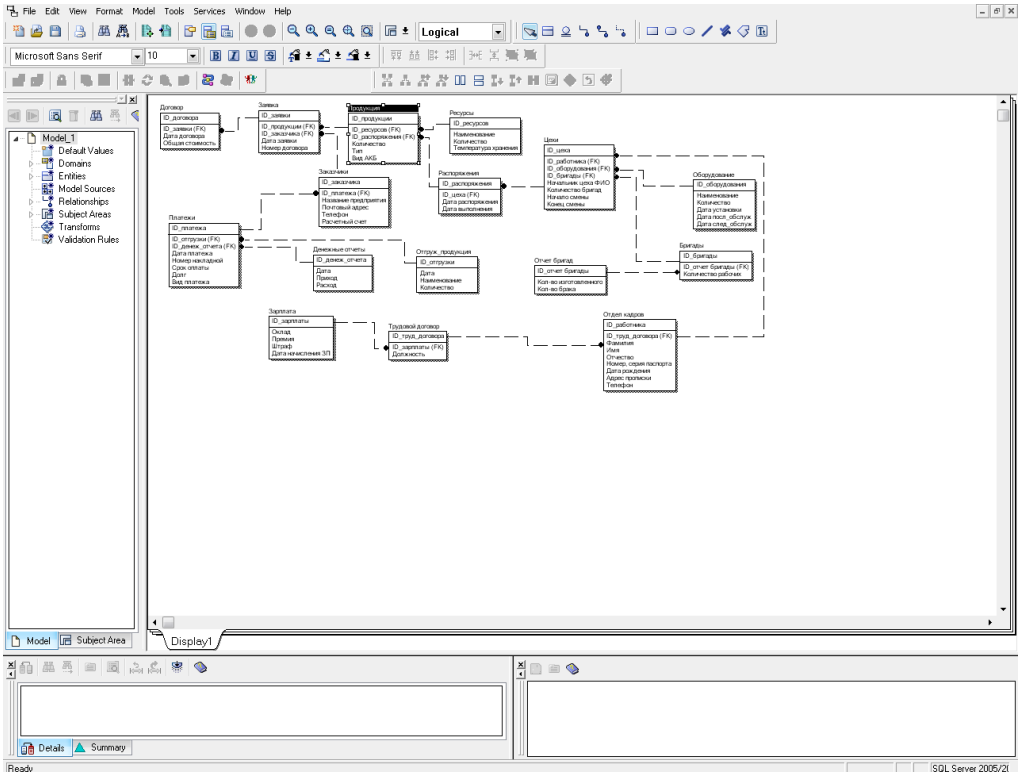


Рисунок 1.2 – Інтерфейс середовища розробки Erwin

ERwin підтримує широкий спектр серверних і настільних СУБД. У цей список входять такі продукти, як Microsoft SQL Server, Oracle, Sybase, DB2, INFORMIX, Red Brick, Teradata, PROGRESS, Microsoft Access, FoxPro, Clipper і багато інших. Для кожної з перерахованих СУБД в ERwin передбачено приєднання по "рідному" для цієї СУБД протоколу і підтримка всіх засобів управління даними, властивих цієї СУБД. ERwin не підтримує моделювання механізмів захисту бази даних, однак за допомогою макромови можна автоматично видати права на об'єкт, користуючись мовою визначення прав, який використовується в певній СУБД.

Розрізняють три рівня логічної моделі, що відрізняються по глибині представлення інформації даних про: діаграму сутність-зв'язок (Entity Relationship Diagram, ERD); модель даних, яка заснована на ключах (Key Based model, KB); повна атрибутивна модель (Fully Attributed model, FA).

Діаграма сутність-зв'язок являє собою модель даних верхнього рівня. Вона включає сутності та взаємозв'язку, що відображають основні бізнес-правила предметної області. Така діаграма не занадто деталізована, в неї включаються основні сутності і зв'язки між ними, які задовольняють основним вимогам, що пред'являються до ІС. Діаграма сутність-зв'язок може включати зв'язки багато-до-багатьох і не включати опис ключів. Як правило, ERD використовується для презентацій та обговорення структури даних з експертами предметної області.

Модель даних, заснована на ключах - більш детальне представлення даних. Вона включає опис всіх сутностей і первинних ключів і призначена для представлення структури даних та ключів, які відповідають предметній області.

Повна атрибутивна модель - найбільш детальне представлення структури даних: дані в третій нормальній формі і включає всі сутності, атрибути та зв'язки.

ERwin має засіб, який виконує завдання, зворотний генерації, що називається "зворотний розробка" (reverse engineering). Тобто ERwin може приєднатися до СУБД, отримати всю інформацію про структуру бази даних і відобразити її у графічному інтерфейсі, зберігши всі сутності, зв'язку, атрибути та інші властивості. Таким чином, можна переносити існуючу структуру даних з однієї платформи на іншу, а також дослідити структуру існуючих баз даних.
ERwin має засіб Complete-Compare, яке є єдиним на даний момент засобом інтерактивного розробки. ERwin демонструє розбіжності між моделлю і базою даних, ці невідповідності можна переносити або залишати без змін. За допомогою цього засобу можна всі зміни моделі вносити в базу даних

автоматично без необхідності контролю за відповідністю моделі і бази даних "вручну", при цьому існуючі дані не будуть порушені.

ERwin тісно інтегрований з іншими продуктами CA/Logic Works. Словник даних, який створений при аналізі бізнес-процесів за допомогою інструменту BPwin, може бути використаний як основа для побудови моделі бази даних. Однак взаємозв'язок між цими двома інструментами двосторонній, моделі BPwin і ERwin можна постійно підтримувати в узгодженому стані. Інтеграція цих двох продуктів дуже важлива з точки зору їх спільного використання при розробці програмного забезпечення, оскільки відпадає необхідність в повторному виконанні дій і процес створення словника даних стає практично автоматичним.

ERwin підтримує стандартну нотації IDEF1x для ER-діаграм, моделей даних, позначень IE і спеціальну позначення, призначену для проектування сховищ даних - Dimensional.

IDEF1X є методом для розробки реляційних баз даних і використовує умовний синтаксис, спеціально розроблений для зручної побудови концептуальної схеми. Концептуальною схемою ми називаємо універсальне представлення структури даних в рамках комерційного підприємства, незалежне від кінцевої реалізації бази даних і апаратної платформи. Будучи статичним методом розробки, IDEF1X спочатку не призначений для динамічного аналізу за принципом "AS IS", тим не менш, він іноді застосовується в цій якості, як альтернатива методу IDEF1. Використання методу IDEF1X, найбільш підходе для побудови логічної структури бази даних після того, як всі інформаційні ресурси досліджені (скажімо з допомогою методу IDEF1) і рішення про впровадження реляційної бази даних, як частини корпоративної інформаційної системи, було прийнято. Однак не варто забувати, що засоби моделювання IDEF1X спеціально розроблені для побудови реляційних інформаційних систем, і якщо існує необхідність проектування іншої системи, скажімо об'єктно-орієнтованої, то краще обрати інші методи моделювання.

Існує кілька очевидних причин, за якими IDEF1X не слід застосовувати у випадку побудови нереляційних систем. По-перше, IDEF1X вимагає від проектувальника визначити ключові атрибути, для того щоб відрізнити одну сутність від іншої, в той час як об'єктно-орієнтовані системи не вимагають завдання ключових ключів, з метою ідентифікування об'єктів. По-друге, у тих випадках, коли більш ніж один атрибут є однозначним і ідентифікує сутність, проектувальник повинен визначити один з цих атрибутів первинним ключем, а всі інші-вторинними. Таким чином, побудована проектувальником IDEF1X-модель призначена для побудови реляційної системи.

Корпоративні системи управління підприємством, створені на основі реляційних СУБД, як правило, ефективно вирішують завдання обліку, контролю та зберігання даних. Проте в силу своєї специфіки, реляційна структура не дозволяє вирішувати задачі аналізу наявної інформації з необхідною продуктивністю. Особливо гостро ця проблема стоїть в гетерогенних інформаційних середовищах, коли в центральному офісі організації і у філіях експлуатуються СУБД різних виробників (рис.1.3).

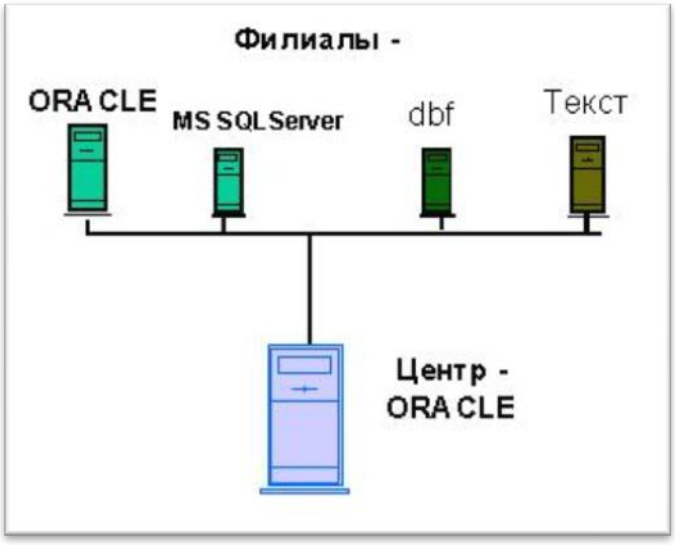


Рис.1.3 – Інформаційне середовище.

Така ситуація часто виникає або в результаті злиття компаній, коли компанія перетворюється на філію більш великої, але при цьому не рентабельно перебудовувати історично сформовану інформаційну

інфраструктуру, або внаслідок незадовільного управління, коли філії не дотримуються корпоративного стандарту і впроваджують власні інформаційні системи. Однією з основних задач, вирішуваних в корпоративних інформаційних системах, є надання аналітичної інформації, необхідної для прийняття рішень.

Для підтримки прийняття рішення необхідний не один заздалегідь підготовлений звіт, а серія різноманітних звітів, при чому менеджер не завжди уявляє, якою саме звіт знадобиться йому в наступні півгодини. Наприклад, при аналізі продажів компанії виявляється, що в лютому поточного року стався спад. Щоб з'ясувати причини спаду, необхідно переглянути звіт про продажі в регіонах. Звіт про продажі в регіонах показує, що спад стався, мабуть, з причини незадовільної роботи однієї з філій, отже, необхідний звіт про роботу даного філії. Організувати виконання таких звітів в гетерогенному середовищі вкрай складно. Для ефективного аналізу даних у цьому випадку необхідно об'єднувати в одному запиті дані з різних джерел. В даний час існують моніторінги транзакцій і генератори звітів (наприклад, Crystal Reports), що володіють такою функціональністю, однак продуктивність таких систем не може бути високою. У процесі аналізу, дані необхідні для прийняття рішень і повинні надходити до споживача в режимі реального часу.

Сховища даних дозволяють розвантажити оперативні бази даних, і тим самим, дозволяють користувачам більш ефективно і швидко отримувати необхідну інформацію. Сховища даних можуть бути включені в загальну корпоративну мережу, по якій в сховище заздалегідь визначене розкладом, як правило, в період найменшого завантаження мережі і серверів копіюється накопичена за день, або за тиждень інформація. Оскільки дані змінюються рідко, то до сховища даних не пред'являються жорсткі вимоги, які звичайно пред'являються до звичайних баз даних - відсутність аномалій при виконанні операцій оновлення або видалення надмірності зберігання інформації. З цієї причини може скластися невірне уявлення, що проектувати сховище

простіше, ніж бази даних, призначені для оперативної обробки інформації. Насправді, проектування і резервного копіювання сховища даних є досить складним завданням.

Оскільки звіт буде створювати кінцевий користувач, повинні бути спрощені вимоги до запитів з метою виключення тих запитів, які могли б вимагати множинних тверджень SQL в реляційних СУБД. Обробка запитів до сховища повинна бути проведена з високою продуктивністю, бажано в реальному масштабі часу, тому повинна бути забезпечена підтримка складних запитів SQL, які вимагають послідовної обробки тисяч або мільйонів записів.

Хоча реалізувати сховище даних можна на будь-якому сервері БД, існують спеціалізовані сервера, спеціально призначені для підтримки сховищ даних. Erwin підтримує генерацію схеми БД для двох таких серверів – Teradata і Red Brick.

Нормалізація даних корисна при моделюванні реляційної структури, але вона зменшує ефективність виконання запитів до сховища даних. У розмірній моделі головною метою є забезпечення високої ефективності перегляду даних і виконання складних запитів. Схема зазвичай перешкоджає ефективності, тому що вимагає об'єднання багатьох таблиць для побудови результуючого набору даних, що збільшує час виконання запиту. Тому при проектуванні не слід зловживати створенням безлічі консольних таблиць.

Генерація схеми бази даних. В ERwin включені оптимізовані шаблони тригерів посилальної цілісності, і потужний незалежний від платформи макромова, що підтримує настройку тригерів і збережених процедур. Виходячи з фізичної структури моделі, генеруються повні визначення наступних елементів бази даних відповідно до цільової СУБД: бази даних/табличні простору, таблиці та подання; стовпці з обмеженнями за замовчуванням і обмеженнями доменів; первинні ключі, зовнішні ключі, індекси; збережені процедури і код тригерів; об'ємний та інші фізичні властивості.[3]

1.3.Характеристика SQL Server 2008 R2

Microsoft SQL Server 2008[4] являє собою розробку в області баз даних і призначений для швидкого створення масштабованих рішень електронної комерції, бізнес-додатків і сховищ даних.

Визначення сутностей.

На цьому етапі вам необхідно визначити сутності, з яких буде створена база даних.

Сутність — це об'єкт бази даних, в якому зберігаються дані. Сутність може представляти собою щось дійсне (будинок, людина, предмет, місце) або абстрактне (банківська операція, відділ компанії, маршрут автобуса). У фізичній моделі сутність називається таблицею. Сутності складаються з атрибутів (стовпців) і записів (рядків в таблиці). Зазвичай бази даних складаються з декількох основних сутностей, пов'язанних з великою кількістю підлеглих сутностей. Основні сутності називаються незалежними: вони не залежать ні від будь-якої іншої сутності. Підлеглі сутності називаються залежними, для того щоб існувала одна з них, повинна існувати пов'язана з нею основна таблиця.

На діаграмах сутності зазвичай представляються у вигляді прямокутників. Будь-яка таблиця має наступні характеристики:

у ній немає однакових рядків;

всі стовпці (атрибути) у таблиці повинні мати різні імена;

елементи в межах однієї колонки мають однаковий тип (рядок, число, дата);

порядок слідування рядків у таблиці може бути довільним.

На цьому етапі вам необхідно виявити всі категорії інформації (сутності), які будуть зберігатися в базі даних.

1.3.1 Визначення атрибутів

Атрибут є властивістю, що описує сутність. Атрибути часто бувають числом, датою або текстом. Всі дані, що зберігаються в атрибуті, повинні мати однаковий тип і володіти однаковими властивостями. У фізичній моделі атрибути називають колонками.

Після визначення сутностей необхідно визначити всі атрибути цих сутностей. Для кожного атрибута визначається тип даних, їх розмір, допустимі значення і будь-які інші правила. До їх числа відносяться правила обов'язкового заповнення, змінності та унікальності.

Правило обов'язкового заповнення визначає, чи є атрибут частиною сутності. Якщо атрибут є необов'язковою частиною сутності, то він може приймати значення NULL-значення, інакше - ні.

Також ви повинні визначити, чи є атрибут змінним. Значення деяких атрибутів не можуть змінитися після створення запису.

І, нарешті, вам потрібно визначити, чи є атрибут унікальним. Якщо це так, то значення атрибута не можуть повторюватися.

1.3.2 Ключі

Ключ (key) називається набір атрибутів, що однозначно визначає запис. Ключі діляться на два класи: прості та складені.

Простий ключ складається лише з одного атрибута. Наприклад, в базі "Паспорта громадян країни" номер паспорта буде простим ключем: адже не буває двох паспортів з однаковим номером.

Складений ключ складається з кількох атрибутів. В тій же базі "Паспорти громадян країни" може бути складовою ключ з наступними атрибутами: прізвище, ім'я, по батькові, дата народження, оскільки цей складений ключ, теоретично, не забезпечує гарантованої унікальності запису. Також існує кілька типів ключем:

Можливий ключ являє собою будь-який набір атрибутів, що однозначно ідентифікують запис у таблиці. Можливий ключ може бути простим або складеним.

Кожна сутність повинна мати, принаймні, один можливий ключ, хоча таких ключів може бути кілька. Жоден з атрибутів первинного ключа не може приймати невизначений (NULL) значення.

Можливий ключ називається також сурогатним.

Первинним ключем називають сукупність атрибутів, однозначно ідентифікуючий запис в таблиці (сутності). Один з можливих ключів стає первинним ключем. На діаграмах первинні ключі часто зображаются вище основного списку атрибутів або виділяються спеціальними символами. Сутність має як ключові, так і звичайні атрибути.

Альтернативні ключі. Будь-який можливий ключ, який не є первинним, називається альтернативним ключем. Сутність може мати кілька альтернативних ключів.

Зовнішнім ключем називається сукупність атрибутів, що посилаються на первинний або альтернативний ключ іншої сутності. Якщо зовнішній ключ не пов'язаний з первинною сутністю, то він може містити тільки невизначений. Якщо при цьому ключ є складеним, то всі атрибути зовнішнього ключа повинні бути невизначеними.

На діаграмах атрибути, що об'єднуються під зовнішні ключі, позначаються спеціальними символами.

1.3.3 Нормалізація

Логічна схема бази даних, що включає таблиці та зв'язки між ними, є основою оптимізації реляційної бази даних. Хороша логічна схема бази даних може забезпечити фундамент для оптимальної продуктивності бази даних і додатки. Погана логічна схема бази даних може знизити продуктивність всієї системи.

Нормалізацією називається процес видалення надлишкових даних з бази даних. Кожен елемент повинен зберігатися в базі в одному і тільки в одному примірнику. Існує п'ять найпоширеніших форм нормалізації. Як правило, база даних приводиться до третьої нормальної формі.

У процесі нормалізації виконуються певні дії по видаленню надлишкових даних. Нормалізація підвищує швидкодію, прискорює сортування і побудову індексу, зменшує кількість індексів в сутності і прискорює операції вставки та оновлення.

Нормалізація логічної схеми бази даних включає використання формальних методів для розділення даних на кількох зв'язаних таблиць. Наявність досить великої кількості вузьких таблиць з невеликою кількістю стовпців є характерною особливістю нормалізованої бази даних. Наявність невеликої кількості широких таблиць з великою кількістю стовпців є ознакою ненормализованной бази даних. Раціональна нормалізація часто покращує продуктивність. При можливості використання корисних індексів SQL Server оптимізатор запитів є дієвим засобом у виборі швидких, ефективних зв'язків між таблицями. Нижче наведені деякі з переваг нормалізації:

більш швидке сортування та створення індексів;

більша кількість кластерних індексів.

більш вузькі і компактні індекси;

менша кількість індексів у таблиці. Це покращує продуктивність інструкцій INSERT, UPDATE і DELETE;

менша кількість значень NULL і більш низька вірогідність неузгодженості. Це підвищує компактність бази даних. При підвищенні нормалізації кількість і складність зв’язків, необхідних для отримання даних, також зростають. Занадто велика кількість складних реляційних зв'язків між занадто великою кількістю таблиць може знизити продуктивність. Раціональна нормалізація часто включає кілька регулярно виконуваних запитів, які використовують сполуки, що включають більше чотирьох таблиць.

Іноді логічна схема бази даних вже зафіксована, і повне повторне проектування не реальне. Однак навіть у цьому випадку є можливість вибірково нормалізувати великі таблиці на декілька більш дрібних. Якщо

доступ до бази даних можливий за допомогою збережених процедур, ця зміна схеми може бути здійснена без нанесення шкоди програми.

при проектуванні реляційних баз даних правила нормалізації вказують на певні атрибути, які повинні бути присутніми або відсутніми у добре спроектованої бази даних. Є декілька правил, які можуть бути корисними в досягненні бездоганною структури бази даних:

таблиця повинна мати ідентифікатор;

у таблиці повинні зберігатися дані тільки для одного типу сутності;

в таблиці слід уникати наявності стовпців, що дозволяють значення NULL;

в таблиці не повинно міститися повторюваних значень або стовпців;

Також, іноді повністю нормалізована база даних може не вписуватися в побудовану вами схему даних. Це може виявлятися, наприклад, на загальній продуктивності системи. У такому випадку варто об'єднати в одну таблицю дані, які дуже довго обробляються в запиті.

Так, при роботі у програмі зі статичною таблицею має прямий сенс завантажити її в локальну таблицю для подальшого звернення з боку програми.

Існують такі варіанти розташування таблиць, коли одинарні зв'язки між ними не дають повної картини. Наприклад, якщо у вас є таблиці Customer, Account Balance і Sales Representative, зв'язки між якими засновані на номері замовника, то для отримання повної картини вам знадобиться створити потрійний зв'язок.

У такому разі потрібно провести денормализацію набору таблиць. Можна створити одну-єдину таблицю, в рядках якої буде міститися інформація з усіх трьох згаданих таблиць. Таким чином, запросивши інформацію про замовника X, ви відразу отримаєте всі необхідні відомості, та додаткові зв'язки вже не знадобляться.

У цих уроках ви отримали багато інформації, яка буде корисною при роботі. Виконуйте нормалізацію таблиць до найвищої міри, яка тільки

можлива. Якщо виникла необхідність порушити правила нормалізації, ще раз уважно все перевірити. Можливо, існує інший, непомічений вами раніше спосіб досягнення мети.[4]

1.3.4 Цілісність даних

Організувавши дані в таблиці і визначивши зв'язку між ними, можна вважати, що була створена модель, правильним чином, яка відображає бізнес-середовище. Тепер треба забезпечити, щоб дані введені в базу, давали правильне уявлення про стан справи. Іншими словами, потрібно забезпечити виконання ділових правил і підтримку цілісності (integrity) бази даних.

Наприклад, ваша компанія займається доставкою книг. Ви навряд чи приймете замовлення від невідомого клієнта, адже тоді ви навіть не зможете доставити заказ. Звідси бізнес-правило: замовлення приймаються лише від клієнтів, інформація про яких є в базі даних.

Коректність даних у реляційних базах забезпечується набором правил. Правила цілісності даних діляться на чотири категорії.

Цілісність сутностей - кожний запис сутності повинен володіти унікальним ідентифікатором і містити дані. Адже треба вам якось розрізняти всі ці записи в базі даних.

Цілісність атрибутів - кожен атрибут приймає лише допустимі значення. Наприклад, сума покупки, безумовно, не може бути менше нуля.

Посилальна цілісність - набір правил, що забезпечують логічну узгодженість первинних і зовнішніх ключів при вставці, оновлення, видалення записів. Посилальна цілісність забезпечує, щоб для кожного зовнішнього ключа існував відповідний первинний ключ.

Реляційні бази даних збирають одноманітні або пов'язані дані в єдиний список. Наприклад, вся інформація про товари може бути перерахована в одній таблиці, а про клієнтів іншого. Таблиці бази даних в чомусь схожі з електронними таблицями і також складаються з рядків і стовпців. Електронні таблиці привабливі своїм стилем обробки інформації, що полегшує їх додавання і модифікацію. На практиці менеджери схильні зберігати

критичну інформацію в електронних таблицях, і багато баз даних беруть свій початок саме з них. Як в електронних таблицях, так і в таблицях баз даних кожна рядок представляє деякий елемент списку, а кожен стовпець є деяким фраг ментом даних про цьому елементі. У той час як електронні таблиці зазвичай мають довільну структуру таблиці бази даних можуть пред'являти суворі вимоги до даних у стовпцях. Так як цілісність рядків і стовпців критична в таблиці бази даних, то і сама конструкція таблиці критична.[5]

1.4 Характеристика програмного забезпечення Delphi 7

Delphi 7[6] представляє собою ряд принципово нових можливостей, які базуються на серйозних змінах мови програмування і ядра середовища розробки, які дозволяють вивести розробку додатків з розвиненим графічним інтерфейсом на новий рівень. Він надає досконалий інструментарій для розробників програмного забезпечення і баз даних, який дозволяє швидко розробляти високопродуктивні і прості в обслуговуванні застосування для Windows. А також за допомогою обширного набору компонентів і повноцінної підтримки різних баз даних стало можливим створювати роботи набагато швидше і ефективніше.

Delphi – це нащадок середовища програмування TurboPascal. Назва середовища походить від назви міста, у Стародавній Греції, де був знаменитий Дельфійский оракул (храм Аполлона у місті Дельфи, жерці якого займалися віщуваннями).

Система візуального об’ектно-орієнтованого проектування Delphi дозволяє:

1. Створювати закінчені форми для Windows самої різної спрямованості.

2. Швидко створювати професійний віконний інтерфейс для будь-яких додатків; інтерфейс задовольняє всі вимоги Windows і автоматично налаштовується на систему, яка встановлено, оскільки використовує функції, процедури і забезпечення бібліотеки Windows.

3. Створювати свої динамічні приєднувані бібліотеки компонентів, форм, функцій, які потім використовувати з деяких інших мов програмування.

4. Створювати потужні системи роботи з базами даних будь-яких типів.

5. Формувати й друкувати складні звіти, які включають таблиці, графіки тощо.

6. Створювати довідкові системи, як для своїх додатків, так і для будь-яких інших.

7. Створювати професійні програми інсталяції для додатків Windows, враховують всю специфіку та всі вимоги ОС.

Інтегроване середовище розробки Delphi – це середовище, яке має усе необхідне для проектування, запуску і тестування створюваних додатків. Більшість версій Delphi випускається у кількох варіантах: а) стандартна, б) професійна версія, в) розробка баз даних предметних областей. Ці варіанти розрізняються переважно різними рівнями доступу до систем управління базами даних. Останні два варіанта є потужними у цьому плані. Бібліотеки компонентів у різних варіантах практично однакові.

Головна програма складається з оголошення списку використовуваних модулів і знання кількох операторів, які створюють об'єкти для необхідних форм і запускаючих додатків виконання. Модульність дуже важлива для створення надійних і легко модифікуючих супроводжуваних додатків. Чітке дотримання принципів модульності разом із принципом приховання інформації, дає змогу виробляти модифікації всередині будь-якого модуля, не чіпаючи інших модулів і головну програму.[6]

Усі об'єкти компонентів розміщуються в об'єктах – формах. Для кожної форми, проектованої при застосуванні, Delphi створює окремий модуль. Саме в модулях здійснюється програмування завдання. У обработчиках подій об'єктів розміщуються описи алгоритмів, які переважно зводяться до опрацювання інформації, котра міститься в властивості одних об'єктів, і завданні за результатами цієї обробки властивостей інших об'єктів.

У процесі проектування Delphi автоматично створює код головної програми розвитку й окремих модулів. У модулі вводяться власні коди, створюючи оброблювачі різних подій.

Головний файл докладання Delphi має таку структуру:

>Program < ім'я >;

{оголошення підключаємих модулів, і навіть локальних типів, класів, констант, змінних, опис локальних функцій і змінних}

>Begin

{оператори тіла програми}

End.

Типова головна програма докладання має такий вигляд:

>Program Project 1;

>Uses

>Forms,

>Unit 1 in '>Unit1.pas' {>Form 1},Unit 2 in '>Unit2.pas' {>Form 2};

{$R *.res}

{можна помістити опис констант, змінних, функцій, процедур, доступних від використання даного файла}

>Begin

>Application.Initialize;

>Application.CreateForm (>TForm 1,Form 1);

>Application.CreateForm (>TForm 2,Form 2);

>Application.Run;

End.

1) Програма починається з ключового слова Program, після якого вказується ім'я програми (воно збігаються з ім'ям файла, у якому збережено проект). Це ж ім'я присвоюється виконуваному файлу докладання. За умовчанням використовується ім'я Project1.

2) Після заголовка з тексту програми розташовуються підключаємі модулі:

>Uses

>Forms,

>Unit 1 in '>Unit1.pas' {>Form 1},Unit 2 in '>Unit2.pas' {>Form 2};

В результаті для підключення кожного програмного перераховуються модулі, завантажуваними програмою. Перший модуль Forms є системним, а такі модулі розроблені самостійно формою. Цей приклад передбачає, що у проекті було створено дві форми із конкретними іменами Form1,Form2 в модулях Unit1,Unit2.

3) Наступний рядок тексту – {$R *. >res} – це директиви компілятора, пов'язані з допомогою файлів ресурсів. Зазначений файл може бути файлом ресурсів Windows. За замовчуванням використовується розширення .RES для файлів ресурсів.

4) Перший оператор у тілі програми >Application. >Initialize; - ініціалізує додаток, наступний його оператор >Application. >CreateForm (>TForm 1,Form1); і Application.CreateForm (>TForm 2,Form 2); – створює об'єкти форми1 і форми2, останній оператор >Application. >Run; – починає виконання програми.

Загальна структура файла модуля:

>Unit < ім'я модуля >;

>Interface - це відкритий інтерфейс модуля (можуть поміщатися списки підключаємих модулів, оголошення типів, констант, змінних, функцій і процедур, яких буде доступ з деяких інших модулів)

>Implementation – це реалізація модуля (можуть поміщатися списки підключаємих модулів, оголошення типів, констант, змінних, функцій і процедур, яких нічого очікувати доступу з деяких інших модулів)

>Initialization – оператори які виконуються з першого разу у зверненні до модуля)

>Finalization – оператори які виконуються незалежно від завершення роботи модуля)[7]

Компіляція та складання проекту можє виконуватися на будь-якій стадії розробки проекту. Під компіляцією розуміється отримання об'єктних модулів (DCU - файлів) з вихідних текстів програмних модулів (PAS - файлів). Під складанням розуміється отримання виконуваного файлу з об'єктних модулів.

Для виконання компіляції досить виконати команду меню Project | Compile <Ім'япроекту> або натиснути комбінацію клавіш Ctrl+F9. При цьому компілюються всі вихідні модулі, вміст яких змінювалося після останньої компіляції. В результаті для кожного програмного модуля створюється файл з розширенням DCU (скор. від Delphi Compiled Unit). Потім середовище

Delphi компілює головний файл проекту і збирає з DCU - модулів виконуваний файл, ім'я якого збігається з ім'ям проекту. Зауважимо, що компілятор середовища Delphi викидає з виконуваного файлу весь використовуваний програмний код, тому не варто хвилюватися з приводу зайвих об'єктів і підпрограм, які можуть присутні в підключених модулях. Існує особливий вид компіляції і збірки — повна примусова компіляція всіх програмних модулів проекту, для яких доступні вихідні тексти, з подальшим складанням виконуваного файлу. При цьому не важливо, вносили в них зміни після попередньої компіляції чи ні. Повна компіляція проекту виконується за допомогою команди головного меню Project | Build <Ім'япроекту>. В результаті теж створюється виконуваний файл, але на це витрачатися трохи більше часу.

Коли після многочисленних компіляцій ви виправите всі помилки і отримаєте – виконуваний файл, можна буде подивитися на результат вашої праці. Для цього треба виконати створену програму за допомогою команди меню Run | Run або клавіші F9. Перед виконанням буде автоматичний процес компіляції (якщо в проект вносились зміни) і після його успішного завершення програма запуститься на виконання. В результаті ви побачите на екрані його головну форму.

Виберіть у меню команду File | New | Application. Середовище Delphi автоматично створить в новому проект чисту форму і помістить її вихідний текст у редактор коду (рис. 1.4).

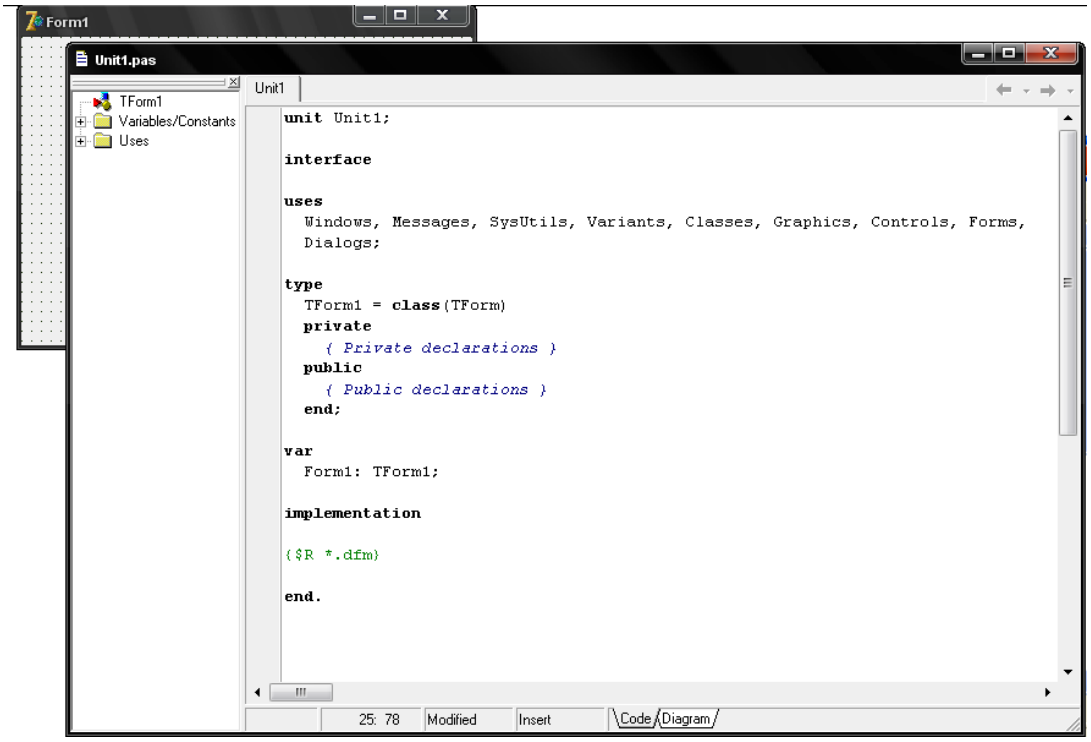


Рисунок 1.4 – Форма та її описання в редакторі кода

Відразу збережіть проект і його форму, щоб дати їм імена. Виберіть у меню команду File | SaveAll і дайте модулю ім'я Main.pas, а проекту — ім'я FormTest.dpr.[8]

1.5 Постановка задачі

1. Розробити модель діяльності підприємства, що визначає мету і стратегії, бізнес-процеси, організаційну структуру і потоки даних.

2. Виконати аналіз програмних продуктів, призначених для моделювання бізнесів і структурне проектування (SADT или IDEF) програмних систем.

3. Дослідити характеристики, призначення і умови застосування CASE-средств AllFusion Process Modeler(BPwin)+AllFusion Data Modeler(Erwin).

4. Дослідити характеристики, призначення і умови застосування вибранного серверу.

5. Виконати аналіз програмних продуктів для проектування додатків адміністратора.

6. За допомогою CASE-засобів BPwin побудувати контекстну діаграму і діаграми декомпозиції в нотації IDEF0 і діаграму потоків даних в нотації Гейна-Сарсона.

7. Розробити ER-діаграму в нотації IDEF1x і створити в середовищі CASE-засобу Erwin для вибранної реляційної СУБД логічну і фізичну модель бази даних проектованої інформаційної системи.

8. Сгенерувати в середовищі Erwin SQL-скрипт для ER-діаграми і виконати перевірку програмного кода в аналізаторі кодів сервера.

9. Створити на сервері схему даних проектованої системи і розробити SQL-оператори для виконування операцій запису, видалення, додавання у таблицю даних.

10. Розробити додаток клієнта для кінцевого користувача

11. Виконати налагоджування, доопрацювання і тестування системи.

РОЗДІЛ 2

РОЗРОБКА МОДЕЛІ ДАНИХ

2.1 Сценарії діяльності підприємства.

Технологія приготування сплаву для відливу решіток

1. Сировина (свинець і сурма різних марок) надходить на завантажувальний транспортер.
2. Спочатку оператор завантажує в котел свинець і нагріває до 400°С.
3. Плавильник через 15 хвилин після початку плавлення з поверхні свинцю знімає нерозчинний шар оксидів.
4. Потім оператор доводить температуру до 450— 500° С і в котел завантажує необхідну кількість сурьми.
5. Плавильник в свою чергу через 15-20 хвилин знову знімає шар окислів, і помішуючи мішалкою стежить за плавкою.
6. Начальник зміни стежить щоб дотримувалися норми процесу плавки (допустима температура, час плавки, загруж. кількість сурми,) і заносить всі дані в журнал.
7. Готовий сплав оператор випускає з нижньої частини котла в розливні котли.
8. Оператор вивантажує сплав з котлів у форми для відливання АК решіток.
9. Заповнена сплавом форма рухається по конвеєрної стрічки на стадію охолодження решіток.
10. Форми завантажуються на конвеєрну стрічку, для транспортування на склад і навантаження в спеціальні ящики, попередньо на стрічці відбираються браковані решітки.
11. Браковані решітки надходять на переплавку в котел.
12. Потім транспортувальники перевозять решітки на склад.

Технологія транспортування готових решіток в цех дозрівання

1. Працівник вручну завантажує готові сітки з конвеєрної стрічки на спец. піддони.

2. Водій перевозить піддони в цех дозрівання.

3. Оператор виставляє потрібну температуру і час, в кожній камері дозрівання.

4. Водій перевозить піддони з гратами на склад для подальшого зберігання.

Технологія виробництва пасти для намазування пластин(решіток)

1. Рідкий свинець розливають у форми циліндрів.

2. Оператор завантажує циліндри в барабан, для одержання порошку.

3. Порошок транспортують на інший цех.

4. Порошок завантажують у величезний вакуумний міксер.

5. Додають H2SO4 і інші добавки.

6. Оператор регулює температуру, час змішення, і заносить дані в журнал.

7. Паста надходить через труби в спец. верстат.

8. Пасту намазують на решітки.

9. Нагрівання решіток до визначенної температури для затвердіння.

Технологія виготовлення пластикових форм АКБ

1. Завантаження твердого поліпропілену в піч.

2. Регулярне помішування поліпропілену в печі.

3. Перевірка і дотримання всіх норм виготовлення.

4. Відливання в спеціальні форми для АКБ.

5. Охолодження форм.

6. Перевезення готових форм на склад.

Технологія перевірки АКБ

1. Перевірка електроліту.

2. Доливання електроліту.

3. Закручування пробки АКБ.

4. Мийка і сушка АКБ.

5. Перевірка АКБ високим струмом.

6. Перевірка АКБ високою напругою.

7. Клеймування АКБ.

8. Перевірка АКБ на техн і механічні пошкодження.

9. Наклеювання етикетки.

10. Перевезення АКБ на склад.

Технологія продажу продукції

1. Замовник подає заявку у відділ на купівлю певної кількості АКБ.

2. Секретар приймає і реєструє заявку.

3. Працівник відділу підраховує загальну вартість АКБ.

4. Ціна та кількість продукції АКБ погоджується між замовником і головним економістом.

5. Складається договір, який скріплюється мокрими печатками обох сторін.

6. Працівник готує документи на оплату і передає їх замовникові.

7. Оплата продукції здійснюється у безготівковому розрахунку.

8. Після отримання платежу працівник відділу оформляє накладну.

9. Замовник пред'являє накладну начальника складу.

10. Начальник складу звіряє номер накладної та номер договору в базі.

11. Вантажник відвантажує вказану кількість партії АКБ в накладній замовника.

2.2 Розробка діаграм IDEF0 і DFD.

BPwin є потужним інструментом для створення моделей, що дозволяють аналізувати, документувати і планувати зміни складних бізнес-процесів. BPwin пропонує засіб для збору всієї необхідної інформації про роботу підприємства і

графічного зображення цієї інформації у вигляді цілісної і несуперечливої моделі.

Кожна діаграма розробляється як окремий лист у форматі від A4 до A1 або Customer (користувацький). На малюнку 2.1. показана основа функціональної моделі Activity - функція, що графічним символом якої є прямокутник (Box - у першоджерелі), в якій вписано ім'я функції у вигляді дієслова або віддієслівного іменника.

Кожна функція (прямокутник Activity) пов'язана чотирма типами зв'язків:

Стрілки входу Input(входять у ліву грань роботи) — зображують дані або об'єкти, змінювані в процесі виконання роботи.

Стрілки керування Control (входять в верхню грань роботи) — зображують правила і обмеження, згідно з якими виконується робота.

Стрілки виходу Output(виходять з правої межі роботи) — зображують дані або об'єкти, що з'являються в результаті виконання роботи.

Стрілки механізму Mechanism (входять в нижню грань роботи) — зображують ресурси, необхідні для виконання роботи, але не змінюються в процесі роботи (наприклад, устаткування, людські ресурси...).

Ми розглянемо моделювання та побудови моделей IDEFO і DFD. Для функціональності системи застосовується діаграма IDEF0 (Integration Definition for Function Modeling) це бізнес-процес, який представляється у вигляді набору елементів-робіт, які взаємодіють між собою, а також показується інформаційні, людські і виробничі ресурси, що споживаються кожною роботою.

З точки зору потоків інформації (документообігу) у системі. Діаграми DFD (Data Flow Diagramming) можуть доповнити те, що вже відображено в моделі IDEF3, оскільки вони описують потоки даних, дозволяючи простежити, яким чином відбувається обмін інформацією між бізнес-функціями всередині системи. У теж час діаграми DFD залишають без уваги взаємодія між бізнес-функціями.

Для побудови послідовності виконуваних робіт, як виробляються технологічні процеси використовується модель діаграми IDEF3. Цей метод привертає увагу до черговості виконання подій. В IDEF3 включені елементи логіки, що дозволяє моделювати й аналізувати альтернативні сценарії розвитку бізнес-процесу.

У контекст входить визначення суб'єкта моделювання, цілі і точки зору на модель.

Під суб'єктом розуміється сама система, при цьому необхідно точно встановити, що входить в систему, а що лежить за її межами, іншими словами, ми повинні визначити, що ми будемо надалі розглядати як компоненти системи, а що як зовнішній вплив.

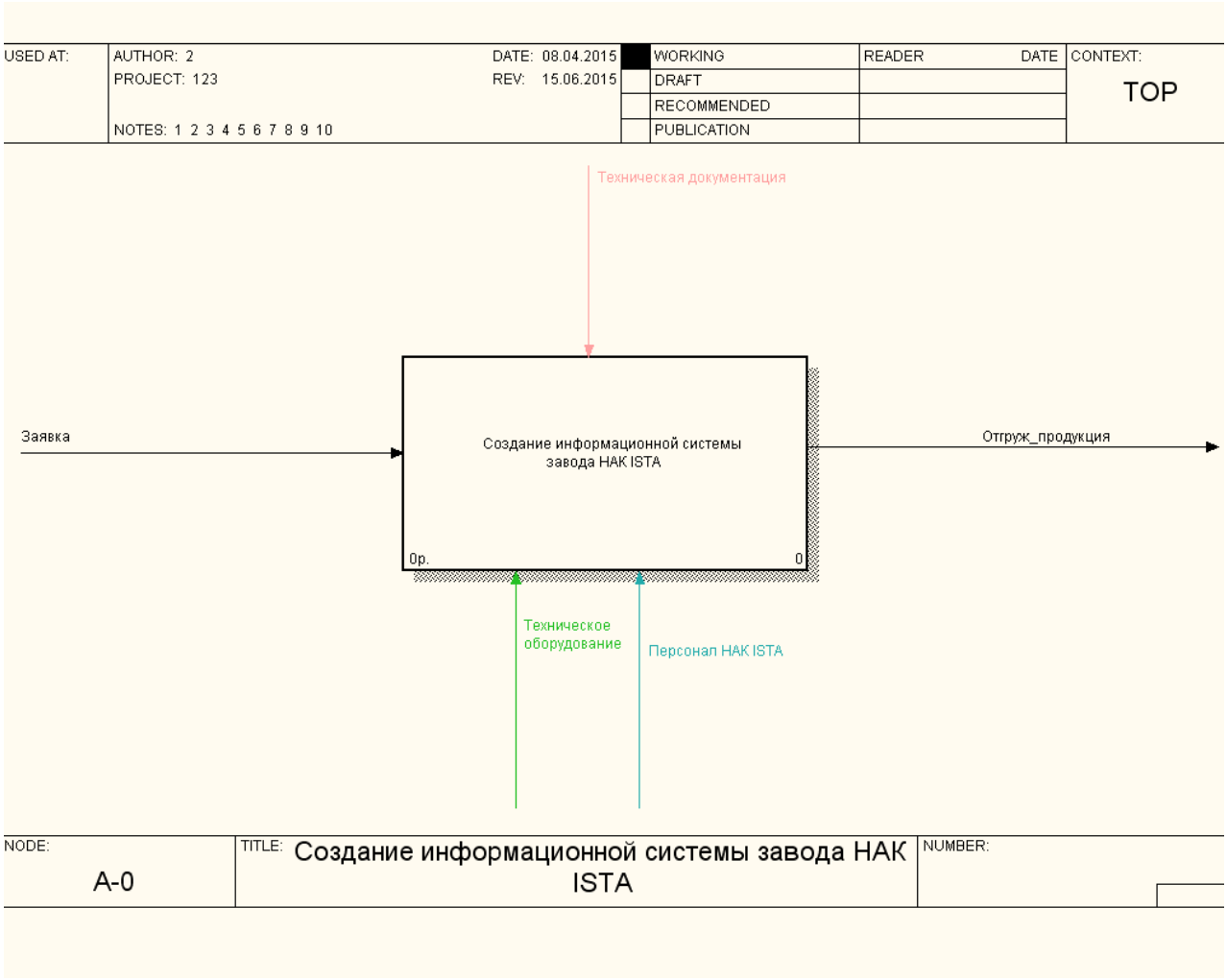


Рисунок 2.1 – Діаграма Activity

Це означає, що здійснення зазначеної функції в прямокутнику призводить до перетворення входу у вихід; що це перетворення здійснюється за допомогою механізму та під управлінням, символизированним стрілкою Control. Перша діаграма звана TOP в ієрархії діаграм IDEF0 завжди зображує функціонування системи в цілому. (верхній або контекстної) - діаграмою, має тільки один прямокутник Activity, який символізує роботу системи в цілому. Такі діаграми називаються контекстними. У контекст входить опис мети моделювання, області (опису того, що буде розглядатися як компонент системи, а що як зовнішній вплив) і точки зору (позиції, з якої буде будуватися модель). Зазвичай в якості точки зору вибирається точка зору особи або об'єкта, відповідального за роботу модельованої системи в цілому. Всі зв'язки на цій діаграмі є зв'язками модельованої системи із середовищем функціонування.

Всі роботи і стрілки повинні бути іменовані. Кожна Activity, починаючи з Activity TOP діаграми, може бути декомпозирована (розділена) на субфункции, подаються кількома Activities - див. рис. 2.2.

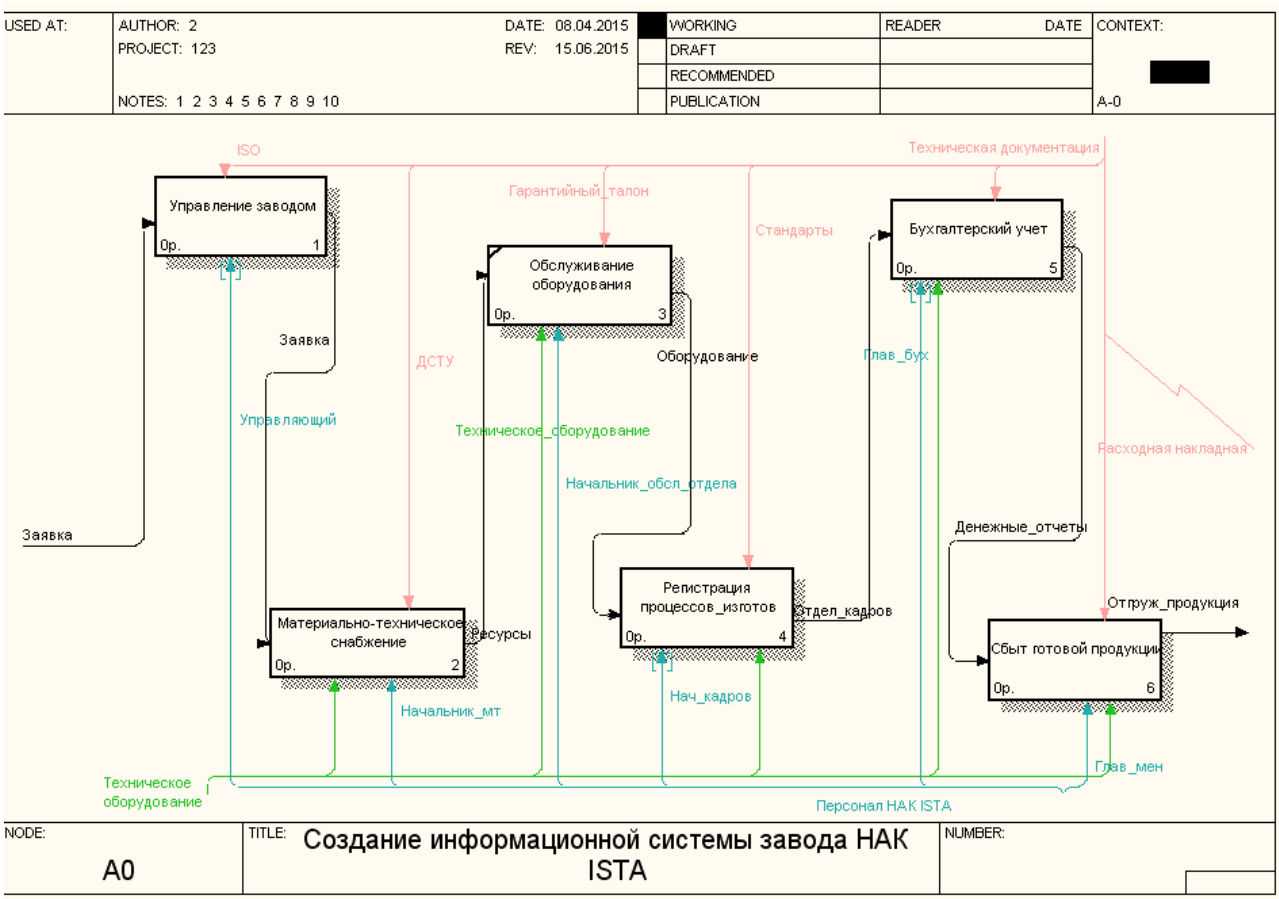


Рисунок 2.2 - Декомпозиція першої діаграми

2.2.1 Побудова діаграми IDEF0

Побудова IDEF0[16]-моделі починається з представлення всієї системи у вигляді найпростішої. Результатом застосування IDEF0 до деякої системі є модель цієї системи, що складається з ієрархічно впорядкованого набору діаграм, тексту документації та словників, пов'язаних один з одним за допомогою перехресних посилань. Двома найбільш важливими компонентами, з яких будуються діаграми IDEF0, є бізнес-функції або роботи (представлені на діаграмах у вигляді прямокутників) і дані і об'єкти (зображувані у вигляді стрілок), що зв'язують між собою роботи. Приклад декомпозиції контекстної роботи показаний на рис.2.4.

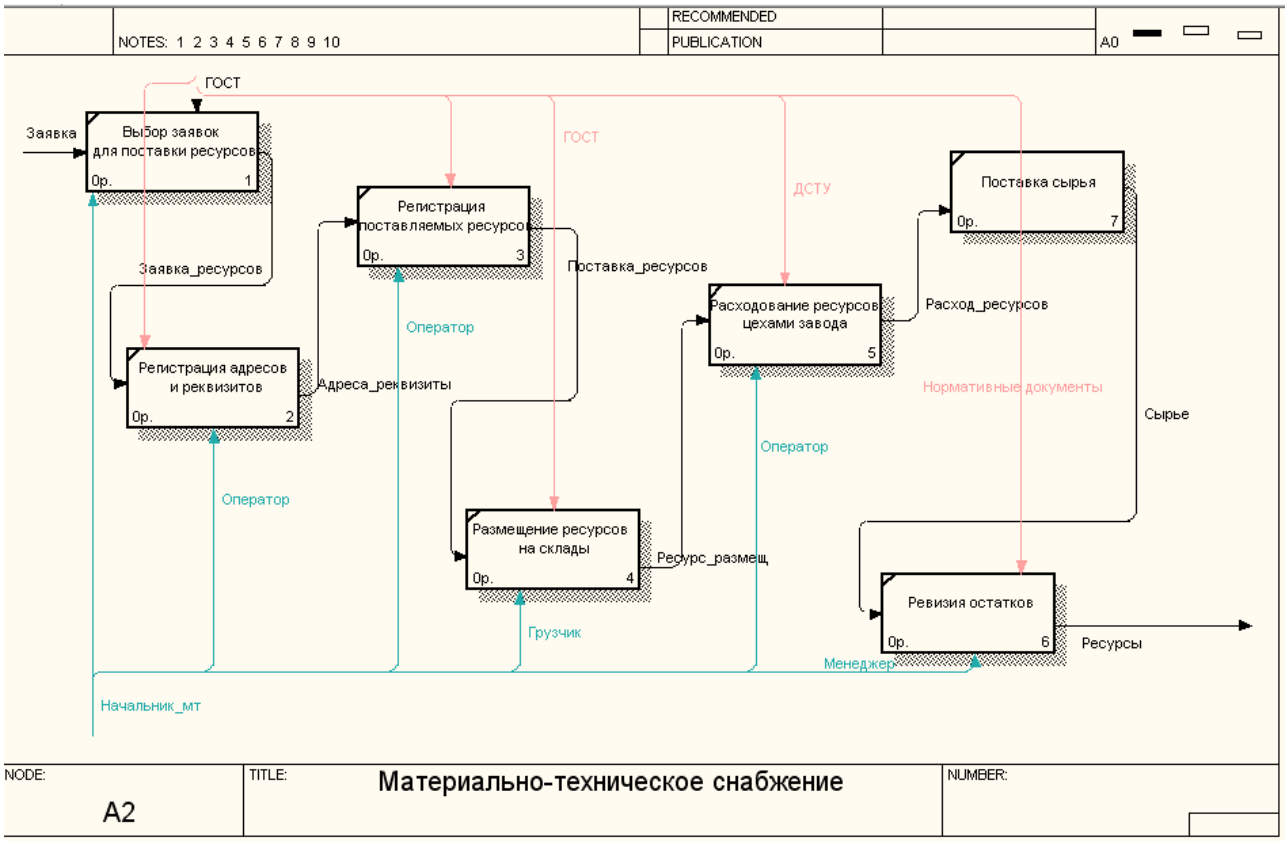


Рисунок 2.4 - Декомпозиція контекстної роботи

Роботи на діаграмі зображуються у вигляді прямокутників (функціональні блоки). Кожна робота зображує якусь функцію або завдання і називається

дієсловом або дієслівної фрази, що позначає дію, наприклад «Виготовлення виробу», «Обслуговування клієнта» і т. д. Стрілки позначаються іменником і позначають об'єкти або інформацію, що зв'язує між собою і з зовнішнім світом.

Потім кожна підсистема розбивається на більш дрібні і так до досягнення потрібного рівня подробиці. В результаті такого розбиття, кожен фрагмент системи зображується на окремій діаграмі декомпозиції. Після того як контекст описаний, проводиться побудова таких діаграм в ієрархії. Кожна наступна діаграма є більш докладним описом (декомпозицією) однієї з робіт вищестоящої на діаграмі. Опис кожної підсистеми проводиться аналітиком спільно з експертом предметної області. Зазвичай експертом є людина, що відповідає за цю підсистему і, тому, досконально знає всі її функції. Таким чином, вся система розбивається на підсистеми до потрібного рівня деталізації, і виходить модель, яка апроксимує систему із заданим рівнем точності.

Отримавши модель, що адекватно відображає поточні бізнес-процеси (так звану модель AS IS), аналітик з легкістю може побачити всі найбільш вразливі місця системи. Після цього, з урахуванням виявлених недоліків, можна будувати модель нової організації бізнес-процесів (модель TO BE).[9]

2.2.2.Методологія діаграми DFD

Для того щоб документувати механізми передачі та обробки інформації в моделюється системі, використовуються діаграми потоків даних (Data Flow Diagrams). Діаграми DFD зазвичай будуються для наочного зображення поточної роботи системи документообігу вашої організації. Найчастіше діаграми DFD використовують як доповнення моделі бізнес-процесів, виконаної в IDEF0.

Всього DFD використовує чотири важливих елементи: Роботи. Роботи в DFD позначають функції або процеси, які обробляють і змінюють інформацію. Роботи представлені на діаграмах у вигляді прямокутників з округленими кутами. (див. Рис.2.5)Стрілки. Стрілки йдуть від

об'єкта-джерела до об'єкту-приймача, позначаючи інформаційні потоки в системі документообігу . (див. Рис.2.5)

Зовнішні посилання. Зовнішні посилання вказують на місце, організацію чи людину, які беруть участь у процесі обміну інформацією з системою, але розташовуються за рамками цієї діаграми. . (див. Рис.2.5)

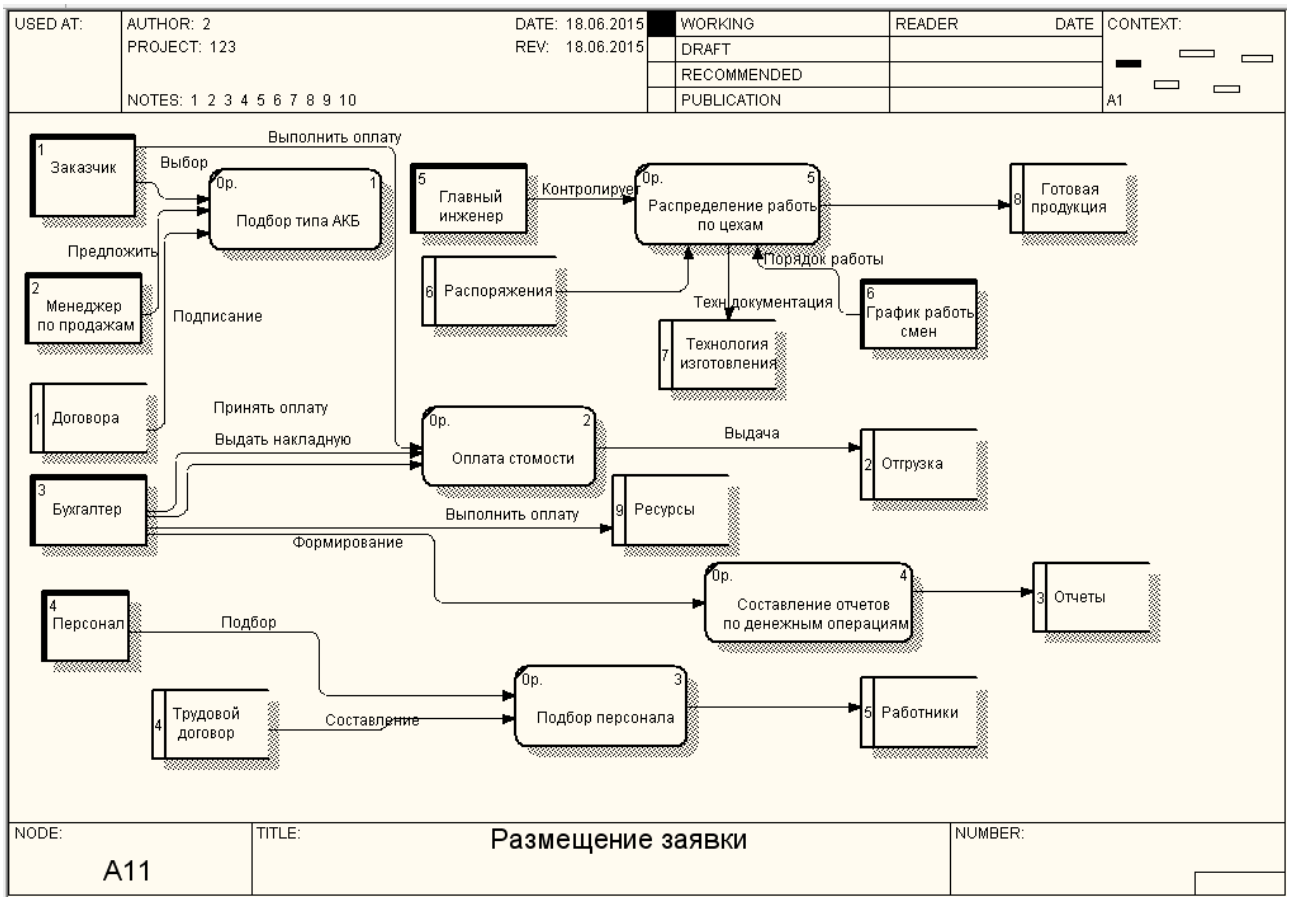


Рисунок 2.5 - Діаграма DFD

Сховища даних. Сховища даних являють собою власне дані, до яких здійснюється доступ, ці дані також можуть бути створені або змінені роботами. На одній діаграмі може бути кілька копій одного і того ж сховища даних. (див. Рис.2.5).

На відміну від стрілок IDEF0, які являють собою жорсткі взаємозв'язку, стрілки DFD показують, як об'єкти (включаючи дані) рухаються від однієї роботи до іншої.

Побудова діаграм DFD. Діаграми DFD можуть бути побудовані з використанням традиційного структурного аналізу, подібно до того, як будуються діаграми IDEF0. Спочатку будується фізична модель, що відображає поточний стан справ. Потім ця модель перетвориться в логічну модель, яка відображає вимоги до існуючої системи. Після цього будується модель, що відображає вимоги до майбутньої системи. І нарешті, будується фізична модель, на основі якої повинна бути побудована нова система.

Альтернативним підходом є підхід, популярний при створенні програмного забезпечення, званий подієвим поділом (event partitioning), в якому різні діаграми DFD вибудовують модель системи. По-перше, логічна модель будується як сукупність робіт та документування того, що вони (ці роботи повинні робити.

Потім модель оточення (environment model) описує систему як об'єкт, який взаємодіє з подіями із зовнішніх сутностей. Модель , оточення зазвичай містить опис мети системи, одну контекстну діаграму та список подій. Контекстна діаграма містить один прямокутник роботи, що зображає систему в цілому, і зовнішні сутності, з якими система взаємодіє.

Нарешті, модель поведінки (behavior model) показує, як система обробляє події. Ця модель складається з однієї діаграми, в якій кожен прямокутник зображує кожну подію з моделі оточення. Сховища можуть бути додані для моделювання даних, які необхідно запам'ятовувати між подіями. Потоки додаються для зв'язку з іншими елементами, і діаграма перевіряється з точки зору відповідності моделі оточення.

Отримані діаграми можуть бути перетворені з метою більш наочного представлення системи, зокрема роботи на діаграмах можуть бути декомпозованих.[2]

Для того щоб було простіше побудувати логічну і фізичну модель даних Erwin ми будемо експортувати діаграми IDEF0 з BPwin. [16]

1. Відкрийте модель BPwin, діаграму якої потрібно експортувати.
2. Оберіть File -> Export -> Arrows... , відкриється діалогове вікно: (Рис.2.6)

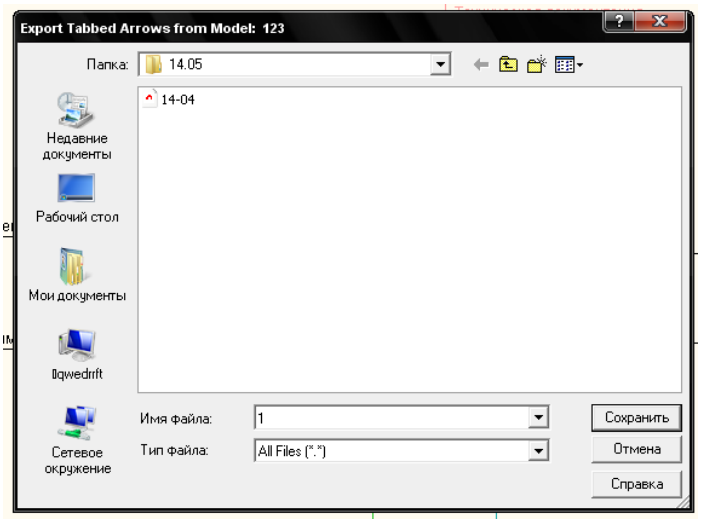


Рисунок 2.6 – Вікно експорту

3.Вказати ім'я файлу з розширенням *.csv і Зберегти. Та обрати Dictionary ->

Entity і у відкритому вікні натисніть на знак Import, відкриється вікно, де потрібно вибрати збережений раніше файл(Рис 2.7).

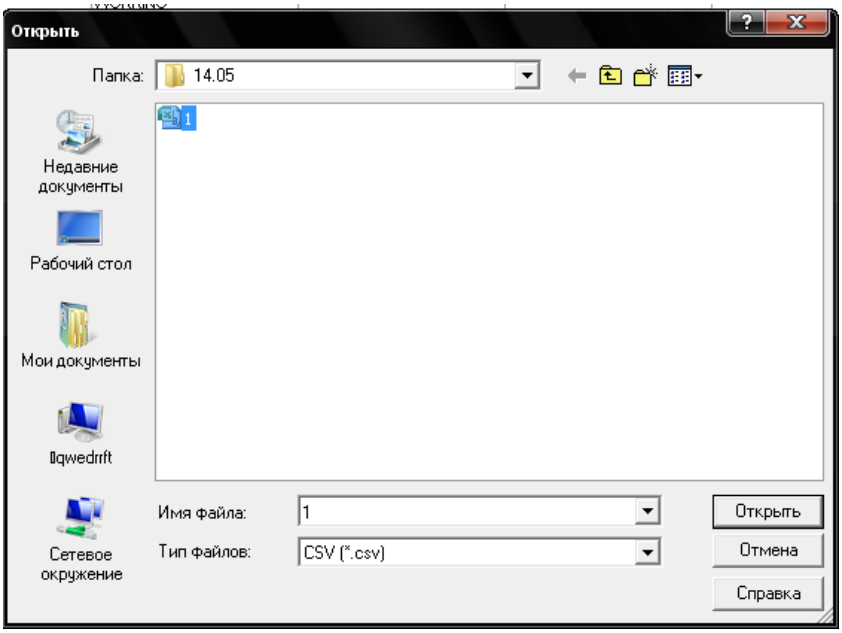


Рисунок 2.7 - Збереження файлу

4. Далі позначаємо наші сутності(Рис 2.8) і натискаємо кнопку Start.

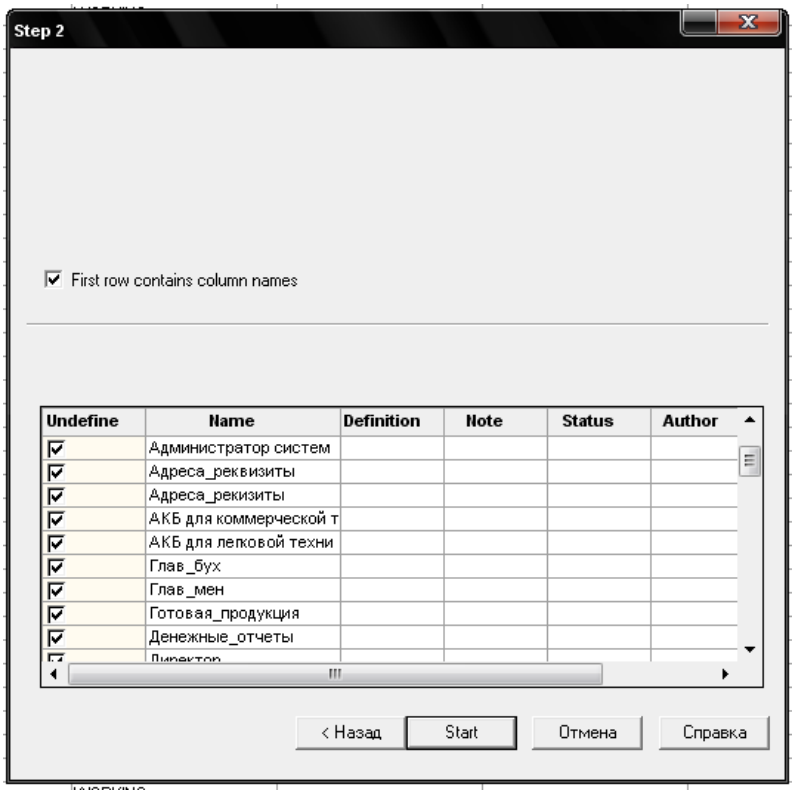


Рисунок 2.8 - Визначення потрібних сутностей

5. Повертаємося у вікно з діаграмою. Потрібно вибрати File -> Export ->Erwin(BPX). Відкриється вікно:

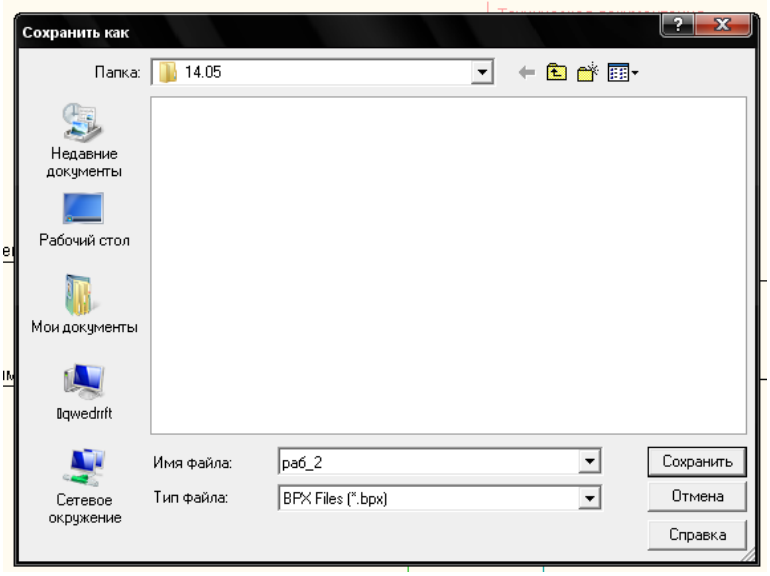


Рисунок 2.9 – Збереження файлу

Збережіть файл з розширенням *.bpx.

6. Відкриваємо Erwin і створюємо нову модель:

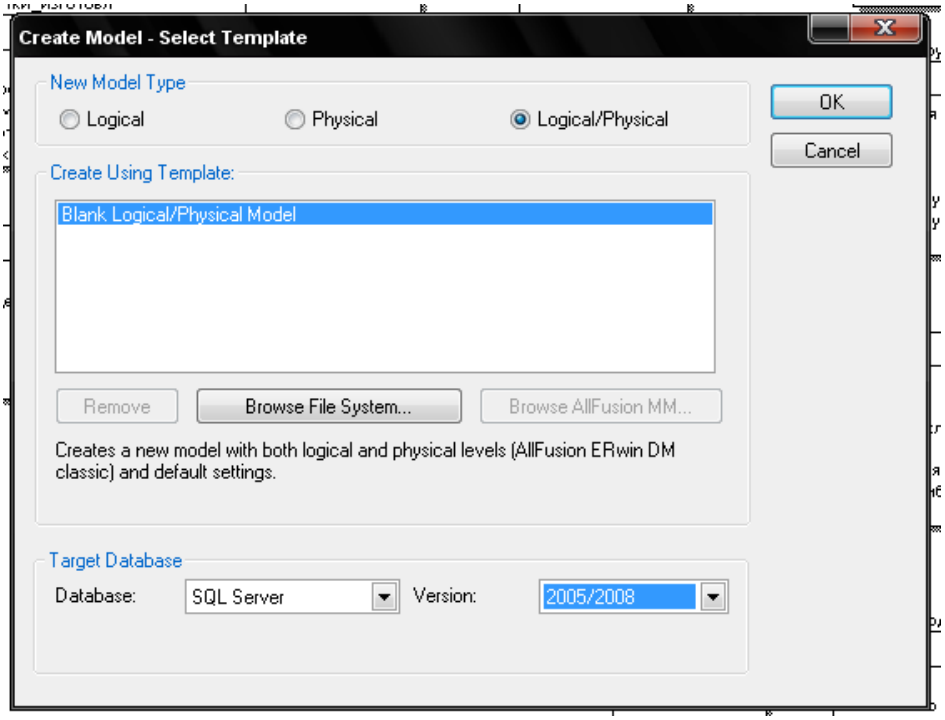


Рисунок 2.10 – Створення моделі даних

7. Оберіть File -> Import -> From Erwin Process Modeler... Відкриється діалогове вікно:

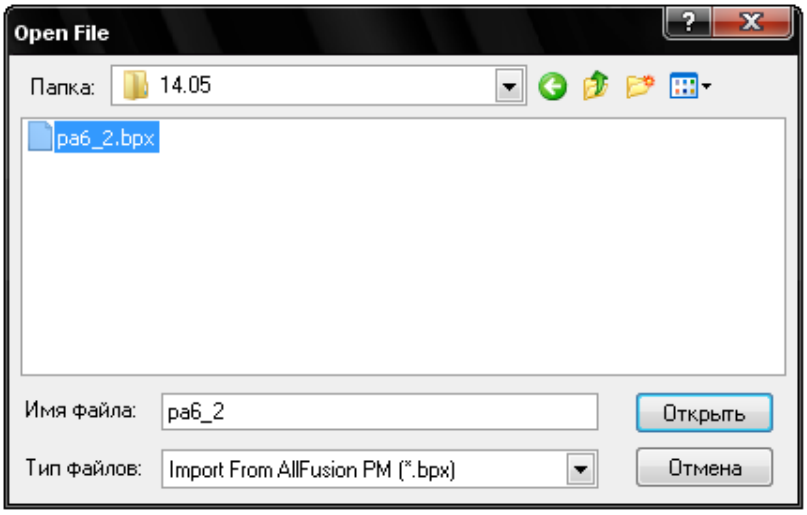


Рисунок 2.11 – Імпортування файла

8.Відкриваємо збережений файл з розширенням *.bpx. Та отримуємо наші сутності. Експорт діаграми з BPwin в ERwin завершений.

2.3 Розробка моделі даних ERwin

2.3.1. Створення логічної моделі даних

До цього раніше ми експортували діаграми в ERwin . Тепер ми заповнюємо сутності.Сутності повинні мати найменування з чітким смисловим значенням, іменуватися іменником в однині, не носити "технічних" найменувань та бути досить важливими для того, щоб їх моделювати. Іменування сутності в однині полегшує надалі читання моделі. Фактично ім'я сутності дається по імені її примірника.

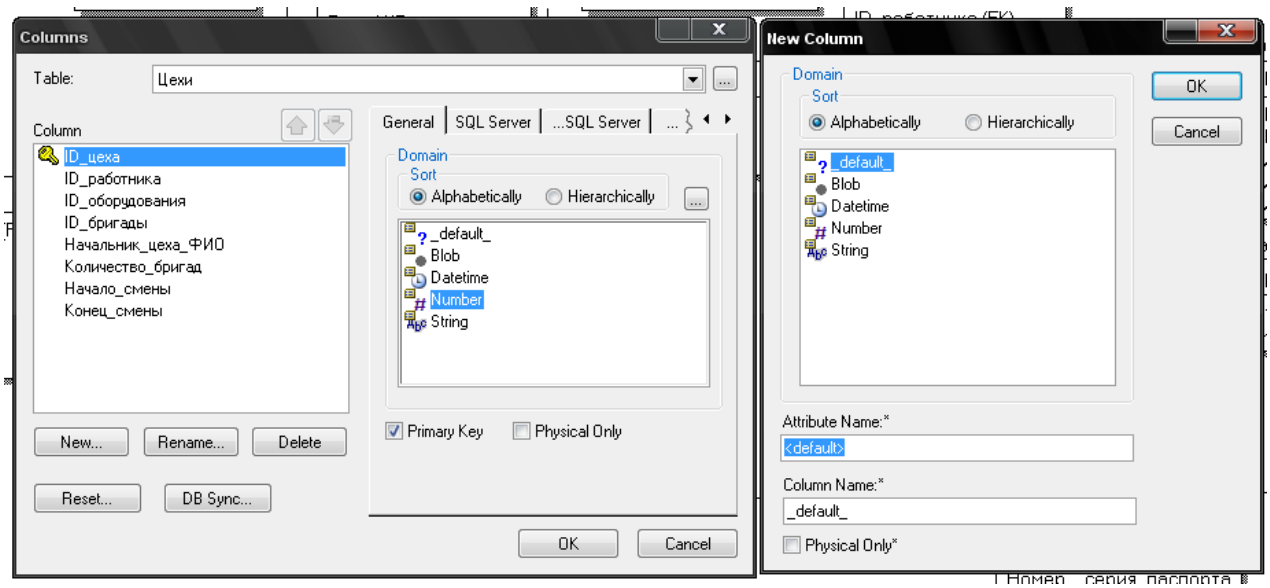


Рисунок 2.14 - Заповнення сутності

Для внесення сутності в модель необхідно (переконавшись попередньо, що ви знаходитесь на рівні логічної моделі - перемикачем між логічною і фізичною моделлю служить розкривний список у правій частині панелі інструментів) "клікнути" по кнопці сутності на панелі інструментів (ERwin Toolbox), потім "клікнути" по тому місцю на діаграмі, де необхідно розташувати нову сутність. Клацнувши правою кнопкою миші по суті і вибравши з контекстного меню пункт Entity Editor, можна викликати діалог Entity Editor, в якому визначаються ім'я, опис та коментарі сутності (рис. 2.14).

Кожна сутність повинна бути повністю визначена за допомогою текстового опису в закладці Definition. Закладки Note, Note 2, Note 3, UDP (User Defined Properties - Властивості, визначені користувачем) служать для внесення додаткових коментарів і визначень до сутності. В колишніх версіях ERwin закладок Note2 і Note3 відповідали вікна Query і Sample.

Закладка Definition використовується для введення визначення суті. Ці визначення корисні як на логічному рівні, оскільки дозволяють зрозуміти, що це за об'єкт, так і на фізичному рівні, оскільки їх можна експортувати як частину схеми і використовувати в реальному БД (CREATE COMMENT on entity_name).

Закладка Note дозволяє додавати додаткові зауваження щодо сутності, які не були відображені у визначенні, введеному в закладці Definition. Тут можна ввести корисне зауваження, що описує будь-який бізнес-правило або угоду з організації діаграми.

В закладці Note 2 можна задокументувати деякі можливі запити, які, як очікується, будуть використовуватися по відношенню до сутності в БД. При переході до фізичного проектування, записані запити допоможуть приймати такі рішення щодо проектування, які зроблять БД більш ефективною.

Також ми повинні створити зв'язку між сутностями (див. рис2.15).

Денормалізація. В результаті нормалізації всі взаємозв'язки даних стають правильно визначено, виключаються аномалії при оперуванні з даними, модель даних стає легше підтримувати. Однак часто нормалізація даних не веде до підвищення продуктивності ІС в цілому.

Для приведення сутності до першої нормальної формі слід:розділити складні атрибути на атомарні, створити нову сутність, перенести в неї все "повторювані атрибути, вибрати можливий ключ для нового РК (або створити новий РК), встановити ідентифікує зв'язок від колишньої сутності до нової, РК колишньої сутності стане зовнішнім ключем (FK) для нової сутності.

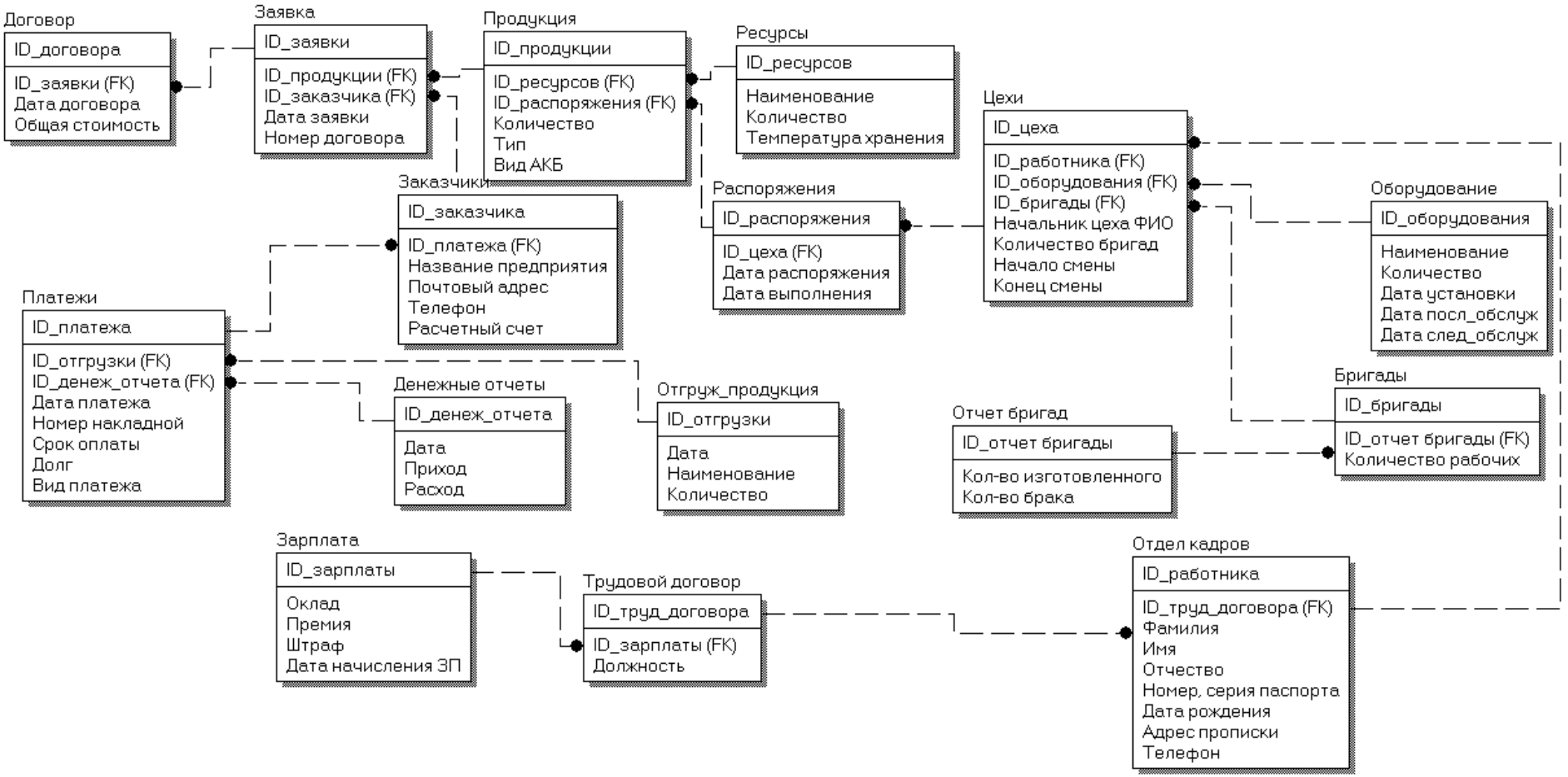


Рисунок 2.15 – Логічна діаграма ERD

2.3.2 Фізична модель даних

Для переходу в логічну модель даних використовуємо зверху випливає вікно де ми і вибираємо Physical.

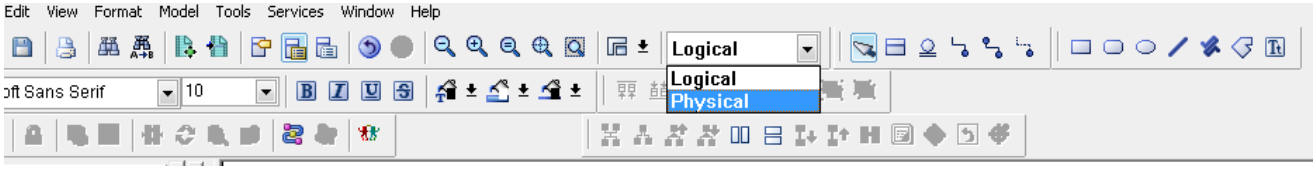


Рисунок 2.16 –Перехід від логічної до фіз. моделі даних

Зовні діалог Column Editor нагадує діалог Attribute Editor (див. рис. 2.17). У правій частині діалогу знаходяться закладки: General. Дозволяє присвоїти колонку певного домену, створити колонку тільки на фізичному рівні і включити її до складу первинного ключа. Закладка, відповідна обраної СУБД (на рис. 2.17 - SQL Server Ім'я закладки встановлюється автоматично відповідної вибраної СУБД.

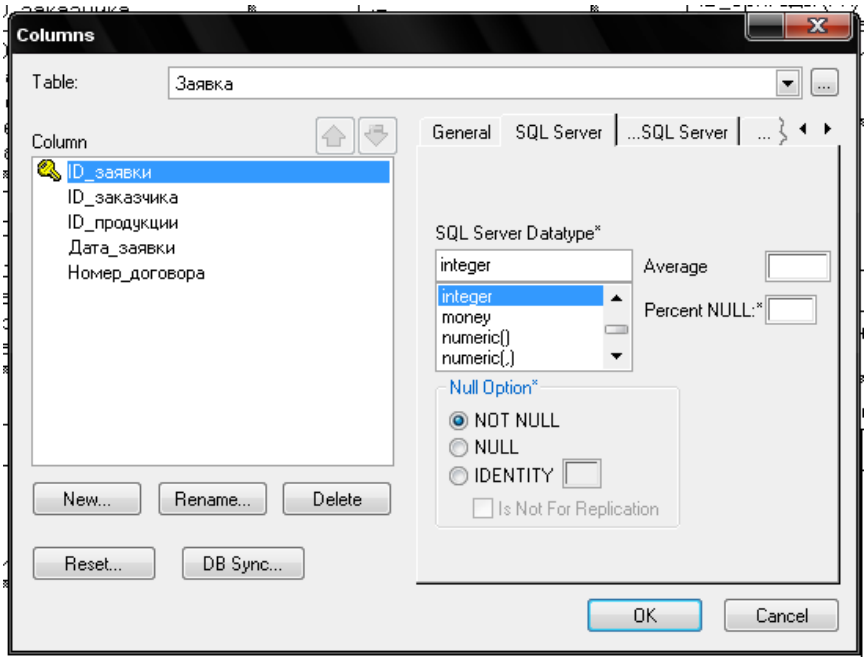


Рисунок 2.17 – Визначення атрибутів

Дозволяє задати тип даних, опцію NULL, правила валідації і значення за замовчуванням. Правила валідації і значення за замовчуванням повинні бути описані і іменовані попередньо відповідно в діалогах Validation Rule Editor і Default/Initial Eritor. Для виклику цих діалогів служать кнопки праворуч від відповідних списків, що розкриваються. Для СУБД Access, AS/400,

PROGRESS та Teradata створюються додаткові дані для завдання властивостей.

2.4 Ініціалізація SQL - скрипта

Після заповнення всіх сутностей, визначення зв'язків і типів даних, ми можемо ініціалізувати SQL-скрипт, і таким чином вивантажимо нашу логічну модель даних на сервер.

Для цього потрібно вибрати Tools - Forward Engineer - Schema Generation.

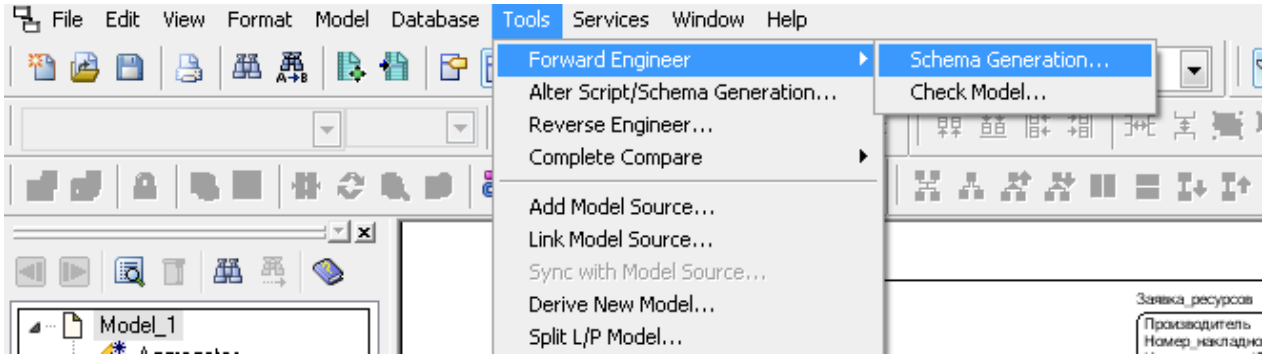


Рисунок 2.18 Перехід до генерації коду

Потім ми обираємо потрібні нам поля атрибутів і натискаємо кнопку Generate

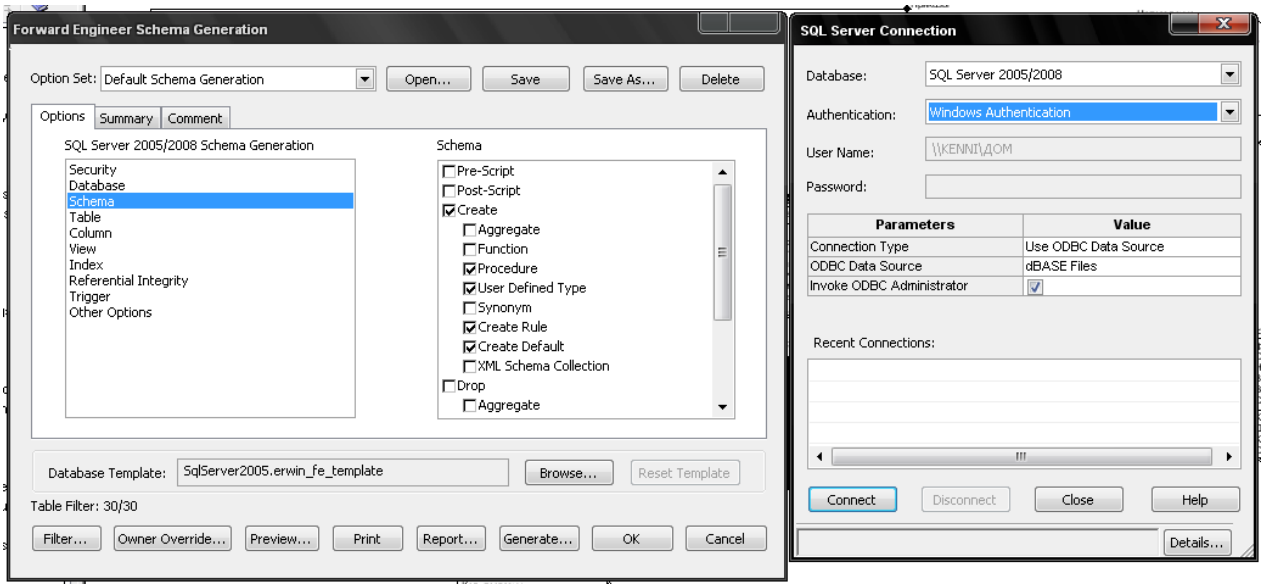


Рисунок 2.19 – Генерація SQL – скрипту

На рисунку 2.19 праворуч з'явилося вікно підключення до SQL server виставляємо потрібні нам поля для з'єднання з сервером і натискаємо кнопку Connect якщо все було зроблено правильно, то наші атрибути та типи даних будуть згенеровані у вигляді SQL – скрипта, який представлений у додатку А.

Також, якщо у вас не вийде згенерувати SQL - скрипт через SQL Server Connection ми можемо його отримати за допомогу звіту(кнопка Report...), тобто генерувати SQL-скрипт у вигляді sql файлу. Просто вказуємо потрібну нам папку для збереження SQL скрипта, потім вручну підключаємо його до нашої БД на сервері. Код генерації SQL-скрипта [ДОДАТОК А]

РОЗДІЛ 3
ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Характеристики серверу

Для спокійного та безперебойного використання серверу, рекомендується використистовувати наступні характеристики :

Xeon E3 1230 - це чотирьохядерний серверний процесор з підтримкою HyperThreading, що дає 8 логічних процесорів в цілому.

Платформи для вбудованих систем на базі процесорів Intel Xeon і наборів мікросхем забезпечують наступні можливості:

• Висока продуктивність процесора

• двопроцесорним конфігурації з багатоядерними процесорами і високою компактністю

• Надійність, відмовостійкість і зручність обслуговування пам'яті

• Сумісність з NEBS

• Підтримка апаратного управління, захисту, віртуалізації і інтелектуального управління живленням

Модуль пам'яті для комп'ютера DDR4 2x8GB 2400 MHz Phoenix Series eXceleram (EPH4082417A) - Оперативна пам'ять комп'ютера або ноутбука - це один з найважливіших елементів системи, який безпосередньо відповідає за швидкодію додатків. Саме в ній зберігаються тимчасові дані і команди, які необхідні процесору в його роботі для виконання конкретних операцій.

Основні характеристики - це тип оперативної пам'яті;

Об `єм; тактова частота;

тайминги (або латентність) оперативної пам'яті;

фірма виробник.

AMD Radeon R5 2x480GB 2.5” SATA III TLC

Об’єм: 480 гб

Форм-фактор: 2.5”

Інтерфейс підключення: SATA III

Швидкість читання: 528 мб/с

Швидкість запису: 448 мб/с

Тип ячійки пам’яті: TLC

Також додатково встановлений ще один блок живлення та аппаратний RAID-контролер + BBU

3.2. Призначення і умови застосування вибранного серверу

Цей сервер призначенний для підтримки бази данних підприємства ІСТА, для безперебійної роботи та без втрати даних під час роботи.

Дата-центр[17] (від англ. Data center), або центр (зберігання і) обробки даних (ЦОД / ЦХОД) - це спеціалізоване будівлю для розміщення (хостингу) серверного та мережевого обладнання та підключення абонентів до каналів мережі Інтернет.

Дата-центр виконує функції обробки, зберігання і розповсюдження інформації, як правило, в інтересах корпоративних клієнтів - він орієнтований на вирішення бізнес-завдань шляхом надання інформаційних послуг. Консолідація обчислювальних ресурсів і засобів зберігання даних в ЦОД дозволяє скоротити сукупну вартість володіння IT-інфраструктурою за рахунок можливості ефективного використання технічних засобів, наприклад, перерозподілу навантажень, а також за рахунок скорочення витрат на адміністрування.

Дата-центри зазвичай розташовані в межах або в безпосередній близькості від вузла зв'язку або точки присутності якого-небудь одного або декількох операторів зв'язку. Основним критерієм оцінки якості роботи будь-

якого дата-центру є час доступності сервера (аптайм).

Надійність[18]

Серверне обладнання найчастіше призначене для забезпечення роботи сервісів в режимі 24/7, тому часто комплектується дублюючими елементами, що дозволяють забезпечити «п'ять дев'яток» (99,999%; час недоступності сервера або простий системи становить менше 6 хвилин в рік). Для цього конструкторами при створенні серверів створюються спеціальні рішення, відмінні від створення звичайних комп'ютерів:

- пам'ять забезпечує підвищену стійкість до збоїв. Наприклад для i386-сумісних серверів, модулі оперативної пам'яті і кеша має посилену технологію корекції помилок (англ. Error Checking and Correction, ECC). На деяких інших платформах, наприклад SPARC (Sun Microsystems), корекцію помилок має вся

  пам'ять. Для власних мейнфреймів IBM розробила спеціальну технологію Chipkill ™.

- Підвищення надійності сервера досягається резервуванням, в тому числі з гарячими підключенням і заміною (англ. Hot-swap) критично важливих компонентів:
- при необхідності вводиться дублювання процесорів (наприклад, це важливо для безперервності виконання сервером завдання довготривалого розрахунку - в разі відмови одного процесора обчислення не обриваються, а тривають, нехай і на меншій швидкості)
- блоків живлення,
- жорстких дисків в складі масиву RAID і самих контролерів дисків,
- груп вентиляторів, що забезпечують охолодження компонентів сервера.
- У функції апаратного моніторингу вводять додаткові канали для контролю більшої кількості параметрів сервера: датчики температури контролюють температурні режими всіх процесорів, модулів пам'яті, температуру в відсіках з встановленими жорсткими дисками; електронні лічильники імпульсів, вбудовані в вентилятори, виконують функції тахометрів і дозволяють, залежно від температури, регулювати швидкість їх обертання; постійний контроль напруги живлення компонентів сервера дозволяє сигналізувати про ефективність роботи блоків живлення; сторожовий таймер не дозволяє залишитися непоміченим зависання системи, автоматично виробляючи примусову перезавантаження сервера.

Умови застосування вибранного серверу

- Окреме приміщення
- Вставновлення серверу не у вологому місці
- Захист кабелів, розеток, аппаратури від замикання
- Допуск до серверу лишь адміністратору
- Після кожного робочого дня проводити перевірку справності серверу
- Кожного місяця проводити повну перевірку справності серверу

3.3 Підключення та створення БД на SQL сервері

Для роботи з MS SQL сервером потрібна аутентифікація.

Рисунок 3.1 – Вікно Connect to Server

У діалоговому вікні Connect to Server (Підключитися до сервера), показаному на рис. 3.1, використовуйте розкривний список Server type (Тип сервера) для вибору компонент БД, з яким потрібно встановити з'єднання, наприклад Database Engine (Ядро бази даних).

В полі Server name (Ім'я сервера) введіть ім'я комп'ютера, на якому запущений SQL Server. У розкривному списку Server name (Ім'я сервера) можна вибрати ім'я екземпляра, з яким раніше вже встановлювалося з'єднання. Щоб підключитися до будь-якого примірника у перший раз, у цьому ж списку виберіть пункт <Browse for more...> (Пошук інших серверів). У діалоговому

У списку Authentication (Аутентифікація) виберіть один з типів аутентифи-ції — Windows Authentication (Автентифікація Windows) або SQL Server Authentication (Аутентифікація SQL Server). При необхідності в полі User name) введіть ім'я облікового запису Windows або SQL Server, а в полі Password.

Windows Authentication (Аутентифікація Windows) для підключення до сервера використовується обліковий запис домену Windows і її пароль. Застосовується тільки в тому випадку, якщо включений режим

аутентифікації Windows і обліковий запис Windows має відповідні права на підключення до SQL Server.

SQL Server Authentication (Аутентифікація SQL Server) Для підключення до сервера використовується обліковий запис SQL Server і її пароль. Щоб не доводилося вводити пароль кожного разу при підключенні, встановіть фляжок Remember password (Запам'ятати пароль).

Якщо замість бази даних за замовчуванням ви хочете підключитися до іншої, щелкните кнопку Options (Параметри), відкрийте вкладку Connection Properties (Сства з'єднання) і у спадному списку Connect to database (Підключитися до бази даних виберіть БД.

Клацніть на кнопці Connect (Підключитися).

Щоб відкрити вікно формування запитів в SQL Server Management Studio, в панелі інструментів Standard (Стандартна) клацніть кнопку необхідного типу запиту, наприклад Database Engine Query (Запит ядра бази даних). Відобразиться діалогове вікно Connect to... (Підключитися до...), аналогічне позначеного на поле Server type (Тип сервера) з'являється вже з перед певним значенням, яке залежить від вибраного типу запиту). Далі потрібно пройти процес підключення, описаний вище, в пунктах 3-6.

Якщо в SQL Server Management Studio вже є з'єднання з сервером і вибрана активна база даних, можна підключитися до поточного екземпляра сервера, застосувавши для входу поточну інформацію аутентифікацію. Для цього в панелі Project Explorer (Оглядач об'єктів) в контекстному меню сервера, виконайте команду New Query (Створити запит) або ж в панелі інструментів Standard (Стандартна) клацніть кнопку New Query (Створити запит). Як правило, вікно SQL Management Studio при роботі з запитами розділене на три частини (рис. 3.2). Зліва зазвичай відображаються панелі, що дозволяють переглядати доступні об'єкти на примірнику сервера БД, вибраному в даний час Registered Servers (Зареєстровані сервери) і Object Explorer (Оглядач об'єктів). В області вгорі праворуч можна ввести запит, а в області нижче — переглянути результат його виконання.

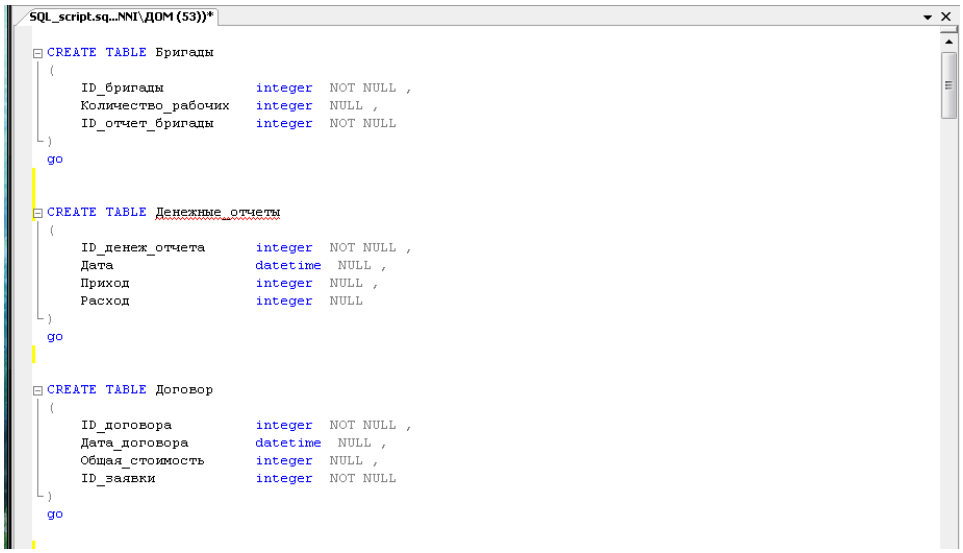


Рисунок 3.2 - Вікно запитів

Як вам вже відомо, для перегляду і зміни параметрів конфігурації SQL Server застосовується збережена процедура sp_configure. Доступні два типи параметрів конфігурації: динамічні та не динамічні. Стосовно до нашого випадку, динамічним параметром є той, який можна змінити без зупинки і запуску SQL Server. Для виконання збереженої процедури sp_configure або запитів інших типів введіть в верхній частині вікна запитів, потім натисніть кнопку Execute (Виконати) у панелі інструментів (червоний знак оклику).

Після того як ми підключилися до сервера і створили БД тепер нам необхідно завантажити наш SQL-скрипт на сервер. Вибираємо створену БД і натискаємо New Query – для створення нового запиту(Рис 3.3) і відкриваємо наш раніше збережений файл через Open File(Рис.3.4).

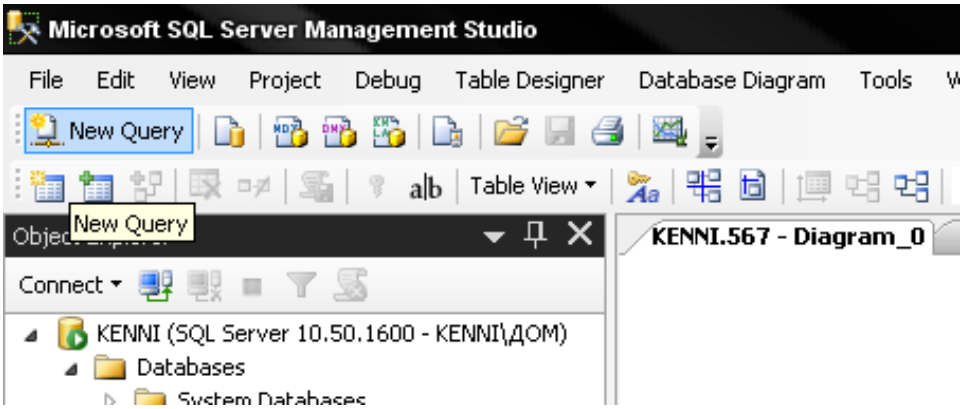


Рисунок 3.3 - Створення запиту

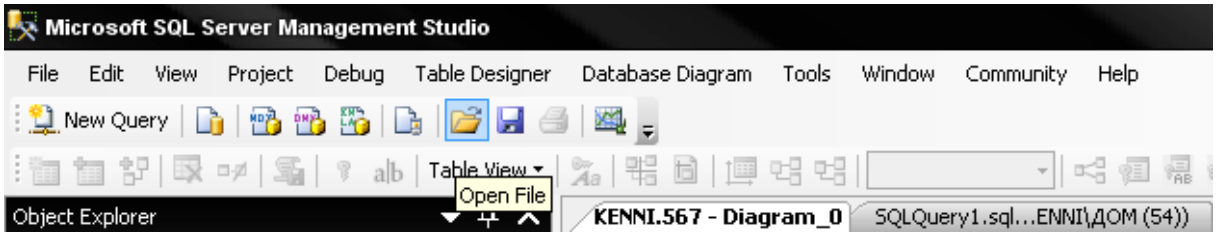


Рисунок 3.4 - Відкриття sql-файла

Після відкриття SQL-скрипта запускаємо його(Рис.3.5). Якщо все зроблено правильно, то ми отримаємо таблиці з типами даних, первинними ключами і зв'язками створених раніше в Erwin Data Modeler.

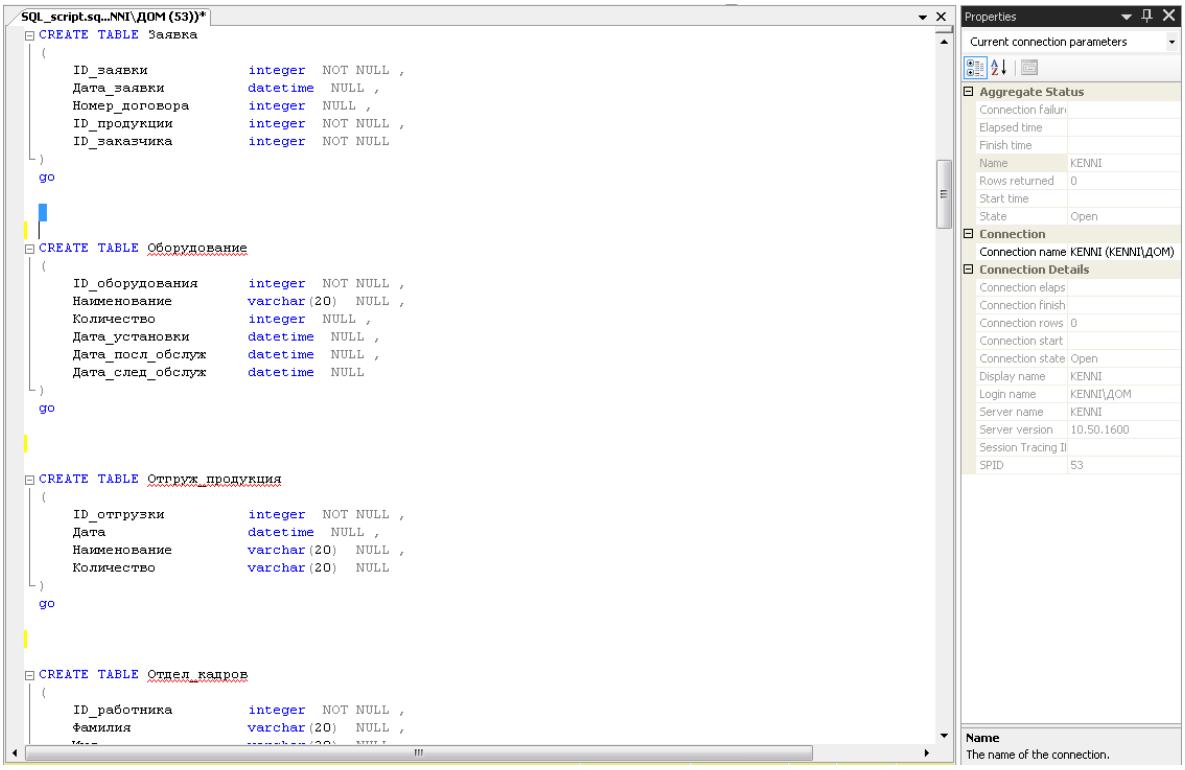


Рисунок 3.5 - Запуск SQL-скрипта

Заходимо в Database Diagrams створюємо діаграму БД New Database Diagram (Рис 3.6) якщо не перенеслися визначаємо і перевіряємо зв'язку між сутностями. Реляційні бази даних дозволяють поєднувати інформацію, що належить різним сутностей. Сутності визначаються в процесі проектування бази. Для цього слід проаналізувати сутність і виявити логічні зв'язки, що

існують між ними. Тип відносини визначає кількість записів сутності, пов'язаних із записом іншої сутності. Відносини поділяються на три основних типи:

один-до-одного

один-до-багатьох

багато-до-багатьох

Проводимо нормалізацію і заповнюємо таблиці даними(Рис 3.6, 3.7).

Нормалізацією називається процес видалення надлишкових даних з бази даних. Кожен елемент повинен зберігатися в базі в одному і тільки в одному примірнику. Існує п'ять найпоширеніших форм нормалізації. Як правило, база даних приводиться до третьої нормальної формі.

У процесі нормалізації виконуються певні дії по видаленню надлишкових даних. Нормалізація підвищує швидкодію, прискорює сортування і побудова індексу, зменшує кількість індексів на сутність, прискорює операції вставки та оновлення.

Нормалізована база даних зазвичай відрізняється більшою гнучкістю. При модифікації запитів або збережених даних в нормалізовану базу зазвичай доводиться вносити менше змін, а внесення змін має менше наслідків.

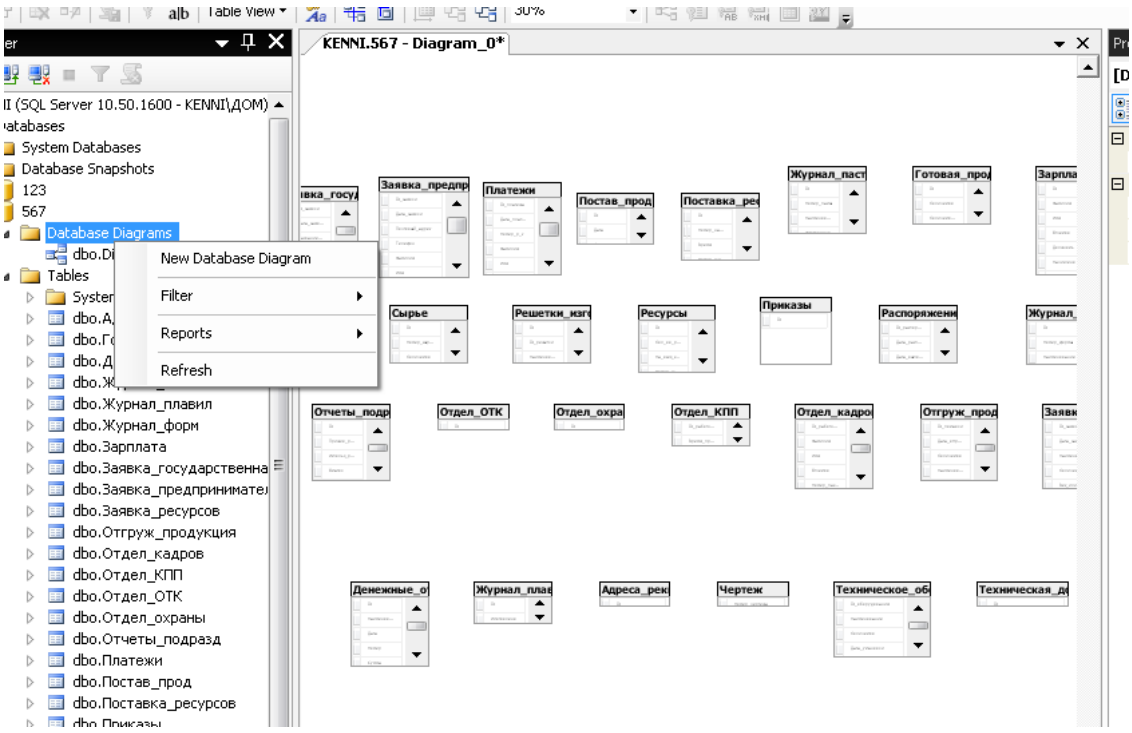


Рисунок 3.6 - Отримана сукупність сутностей на сервері

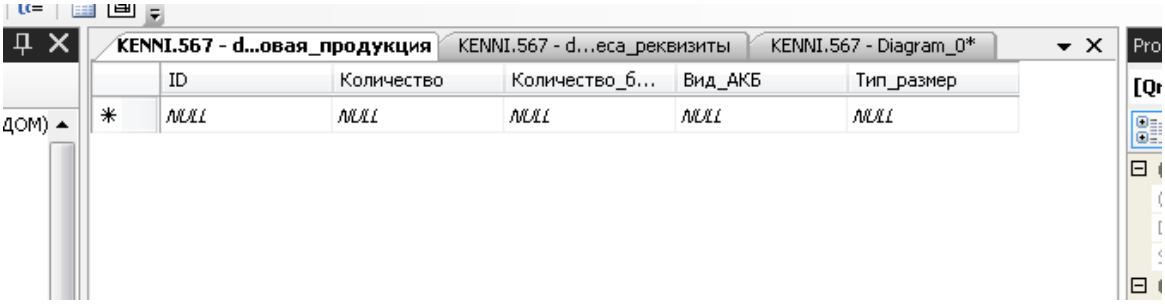


Рисунок 3.7 Заповнення таблиць БД

Після заповнення таблиць даними, в кожній таблиці визначаємо первині ключи та зв’язуємо таблиці між собою.

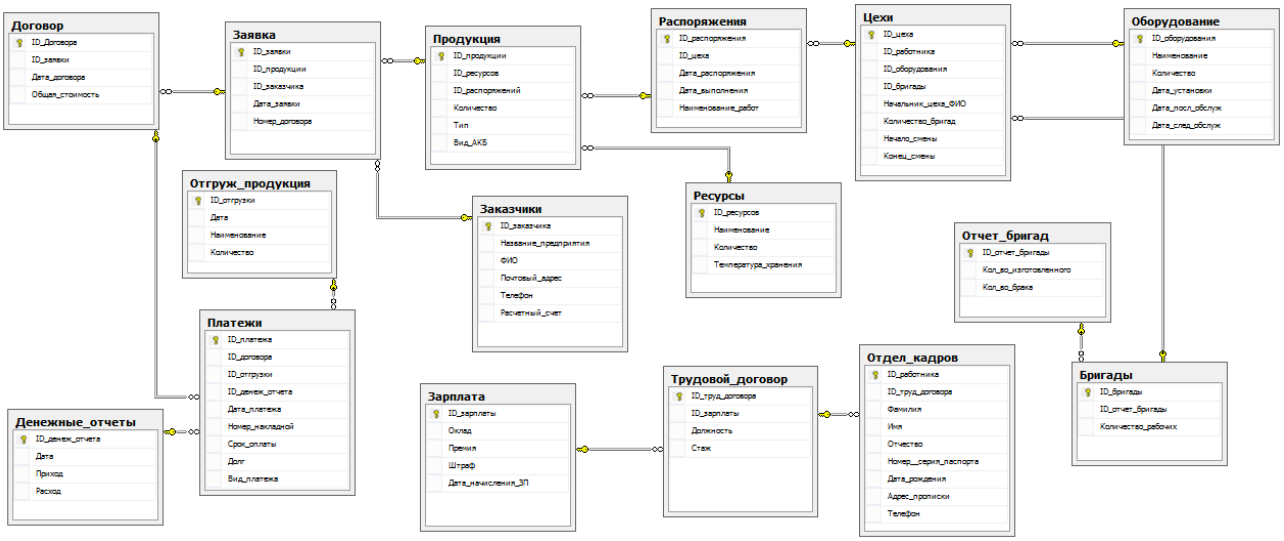


Рисунок 3.8 Схема даних

4.1 Розробка додатка клієнта та тестування системи

Delphi – це середовище швидкої розробки, в якій в якості мови програмування використовується мова Delphi. Мова Delphi – строго типізована об'єктно орієнтована мова, в основі якої лежить добре знайомий програмістам Object Pascal.

Спочатку створимо новий проект і збережемо його в окрему папку. У полі Form1.Caption напишемо(Рис.4.1) Вход в систему - цей текст буде заголовком головного вікна.

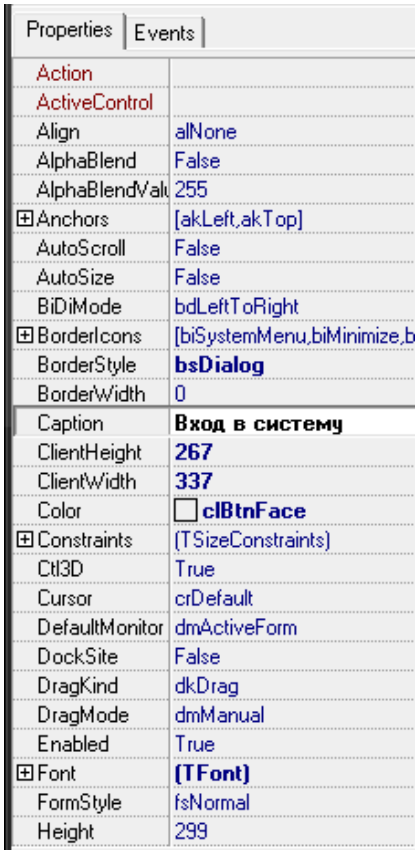
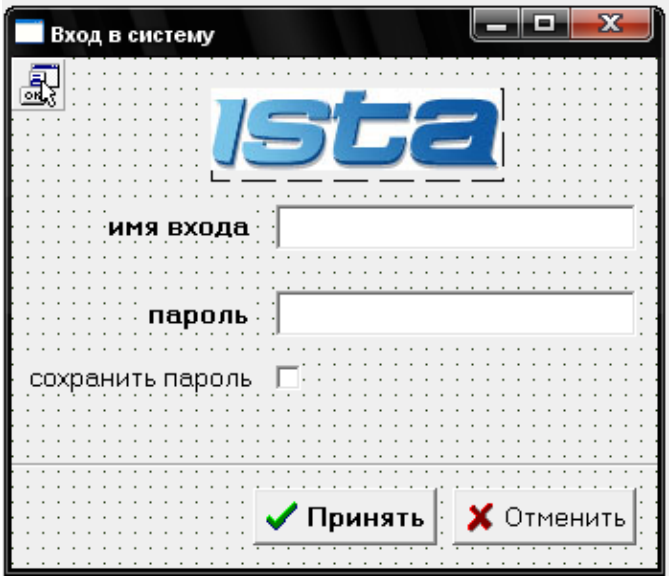


Рисунок 4.1 - Поле Caption

Потім помістимо на форму компоненти Label та Edit з вкладки Standart. Перша форма – це форма авторизації користувача в программі(Рис 4.2). Добавимо BitBtn з вкладки Additional це будуть наші кнопки для підтвердження або скасування авторизації. Ще додаємо CheckBox - для запам'ятовування паролю та Image – значок зображення нашого проекту, потім в полі Unit – FormStart описуємо компоненти програми для реалізації.



4.2 – Форма авторизації

Створюємо головну форму програми яка містить в собі таблиці, контакти, структури, та історію підприємства. форма має вигляд(Рис. 4.3). Інтерфейс користувача починається через головне меню, яке зв'язано з SQL Server 2008 через ADOConnection. Встановлення форм на екрані здійснюється через команди File|New Form або File|New Data Module. На формі знаходяться компоненти DataSourse (Джерело даних) і ADOtable (Таблиці). Компонент ADOtable представляє набір даних, який може бути пов'язаний з однією таблицею БД.

Також розміщені компоненти MainMenu та PopupMenu (спливаюче меню), які викликаються в додатку клацанням правою кнопкою миші. Контекстне меню (MainMenu) є стандартною і зручною можливістю багатьох програм. У меню відбитий перелік команд-дій, які може виконати система. Якщо користувач планує дізнатися про продукцію, яка знаходиться на складі, то він з меню викликає форму готової продукції.

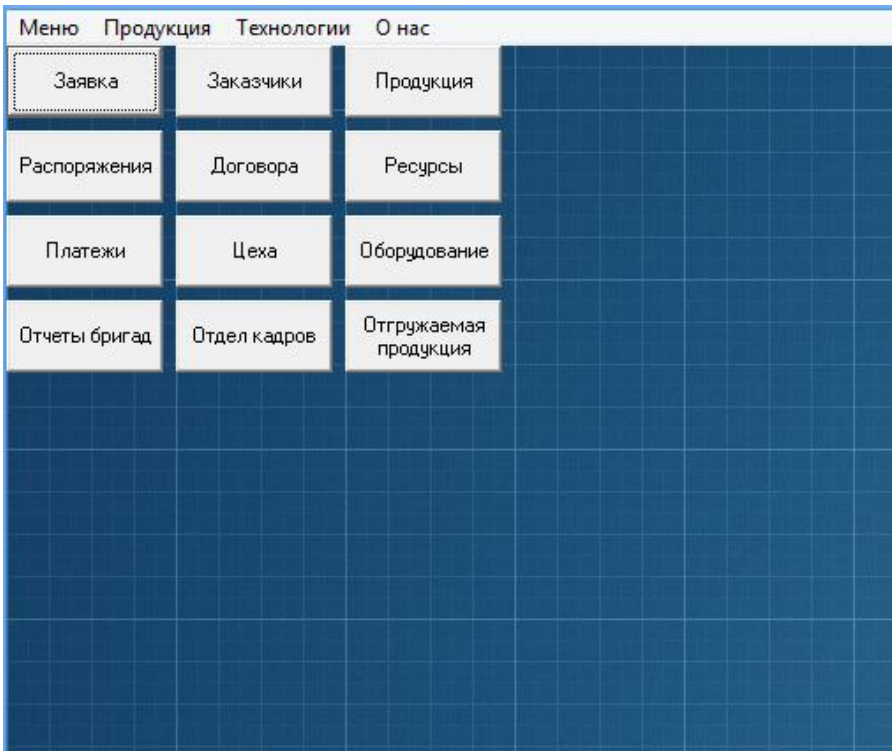


Рисунок 4.3 – Головна форма додатку клієнта

На головній формі добавляємо вікно панелі меню, та кнопки які буду відображати таблиці. Спочатку побудуємо екранну форму шляхом вибору команди меню View|Forms. Розташуємо у формі компонент TDBGrid, узявши його з палітри компонентів (сторінка Data Controls). Встановимо властивість DataSource компонента TDBGrid в значення DataSource заздалегідь додавши ім'я модуля Unit, в якому описаний TdataModule. Компонент TDBGrid служить для відображення записів набору даних в табличній формі. Додамо компонент DBLookupComboBox (випадаючий список), де виберемо для віддображення поле Назва цеху. Додамо у форму компонент кнопки TButton (сторінка Standard палітри компонентів). Змінимо заголовок кнопки (властивість Caption) на Зачинити. Форма представлена на рисунку 4.4.

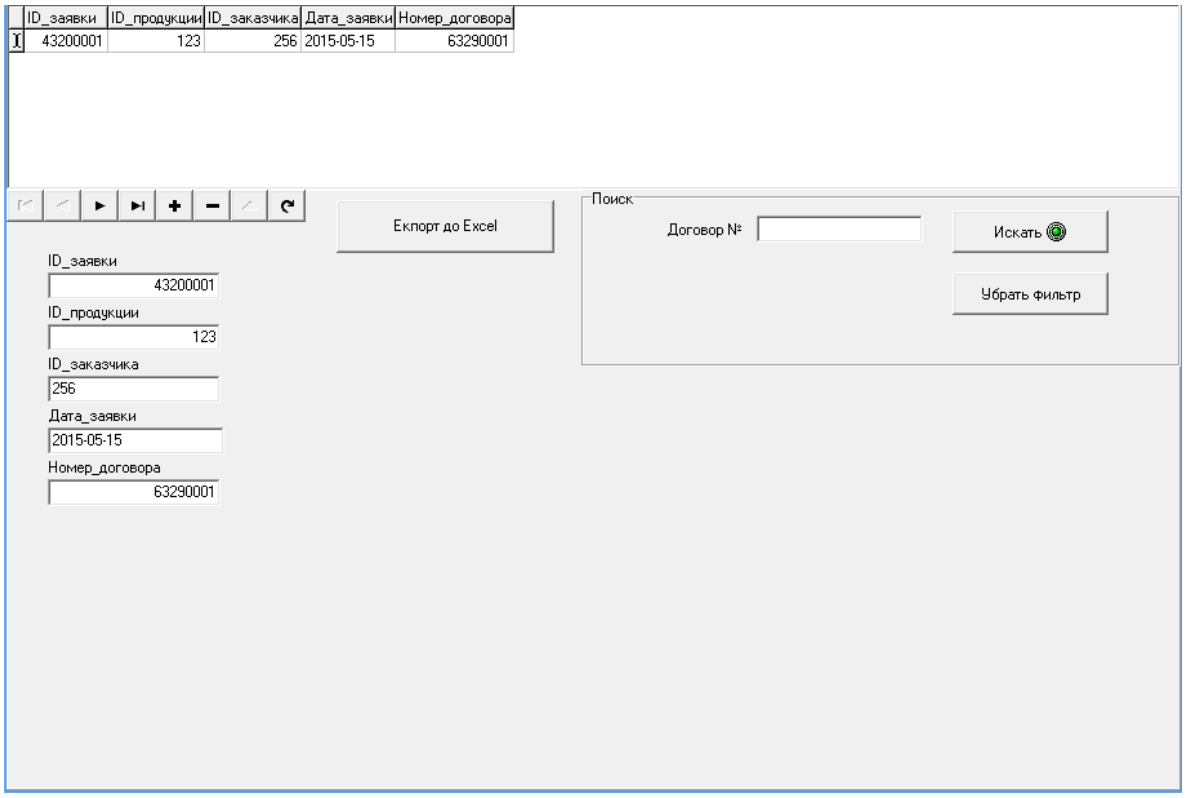


Рисунок 4.4 - Екранна форма Заявка

Для зміни запису слід перемістити покажчик поточного запису в потрібне місце і змінити значення там, де це необхідно. Набор даних автоматично перейде в режим редагування.

На екранній формі можна створити набір кнопок, які дозволять проводити навігацію зображену на рис 4.5 (TDBNavigator) по таблицях баз даних і управляти її записами. Перерахуємо їх назви і призначення, починаючи з лівої кнопки:

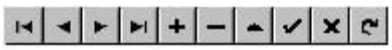


Рисунок 4.5 - Панель (TDBNavigator)

Приклад останніх форм наведений у Додатку Н.

Запуск системи здійснюється по команді: Run|Run. Запуск системи можна здійснити і за допомогою гарячої клавіші F9.

В разі зависання системи або за відсутності виходу з системи переривання роботи здійснюється по команді Run|Program Reset.

Після переривання роботи системи можна продовжувати її розробку або відладку.

Код програми представлений у [ДОДАТОК Б]

Таблиця 4.1. Опис кнопок панелі TDBNavigator

nbFirst	Переміщення до першого запису
nbPrior	переміщення до попереднього запису
nbNext	переміщення до наступного запису
nbLast	переміщення до останнього запису
nbInsert	вставити новий запис перед поточним
nbDelete	видалити поточний запис
nbEdit	редагувати поточний запис
nbRefresh	очистити буфер, пов'язаний з набором даних

Delphi дозволяє приєднувати текстові документи. Функція ShellExecute призначена для відкриття або друку файлу, документа:

Shellexecute(Handle,'open','text\dogovor.doc',nil,nil,SW_SHOW);

Перший параметр це handle батьківського вікна, другий параметр - рядок, який вказує, що треба зробити з файлом, третій параметр містить ім'я файлу, четвертий- вказує додаткові параметри запуску виконуваного файлу, п'ятий параметр визначає директорію за умовчанням, останній параметр визначає де буде відображатися файл після відкриття(рис. 4.6):

Трудовий договір з працівником

м._____ « »_____20_г.

1. Підприємство (організація)_____
(Найменування)

у особі _____,
(Посада, Ф.І.О.)

іменоване надалі «Працедавець», і громадянин _____,
(Ф.І.О.)

іменований надалі «Працівник», уклали цей договір про нижченаведене:

2. Працівник_____
(Ф.І.О.)

приймається на роботу_____
(Найменування структурного підрозділу підприємства: цех, відділ, лабораторія і т. д.)

по професії, посаді_____
(Повне найменування професії, посади)

кваліфікації_____
(Розряд, кваліфікаційна категорія)

3. Договір є: договором по основній роботі
договором за сумісництвом
(Потрібне підкреслити)

4. Вигляд договору:
- на невизначений термін (безстроковий)
- на певний термін_____
(Вказати причину укладення термінового договору)
- на час виконання певної роботи_____
(Вказати який)

5. Термін дії договору: Початок роботи_____
Закінчення роботи_____

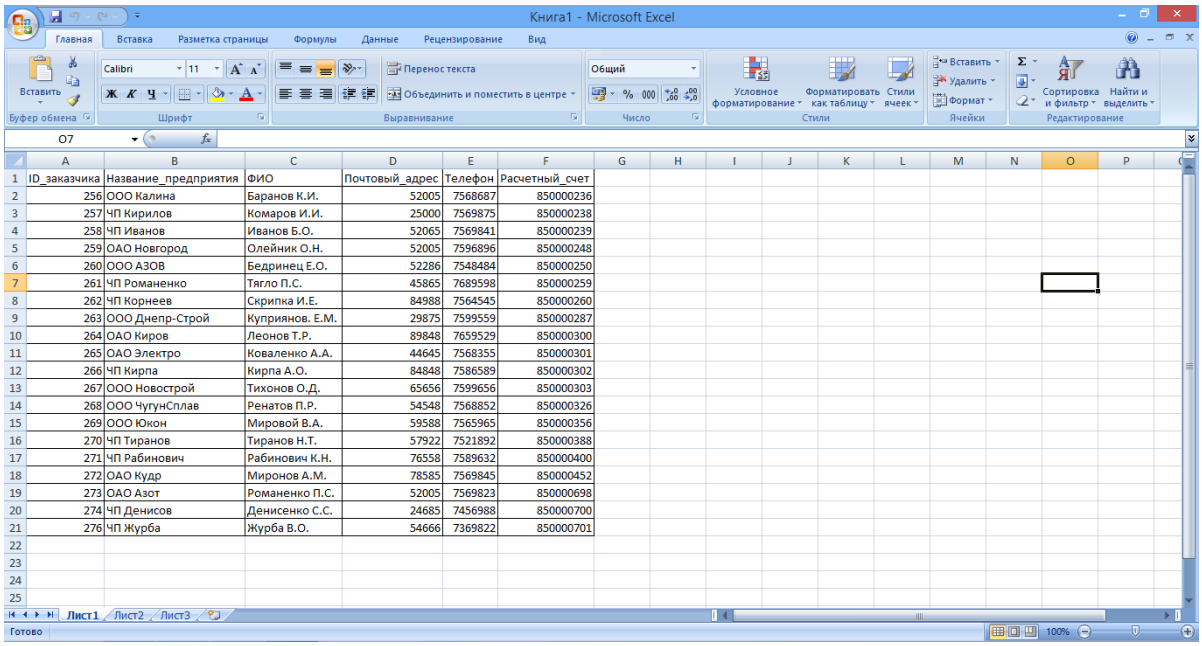
Рисунок 4.6 – Фрагмент трудового договіру

Приклад останніх форм наведений у Додатку В.

Запуск системи здійснюється по команді: Run|Run. Запуск системи можна здійснити і за допомогою гарячої клавіші F9.

В разі зависання системи або за відсутності виходу з системи переривання роботи здійснюється по команді Run|Program Reset. Після переривання роботи системи можна продовжувати її розробку або відладку.

Також є звіт у форматі .xls, тобто всі дані які занесені в таблиці виводяться за допомоги кнопки у вигляді зображеному на рисунку 4.7



ID_заказчика	Название_предприятия	ФИО	Почтовый_адрес	Телефон	Расчетный_счет
256	ООО Калина	Баранов К.И.	52005	7568687	850000236
257	ЧП Кирилов	Комаров И.И.	25000	7569875	850000238
258	ЧП Иванов	Иванов Б.О.	52065	7569841	850000239
259	ОАО Новгород	Олейник О.Н.	52005	7596896	850000248
260	ООО АЗОВ	Бедринец Е.О.	52286	7548484	850000250
261	ЧП Романенко	Тягло П.С.	45865	7689598	850000259
262	ЧП Корнеев	Скрипка И.Е.	84988	7564545	850000260
263	ООО Днепр-Строй	Куприянов. Е.М.	29875	7599559	850000287
264	ОАО Киров	Леонов Т.Р.	89848	7659529	850000300
265	ОАО Электро	Коваленко А.А.	44645	7568355	850000301
266	ЧП Кирпа	Кирпа А.О.	84848	7586589	850000302
267	ООО Новострой	Тихонов О.Д.	65656	7599656	850000303
268	ООО ЧугунСплав	Ренатов П.Р.	54548	7568852	850000326
269	ООО Юкон	Мировой В.А.	59588	7565965	850000356
270	ЧП Тиранов	Тиранов Н.Т.	57922	7521892	850000388
271	ЧП Рабинович	Рабинович К.Н.	76558	7589632	850000400
272	ОАО Кудр	Миронов А.М.	78585	7569845	850000452
273	ОАО Азот	Романенко П.С.	52005	7569823	850000698
274	ЧП Денисов	Денисенко С.С.	24685	7456988	850000700
276	ЧП Журба	Журба В.О.	54666	7369822	850000701

Рисунок 4.7 – Виведення звіту у вигляді таблиці

ВИСНОВКИ

У першій частині створено модель діяльності підприємства, що визначає його цілі і стратегії, бізнес-процеси, організаційну структуру. Якщо не зроблено коректний опис бізнес-процесів, не слід переходити до наступних стадій аналізу діяльності підприємства і тим більше до його автоматизації.

Розробку інформаційної системи підприємства було реалізовано засобами моделювання AllFusion Process і Data Modeler. Використання його можливостей, дозволяє співвідносити корпоративні ініціативи та завдання з бізнес-вимогами і процесами інформаційної архітектури та проектування додатків. Також було згенеровано SQL-скрипт, зроблено перевірку програмного коду в аналізаторі кодів сервера, створено на сервері схему даних проектованої системи і розроблено операторі для виконання операцій з таблицями даних.

Детальний аналіз характеристик процесів, що моделювалися, дозволяють обґрунтувати необхідні бізнес-процеси, що дає можливість оптимізувати діяльність підприємства і перевірити її на відповідність стандартам ISO, спроектувати оргструктуру, знизити витрати, виключити непотрібні операції і підвищити ефективність.

Дієздатність спроектованої бази даних показана та реалізована за допомогою програмного додатку Delphi.

СПИСОК ПОСИЛАНЬ

1. Маклаков С.В. BPwin и Erwin. CASE-средства для разработки информационных систем 2000. 202.

2. Маклаков, С.В. BPwin и ERwin. CASE-средства разработки информационных систем / С.В. Маклаков. – М. : ДИАЛОГ-МИФИ, 2001. – 256с.

3. Дубейковский, В. И. Практика функционального моделирования с AllFusion Data Modeler 4.1. (ERwin) Где? Зачем? Как? / В.И. Дубейков-ский. – М. : ДИАЛОГ-МИФИ, 2004. – 464 с.

4. Жилинский.А Самоучитель Microsoft SQL Server 2008. — СПб.: БХВ-Петербург, 2009. — 240 с.: ил. ISBN 978-5-9775-0217-7

5. Нильсен, Пол. Microsoft SQL Server 2005. Библия пользователя. : Пер. с англ.. – М. : ООО «И.Д. Вильямс», 2008. – 1232 с.

6. Фаронов В.В. Delphi 2005. Разработка приложений для баз данных и Интернета. – Спб.: Питер, 2006. – 603с.: ил.

7. http://abdulowa-alfia.narod.ru/olderfiles/2/Programmirovanie_na_yazyke_Delphi.pdf

8. Архангельський А.Я. Програмування в Delphi. – М.: ТОВ «>Бином-Пресс», 2004. – 1152 з.: мул.

9. refdb.ru/look/1110453.html

10. Гофман В.Е.,Хомоненко А.Д. Delphi. Швидкий старт. – СПб:БХВ-Петербург, 2003. – 288 з.: мул.

11. ДНАОП 0.00-1.31-99 “Правила охорони праці під час експлуатації електронно обчислювальних машин”.

12. ДБН.В.2.5-28-2006 “ Природне і штучне освітлення”, ДСан ПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин».

13. ДСН 3.3.6.042-99 “Санітарні норми мікроклімату виробничих приміщень”.

14. ДНАОП 0.00-1.31-99 «Затвердження правил охорони праці під час експлуатації електронно-обчислювальних машин».

15. Електронний ресурс: http://www.wikipedia.org/

16. Сорока К.О. Основи теорії систем і системного аналізу –127 с https://core.ac.uk/download/pdf/11320265.pdf:

17. Дата-центр або центр (зберігання і) обробки даних (ЦОД / ЦХОД) [Електронний ресурс] - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Дата-центр

18. Сервер (апаратне забезпечення) [Електронний ресурс] - Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Сервер_(аппаратное_обеспечение)#Надёжность

ДОДАТОК А

распечатать отдельно

ДОДАТОК Б

В І Д Г У К

на кваліфікаційну роботу бакалавра на тему:

«Розробка інформаційної системи підприємства "ІСТА»

студента групи 126-17-1 Проценко Кирило Віталійовича

Метою даної кваліфікаційної роботи є створення інформаційної системи підприємства «ІСТА».

Дана тема актуальна тому, що в умовах карантину всі країни світу отримали дуже багато втрат і тому оптимізація інформаційної системи підприємства є найкращим рішенням.

Така тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності фахівця галузі знань 12 «Інформаційні технології» спеціальності 126 «Інформаційні системи та технології» – розробка та експлуатація різнопланових інформаційних систем і пов'язаних з ними програмних засобів.

Завдання кваліфікаційної роботи віднесені в освітньо-професійній програмі підготовки випускника відповідної спеціальності до класу евристичних, рішення яких заснована на знаково-розумових уміннях фахівця.

Практичне значення результатів кваліфікаційної роботи полягає в можливості його використання для підвищення рівня інформаційної системи «ІСТА»

Оформлення графічних та текстових матеріалів пояснювальної записки кваліфікаційної роботи виконано на досить високому рівні і без відхилень від стандарту.

Ступінь самостійності виконання кваліфікаційної роботи повною мірою відповідає першому освітньо-кваліфікаційному рівню вищої освіти, тобто ступеню бакалавра.

В роботі досить повно виконано огляд сучасного стану ринку програмних додатків в сфері створення сайтів відповідного напряму.

Деякі дискусійні положення та несуттєві недоліки пов’язані з не досить чітким викладенням та описом структури роботи та дещо невисокою якістю ілюстративних матеріалів.

Незважаючи на вищевказані зауваження, кваліфікаційна робота в цілому заслуговує оцінки «задовільно», а її виконавець, студент Проценко К.В., присвоєння йому відповідної кваліфікації.

Керівник кваліфікаційної роботи, професор кафедри ІТКІ **В.В. Гнатушенко**

ДОДАТОК В

Рецензія

на кваліфікаційну роботу бакалавра

студента групи _____

спеціальності _____

Метою роботи є розробка інформаційної системи для підприємства «ІСТА». Завдання і зміст відповідають основній меті - перевірці знань і оцінці ступеня сформованості компетенцій студента в рамках його підготовки за галузі знань 12 «Інформаційні технології» спеціальності 126 «Інформаційні системи та технології».

Зміст пояснювальної записки відповідає затвердженій темі.

Актуальність обраної теми обумовлена тим, що створення інформаційної системи для підприємств на даний момент одне з найважливіших моментів роботи підприємств.

Кінцевий результат роботи полягає в проектуванні і розробці бази даних для підприємства та створенню програмного додатку для кінцевого користувача.

Оформлення пояснювальної записки виконано не в повній відповідності з діючими стандартами і нормативними вимогами, є зауваження.

Практичне значення отриманих результатів полягає в розробці відповідних моделей і програмній реалізації засобів зберігання і маніпулювання цифровими даними користувача на віддалених ресурсах.

До числа зауважень роботи слід віднести деяку непослідовність в описі стадій розробки проекту. Однак, зазначені зауваження не роблять істотного впливу на підсумкові результати.

Вважаю, що кваліфікаційна робота в цілому заслуговує оцінки «задовільно», а студент Проценко К.В. присвоєння йому кваліфікації «фахівець в області інформаційних технологій »за напрямом 126 «інформаційні системи та технологіїї»

Рецензент, доц. каф. САУ
НТУ «ДП» **Т.В. Хом’як**

ДОДАТОК Г

SQL- скрипт

```
CREATE TABLE Бригады
(
	ID_бригады		integer  NOT NULL ,
	Количество_рабочих	integer  NULL ,
	ID_отчет_бригады	integer  NOT NULL
)
go
CREATE TABLE Денежные_отчеты
(
	ID_денеж_отчета		integer  NOT NULL ,
	Дата			datetime  NULL ,
	Приход			integer  NULL ,
	Расход			integer  NULL
)
go
CREATE TABLE Договор
(
	ID_договора		integer  NOT NULL ,
	Дата_договора		datetime  NULL ,
	Общая_стоимость		integer  NULL ,
	ID_заявки		integer  NOT NULL
)
go
CREATE TABLE Заказчики
(
	ID_заказчика		integer  NOT NULL ,
	Название_предприятия varchar(20)  NULL ,
	Почтовый_адрес		integer  NULL ,
	Телефон			integer  NULL ,
	Расчетный_счет		integer  NULL ,
	ID_платежа		integer  NOT NULL
)
```

```
go
CREATE TABLE Зарплата
(
        ID_зарплаты         integer  NOT NULL ,
        Оклад               integer  NULL ,
        Премия              integer  NULL ,
        Штраф               integer  NULL ,
        Дата_начисления_ЗП   datetime  NULL
)
go
CREATE TABLE Заявка
(
        ID_заявки           integer  NOT NULL ,
        Дата_заявки         datetime  NULL ,
        Номер_договора      integer  NULL ,
        ID_продукции        integer  NOT NULL ,
        ID_заказчика        integer  NOT NULL
)
go
CREATE TABLE Оборудование
(
        ID_оборудования     integer  NOT NULL ,
        Наименование        varchar(20)  NULL ,
        Количество          integer  NULL ,
        Дата_установки      datetime  NULL ,
        Дата_посл_обслуж    datetime  NULL ,
        Дата_след_обслуж    datetime  NULL
)
go
CREATE TABLE Отгруж_продукция
(
        ID_отгрузки         integer  NOT NULL ,
        Дата                datetime  NULL ,
        Наименование        varchar(20)  NULL ,
```

```
        Количество          varchar(20)  NULL
)
go
CREATE TABLE Отдел_кадров
(
        ID_работника        integer  NOT NULL ,
        Фамилия             varchar(20)  NULL ,
        Имя                 varchar(20)  NULL ,
        Отчество            varchar(20)  NULL ,
        Номер__серия_паспорта varchar(20)  NULL ,
        Дата_рождения       datetime  NULL ,
        Адрес_прописки      varchar(20)  NULL ,
        Телефон             varchar(20)  NULL ,
        ID_труд_договора    integer  NOT NULL
)
go
CREATE TABLE Отчет_бригад
(
        ID_отчет_бригады    integer  NOT NULL ,
        Кол_во_изготовленного integer  NULL ,
        Кол_во_брака        integer  NULL
)
go
CREATE TABLE Платежи
(
        ID_платежа          integer  NOT NULL ,
        Дата_платежа        datetime  NULL ,
        Номер_накладной     integer  NULL ,
        Срок_оплаты         datetime  NULL ,
        Долг                integer  NULL ,
        Вид_платежа         integer  NULL ,
        ID_отгрузки         integer  NOT NULL ,
        ID_денеж_отчета     integer  NOT NULL
)
```

```
go
CREATE TABLE Продукция
(
        ID_продукции        integer  NOT NULL ,
        Количество          integer  NULL ,
        Тип                 varchar(20)  NULL ,
        Вид_АКБ             varchar(20)  NULL ,
        ID_ресурсов         integer  NOT NULL ,
        ID_распоряжения     integer  NOT NULL
)
go
CREATE TABLE Распоряжения
(
        ID_распоряжения     integer  NOT NULL ,
        Дата_распоряжения   datetime  NULL ,
        Дата_выполнения     datetime  NULL ,
        ID_цеха             integer  NOT NULL
)
go
CREATE TABLE Ресурсы
(
        ID_ресурсов         integer  NOT NULL ,
        Наименование        varchar(20)  NULL ,
        Количество          integer  NULL ,
        Температура_хранения integer  NULL
)
go
CREATE TABLE Трудовой_договор
(
        ID_труд_договора    integer  NOT NULL ,
        Должность           varchar(20)  NULL ,
        ID_зарплаты         integer  NOT NULL
)
go
```

```
CREATE TABLE Цехи
(
	ID_цеха		integer  NOT NULL ,
	Начальник_цеха_ФИО   varchar(20)  NULL ,
	Количество_бригад    integer  NULL ,
	Начало_смены	datetime  NULL ,
	Конец_смены	datetime  NULL ,
	ID_бригады	integer  NOT NULL ,
	ID_оборудования	integer  NOT NULL ,
	ID_работника	integer  NOT NULL
)
go
ALTER TABLE Бригады
	ADD CONSTRAINT R_11 FOREIGN KEY (ID_отчет_бригады) REFERENCES
Отчет_бригад(ID_отчет_бригады)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go
ALTER TABLE Договор
	ADD CONSTRAINT R_3 FOREIGN KEY (ID_заявки) REFERENCES
Заявка(ID_заявки)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go
ALTER TABLE Заказчики
	ADD CONSTRAINT R_8 FOREIGN KEY (ID_платежа) REFERENCES
Платежи(ID_платежа)
		ON DELETE NO ACTION
		ON UPDATE NO ACTION
go
ALTER TABLE Заявка
	ADD CONSTRAINT R_2 FOREIGN KEY (ID_продукции) REFERENCES
Продукция(ID_продукции)
		ON DELETE NO ACTION
```

```
        ON UPDATE NO ACTION
go
ALTER TABLE Заявка
    ADD CONSTRAINT R_4 FOREIGN KEY (ID_заказчика) REFERENCES
Заказчики(ID_заказчика)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go
ALTER TABLE Отдел_кадров
    ADD CONSTRAINT R_9 FOREIGN KEY (ID_труд_договора) REFERENCES
Трудовой_договор(ID_труд_договора)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go
ALTER TABLE Платежи
    ADD CONSTRAINT R_15 FOREIGN KEY (ID_отгрузки) REFERENCES
Отгруж_продукция(ID_отгрузки)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go
ALTER TABLE Платежи
    ADD CONSTRAINT R_17 FOREIGN KEY (ID_денеж_отчета) REFERENCES
Денежные_отчеты(ID_денеж_отчета)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go
ALTER TABLE Продукция
    ADD CONSTRAINT R_6 FOREIGN KEY (ID_ресурсов) REFERENCES
Ресурсы(ID_ресурсов)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
go
ALTER TABLE Продукция
    ADD CONSTRAINT R_7 FOREIGN KEY (ID_распоряжения) REFERENCES
```

```
Распоряжения(ID_распоряжения)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
ALTER TABLE Распоряжения
        ADD CONSTRAINT R_14 FOREIGN KEY (ID_цеха) REFERENCES Цехи(ID_цеха)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
ALTER TABLE Трудовой_договор
        ADD CONSTRAINT R_10 FOREIGN KEY (ID_зарплаты) REFERENCES
Зарплата(ID_зарплаты)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
ALTER TABLE Цехи
        ADD CONSTRAINT R_12 FOREIGN KEY (ID_бригады) REFERENCES
Бригады(ID_бригады)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
ALTER TABLE Цехи
        ADD CONSTRAINT R_13 FOREIGN KEY (ID_оборудования) REFERENCES
Оборудование(ID_оборудования)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
ALTER TABLE Цехи
        ADD CONSTRAINT R_16 FOREIGN KEY (ID_работника) REFERENCES
Отдел_кадров(ID_работника)
                ON DELETE NO ACTION
                ON UPDATE NO ACTION
go
```

ДОДАТОК Д

Код Delphi

```
unit FormMain;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, Menus, ShellApi, ExtCtrls, jpeg;
type
  TMainForm = class(TForm)
    MainMenu1: TMainMenu;
    N1: TMenuItem;
    N2: TMenuItem;
    N3: TMenuItem;
    N4: TMenuItem;
    N5: TMenuItem;
    N6: TMenuItem;
    N7: TMenuItem;
    N8: TMenuItem;
    N9: TMenuItem;
    N10: TMenuItem;
    Button1: TButton;
    Button2: TButton;
    ImageFon: TImage;
    Button13: TButton;
    Button14: TButton;
    Button15: TButton;
    Button16: TButton;
    Button17: TButton;
    Button18: TButton;
    Button19: TButton;
```

```
    Button20: TButton;
    Button21: TButton;
    Button22: TButton;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure N6Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure N9Click(Sender: TObject);
    procedure N7Click(Sender: TObject);
    procedure N2Click(Sender: TObject);
    procedure N10Click(Sender: TObject);
    procedure N4Click(Sender: TObject);
    procedure Button13Click(Sender: TObject);
    procedure Button14Click(Sender: TObject);
    procedure Button15Click(Sender: TObject);
    procedure Button16Click(Sender: TObject);
    procedure Button17Click(Sender: TObject);
    procedure Button18Click(Sender: TObject);
    procedure Button19Click(Sender: TObject);
    procedure Button20Click(Sender: TObject);
    procedure Button21Click(Sender: TObject);
    procedure Button22Click(Sender: TObject);
    private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  MainForm: TMainForm;
```

```
implementation
uses FormStart, FormHistory, FormContact, FormZakazchiki, FormZayavka,
FormProduct, FormRaspor,
FormDogovor, FormResurs, FormPlateg, FormCex, FormOboryd,
FormReport_Brugad, FormOtdel_kadrov,
FormOtgryzka ;
{$R *.dfm}
procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  StartForm.Close;
end;
procedure TMainForm.N6Click(Sender: TObject);
begin
   ShellExecute(Handle, 'open', 'D:\Ñåðâåé\ista ñàéò\Èñòîðèÿ.htm', nil, nil,
SW_NORMAL);
end;
procedure TMainForm.N7Click(Sender: TObject);
begin
  ShellExecute(Handle, 'open', 'D:\Ñåðâåé\ista ñàéò\Ñòðóêòóðà.htm', nil, nil,
SW_NORMAL);
end;
procedure TMainForm.N2Click(Sender: TObject);
begin
  ShellExecute(Handle, 'open', 'D:\Ñåðâåé\ista ñàéò\Ïðîäóêöèÿ.html', nil, nil,
SW_NORMAL);
end;
procedure TMainForm.Button2Click(Sender: TObject);
begin
  If ZakazchikiForm = nil
    then ZakazchikiForm := TZakazchikiForm.Create(Self);
```

```
  ZakazchikiForm.ShowModal;
end;
procedure TMainForm.Button1Click(Sender: TObject);
begin
  If ZayavkaForm = nil
    then ZayavkaForm := TZayavkaForm.Create(Self);
  ZayavkaForm.ShowModal;
end;
  procedure TMainForm.N9Click(Sender: TObject);
begin
  ContactDlg := TContactDlg.Create(Self);
  ContactDlg.ShowModal;
end;
procedure TMainForm.N10Click(Sender: TObject);
begin
  ShellExecute(Handle, 'open', 'D:\Ñåðâåé\ista ñàéò\Ñòàíäàðòû.html', nil, nil,
SW_NORMAL);
end;
procedure TMainForm.N4Click(Sender: TObject);
begin
  ShellExecute(Handle, 'open', 'D:\Ñåðâåé\ista ñàéò\ÎÎÎ_Óêðñïëàâ_.html', nil, nil,
SW_NORMAL);
end;
procedure TMainForm.Button13Click(Sender: TObject);
begin
  If ProductForm = nil
    then ProductForm := TProductForm.Create(Self);
  ProductForm.ShowModal;
end;
procedure TMainForm.Button14Click(Sender: TObject);
```

```
begin
  If RasporForm = nil
    then RasporForm := TRasporForm.Create(Self);
  RasporForm.ShowModal;
end;
procedure TMainForm.Button15Click(Sender: TObject);
begin
  If DogovorForm = nil
    then DogovorForm := TDogovorForm.Create(Self);
  DogovorForm.ShowModal;
end;
procedure TMainForm.Button16Click(Sender: TObject);
begin
  If ResursForm = nil
    then ResursForm := TResursForm.Create(Self);
  ResursForm.ShowModal;
end;
procedure TMainForm.Button17Click(Sender: TObject);
begin
  If PlategForm = nil
    then PlategForm := TPlategForm.Create(Self);
  PlategForm.ShowModal
end;
procedure TMainForm.Button18Click(Sender: TObject);
begin
  If CexForm = nil
    then CexForm := TCexForm.Create(Self);
  CexForm.ShowModal
end;
procedure TMainForm.Button19Click(Sender: TObject);
```

```
begin
  If OborydForm = nil
    then OborydForm := TOborydForm.Create(Self);
  OborydForm.ShowModal
end;
procedure TMainForm.Button20Click(Sender: TObject);
begin
  If Report_BrugadForm = nil
    then Report_BrugadForm := Report_BrugadForm.Create(Self);
  Report_Brugad.ShowModal
end;
procedure TMainForm.Button21Click(Sender: TObject);
begin
  If Otdel_kadrovForm = nil
    then Otdel_kadrovForm := Report_BrygadForm.Create(Self);
  Otdel_kadrov.ShowModal
end;
procedure TMainForm.Button22Click(Sender: TObject);
begin
  If OtgryzkaForm = nil
    then OtgryzkaForm := OtgryzkaForm.Create(Self);
  Otgryzka.ShowModal
end;
end.

unit FormStart;
interface
uses
  Windows, ExtCtrls, DB, ADODB, Classes, ActnList, StdCtrls, ComCtrls,
ShellApi,
```

```
  Buttons, Controls, Forms, Registry, SysUtils, ImgList, Dialogs, jpeg;
type
  TStartForm = class(TForm)
    OkBitBtn: TBitBtn;
    CancelBitBtn: TBitBtn;
    ActionList: TActionList;
    Bevel: TBevel;
    OkAction: TAction;
    CancelAction: TAction;
    StateLabel: TLabel;
    SavePasswordCheckBox: TCheckBox;
    PasswordEdit: TEdit;
    PasswordLabel: TLabel;
    UserLabel: TLabel;
    UserNameEdit: TEdit;
    Image1: TImage;
    procedure OkActionExecute(Sender: TObject);
    procedure FormKeyPress(Sender: TObject; var Key: Char);
    procedure CancelActionExecute(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure OkActionUpdate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
  private
    RegKey: string;
    Trial: integer;
    RegEnter: TRegistry;
    procedure GetLogin;
    procedure SaveLogin;
    procedure ClearLogin;
    function EncryptDecrypt(sInput:string; sKey:string): string;
```

```
    procedure RunProgram;
  public
    end;
var
  StartForm: TStartForm;
implementation
uses DataModule, FormMain;
{$R *.dfm}
const
  SUser = 'L';  //Login
  SPassword = 'P';   //Password
  SSavePassword = 'SP';  // SavePassword
  SDataBaseName = 'B';
  SServerName = 'S';
  SRootKey = HKEY_CURRENT_USER;
  SError = 'Îøèáêà';
  ICountTrialEnter = 3;  //êîëè÷åñòâî ïîïûòîê ââîäà ïàðîëÿ
function TStartForm.EncryptDecrypt(sInput:string; sKey:string): string;
var
  longkey: string;
  i: integer;
  toto: char;
begin
  for i := 0 to (length(sInput) div length(sKey)) do
    longkey := longkey + sKey;
  for i := 1 to length(sInput) do
    begin
      toto := chr((ord(sInput[i]) xor ord(longkey[i]))); // XOR àëãîðèòì
      Result := Result + toto;
    end;
```

```
end;
procedure TStartForm.FormCreate(Sender: TObject);
begin
  Trial := 0;
  RegKey := ExtractFileName(Application.ExeName);
  Delete(RegKey, Pos(ExtractFileExt(Application.ExeName), RegKey),
Length(ExtractFileExt(Application.ExeName)));
  RegKey := '\Software\' + RegKey;
  GetLogin;
end;
procedure TStartForm.FormActivate(Sender: TObject);
begin
  if (UserNameEdit.Text <> '')and(PasswordEdit.Text <> '') and
SavePasswordCheckBox.Checked and OkBitBtn.CanFocus
    then OkBitBtn.SetFocus;
end;
procedure TStartForm.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if key  =#13 then
    begin
      FindNextControl(ActiveControl, true, true, false).SetFocus;
      key := #0;
    end;
end;
procedure TStartForm.OkActionExecute(Sender: TObject);
begin
  inc(Trial);
  OkAction.Enabled := False;
  Application.ProcessMessages;
  if SQLDataModule.EnterSQL(UserNameEdit.Text, PasswordEdit.Text)
```

```
    then
      begin
        SaveLogin;
        if not Application.Terminated
          then RunProgram;
      end
    else
      begin
        UserNameEdit.SetFocus;
        ClearLogin;
        ShowMessage('gfhjkm');
        if Trial >= ICountTrialEnter
          then Close;
      end;
end;
procedure TStartForm.CancelActionExecute(Sender: TObject);
begin
  ClearLogin;
  Application.Terminate;
end;

procedure TStartForm.ClearLogin;
begin
  RegEnter := TRegistry.Create;
  try
    RegEnter.RootKey := SRootKey;
    if RegEnter.OpenKey(RegKey, True) then
      begin
        RegEnter.WriteString(SUser, '');
        RegEnter.WriteBool(SSavePassword, False);
```

```
      RegEnter.WriteString(SPassword, '');
      RegEnter.CloseKey;
    end
  finally
    RegEnter.Free;
  end;
end;
procedure TStartForm.GetLogin;
begin
  RegEnter := TRegistry.Create;
  try
    RegEnter.RootKey := SRootKey;
    if RegEnter.OpenKeyReadOnly(RegKey) then
      begin
        UserNameEdit.Text := EncryptDecrypt(RegEnter.ReadString(SUser), SUser);
        SavePasswordCheckBox.Checked := RegEnter.ReadBool(SSavePassword);
        if SavePasswordCheckBox.Checked
          then PasswordEdit.Text :=
EncryptDecrypt(RegEnter.ReadString(SPassword), SPassword)
          else PasswordEdit.Text := '';

        RegEnter.CloseKey;
      end
  finally
    RegEnter.Free;
  end;
end;
procedure TStartForm.SaveLogin;
begin
  RegEnter := TRegistry.Create;
```

```
  try
    RegEnter.RootKey := SRootKey;
    if RegEnter.OpenKey(RegKey, True) then
      begin
        RegEnter.WriteString(SUser, EncryptDecrypt(UserNameEdit.Text, SUser));
        RegEnter.WriteBool(SSavePassword, SavePasswordCheckBox.Checked);
        if SavePasswordCheckBox.Checked
          then RegEnter.WriteString(SPassword, EncryptDecrypt(PasswordEdit.Text,
SPassword))
          else RegEnter.WriteString(SPassword, '');

        RegEnter.CloseKey;
      end
  finally
    RegEnter.Free;
  end;
end;
procedure TStartForm.OkActionUpdate(Sender: TObject);
begin
  OkAction.Enabled := (Trim(UserNameEdit.Text) <>
'')and(Trim(PasswordEdit.Text) <> '')and(StateLabel.Tag = 0);
end;
procedure TStartForm.RunProgram;
begin
  try
    MainForm := TMainForm.Create(nil);
    MainForm.Show;
    StartForm.Hide;
  except
    on E: Exception do Application.MessageBox(PChar(E.Message), SError,
```

```
MB_OK + MB_ICONHAND);
  end;
end;
end.
unit FormContact;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, ShellApi;
type
  TContactDlg = class(TForm)
    OKBtn: TButton;
    Bevel1: TBevel;
    Label1: TLabel;
    procedure Label1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Label1MouseEnter(Sender: TObject);
    procedure Label1MouseLeave(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  ContactDlg: TContactDlg;
implementation
{$R *.dfm}
procedure TContactDlg.Label1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  ShellExecute(Handle, 'open', 'http://ista.com.ua', nil, nil, SW_NORMAL);
```

```
end;
procedure TContactDlg.Label1MouseEnter(Sender: TObject);
begin
  Label1.Font.Style := Label1.Font.Style + [fsUnderline];
  Label1.Font.Color := clBlue;
  Label1.Cursor :=crHandPoint;
end;
procedure TContactDlg.Label1MouseLeave(Sender: TObject);
begin
  Label1.Font.Style := Label1.Font.Style - [fsUnderline];
  Label1.Font.Color := clDefault;
  Label1.Cursor :=crDefault;
end;
end.


unit FormHistory;
interface
uses Windows, SysUtils, Classes, Graphics, Forms, Controls, StdCtrls,
  Buttons, ExtCtrls, ShellApi;
type
  THistoryDlg = class(TForm)
    OKBtn: TButton;
    Bevel1: TBevel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
```

```
  HistoryDlg: THistoryDlg;
implementation
{$R *.dfm}
end.

unit FormZakazchiki;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, ComCtrls, StdCtrls, Mask, DBCtrls, DB,
  ExtCtrls, ADODB, Buttons;
type
  TZakazchikiForm = class(TForm)
    DBGrid1: TDBGrid;
    zakazchikiADOQuery1: TADOQuery;
    zakazchikiDataSource1: TDataSource;
    Panel1: TPanel;
    DBNavigator1: TDBNavigator;
    GroupBox1: TGroupBox;
    Label10: TLabel;
    NameFindEdit: TEdit;
    Label11: TLabel;
    DogFindEdit: TEdit;
    FindSpeedButton: TSpeedButton;
    DelFindBitBtn: TBitBtn;
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    Label3: TLabel;
```

```
    DBEdit3: TDBEdit;
    Label4: TLabel;
    DBEdit4: TDBEdit;
    Label5: TLabel;
    DBEdit5: TDBEdit;
    Label6: TLabel;
    DBEdit6: TDBEdit;
    zakazchikiADOQuery1ID_: TIntegerField;
    zakazchikiADOQuery1_: TStringField;
    zakazchikiADOQuery1DSDesigner: TStringField;
    zakazchikiADOQuery1_2: TIntegerField;
    zakazchikiADOQuery1DSDesigner2: TIntegerField;
    zakazchikiADOQuery1_3: TIntegerField;
    Button1: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FindSpeedButtonClick(Sender: TObject);
    procedure DelFindBitBtnClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  ZakazchikiForm: TZakazchikiForm;
implementation
uses DataModule, Report;
{$R *.dfm}
procedure TZakazchikiForm.FormCreate(Sender: TObject);
```

```
begin
  zakazchikiADOQuery1.Open;
end;
procedure TZakazchikiForm.FindSpeedButtonClick(Sender: TObject);
begin
  zakazchikiADOQuery1.Filter := '';
  if DogFindEdit.Text <> ''
    then  zakazchikiADOQuery1.Filter := 'ID_çàêàç÷èêà = ' + DogFindEdit.Text;
  if NameFindEdit .Text <> ''
    then
      if DogFindEdit.Text <> ''
        then zakazchikiADOQuery1.Filter := zakazchikiADOQuery1.Filter + ' AND '
+ 'ÔÈÎ = ' + DogFindEdit.Text
        else zakazchikiADOQuery1.Filter := 'ÔÈÎ like  %' + NameFindEdit.Text +
'%';
  zakazchikiADOQuery1.Filtered := true;
end;
procedure TZakazchikiForm.DelFindBitBtnClick(Sender: TObject);
begin
  zakazchikiADOQuery1.Filter := '';
  zakazchikiADOQuery1.Filtered := False;
  DogFindEdit.Text := '';
  NameFindEdit.Text := '';
end;
procedure TZakazchikiForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  DelFindBitBtnClick(self);
end;
procedure TZakazchikiForm.Button1Click(Sender: TObject);
```

```
begin
  ExportDBGridToExcel(DBGrid1);
end;
end.

unit FormZayavka;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, ADODB, DBCtrls, Grids, DBGrids, ExtCtrls, StdCtrls, Mask,
  Buttons;

type
  TZayavkaForm = class(TForm)
    Panel1: TPanel;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    zayavkaDataSource1: TDataSource;
    Button1: TButton;
    Label1: TLabel;
    DBEdit1: TDBEdit;
    Label2: TLabel;
    DBEdit2: TDBEdit;
    Label3: TLabel;
    DBEdit3: TDBEdit;
    Label4: TLabel;
    DBEdit4: TDBEdit;
    Label5: TLabel;
    DBEdit5: TDBEdit;
    GroupBox1: TGroupBox;
```

```
    Label11: TLabel;
    FindSpeedButton: TSpeedButton;
    DogFindEdit: TEdit;
    zayavkaADOQuery1: TADOQuery;
    DelFindBitBtn: TBitBtn;
    procedure DogWordButtonClick(Sender: TObject);
    procedure FindSpeedButtonClick(Sender: TObject);
    procedure DelFindBitBtnClick(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  ZayavkaForm: TZayavkaForm;
implementation
uses DataModule, Report;
{$R *.dfm}
procedure TZayavkaForm.FindSpeedButtonClick(Sender: TObject);
begin
  zayavkaADOQuery1.Filter := '';
  if DogFindEdit.Text <> ''
    then  zayavkaADOQuery1.Filter := 'Íîìåð_äîãîâîðà = ' + DogFindEdit.Text;
   zayavkaADOQuery1.Filtered := true;
end;
procedure TZayavkaForm.DelFindBitBtnClick(Sender: TObject);
begin
```

```
  zayavkaADOQuery1.Filter := '';
  zayavkaADOQuery1.Filtered := False;
  DogFindEdit.Text := '';
end;
procedure TZayavkaForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  DelFindBitBtnClick(self);
end;

procedure TZayavkaForm.DogWordButtonClick(Sender: TObject);
begin
  PrintDogovorGos(zayavkaADOQuery1.FieldByName('id_çàÿâêè').AsInteger);
end;
procedure TZayavkaForm.Button1Click(Sender: TObject);
begin
  ExportDBGridToExcel(DBGrid1);
end;
procedure TZayavkaForm.FormCreate(Sender: TObject);
begin
  zayavkaADOQuery1.Active := True;
end;
end.

unit DataModule;
interface
uses
  Forms, SysUtils, Classes, Windows, Controls, ADODB, DB, DBGrids, Variants;
type
  TSQLDataModule = class(TDataModule)
```

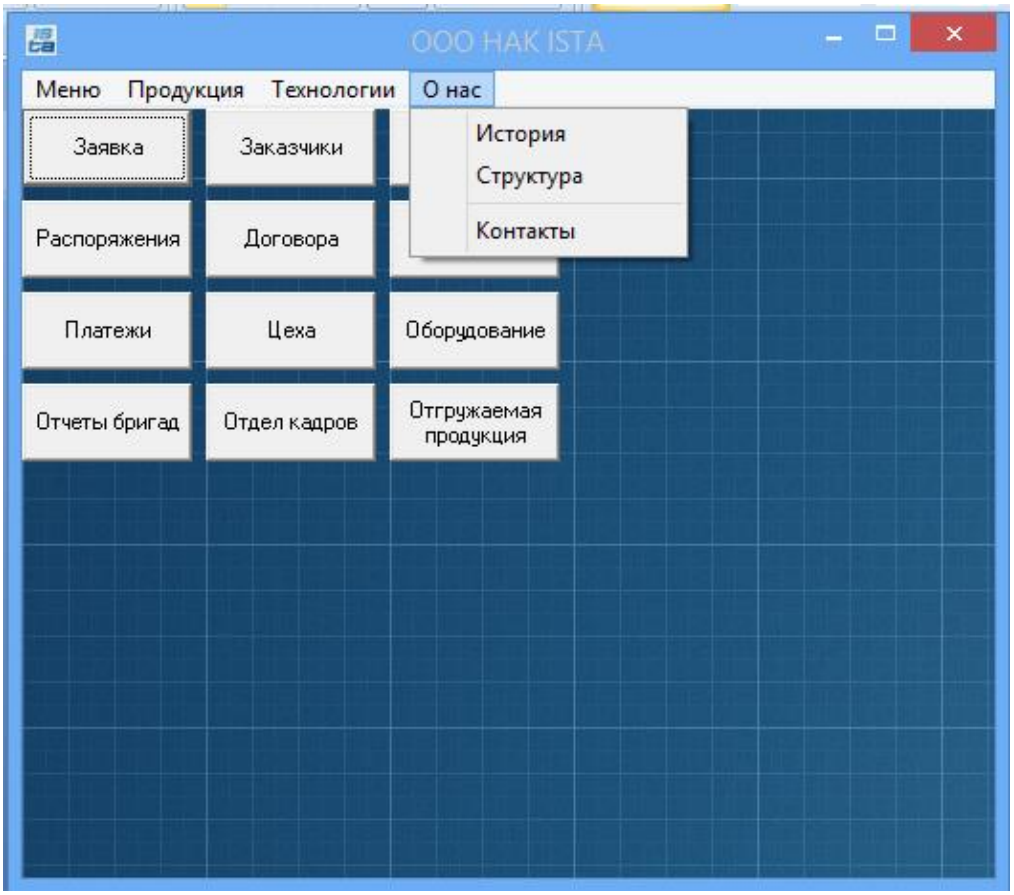
```
    SQL_ADOConnection: TADOConnection;
    SelectQuery: TADOQuery;
    InsertQuery: TADOQuery;
    UpdateQuery: TADOQuery;
    ADOCommand: TADOCommand;
    ADOStoredProc: TADOStoredProc;
    procedure DataModuleCreate(Sender: TObject);
    procedure DataModuleDestroy(Sender: TObject);
  private
    DataBaseName: string;
  public
    function EnterSQL(const Login, Password: string): boolean;
  end;
var
  SQLDataModule: TSQLDataModule;
implementation
uses  IniFiles, Graphics, Dialogs, Math, FMTBcd;
{$R *.dfm}
const
  ServerConnectionString =
  'Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;User ID=sa;Initial Catalog=delphinew;Data Source=HOME;';
{ TSQLDataModule }
procedure TSQLDataModule.DataModuleCreate(Sender: TObject);
begin
  SQL_ADOConnection.Connected := False;
  SQL_ADOConnection.ConnectionString := '';
 end;
procedure TSQLDataModule.DataModuleDestroy(Sender: TObject);
begin
```

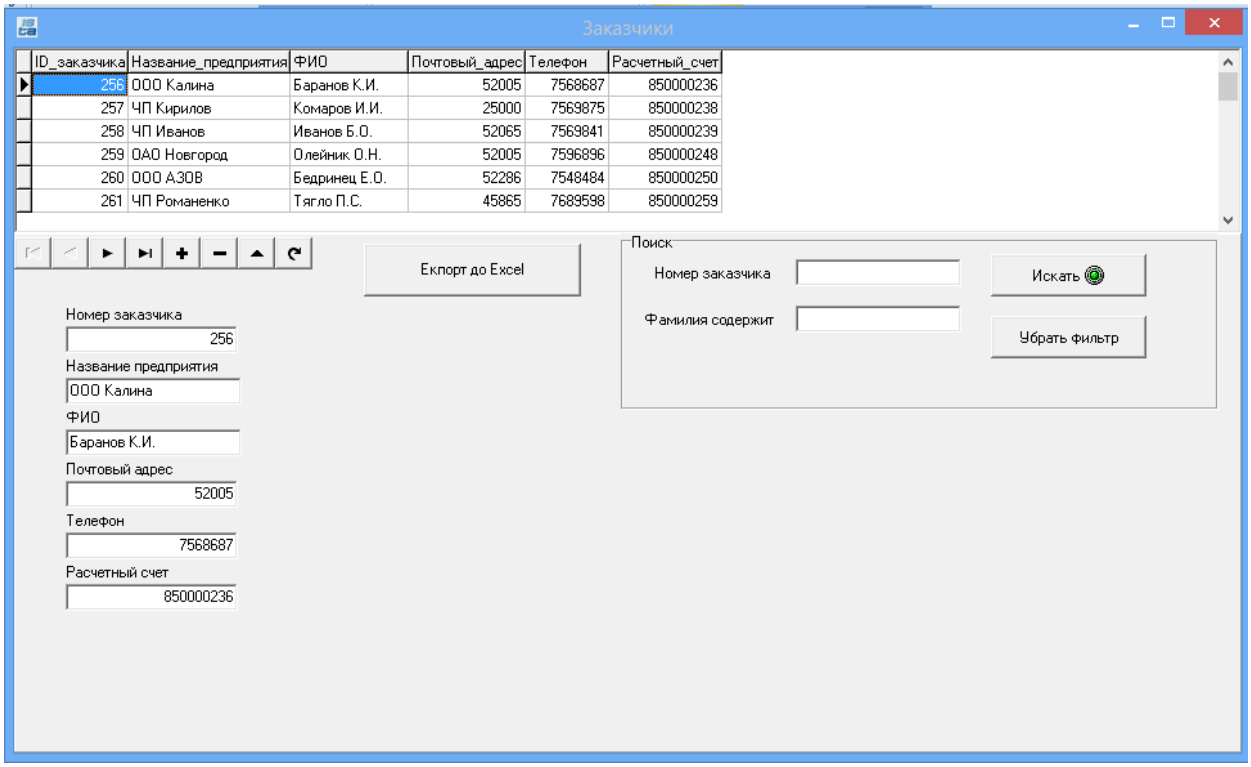
```
  try
    if SQL_ADOConnection.InTransaction
      then SQL_ADOConnection.RollbackTrans;
    SQL_ADOConnection.Connected := False;
  except
  end;
end;
function TSQLDataModule.EnterSQL(const Login, Password: string): boolean;
var
  OldCursor: TCursor;
begin
  OldCursor := Screen.Cursor;
  Screen.Cursor := crSQLWait;
  try
    if (Login = 'sa') and (Password = '12345')
      then
        begin
          SQL_ADOConnection.ConnectionString := ServerConnectionString;
          try
            SQL_ADOConnection.Connected := True;
          except
            on E: Exception do
              Application.MessageBox(PChar(E.Message), 'Помилка!', MB_OK +
MB_ICONHAND);
          end;
        end;
  finally
    Result := SQL_ADOConnection.Connected;
    Screen.Cursor := OldCursor;
  end;
```

```
end;
end.
```

ДОДАТОК Е

Форми Delphi





ДОДАТОК Є

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:
Автор: Кирило Проценко

Співавтор:

Назва: Проценко К.В 126-17-1.docx

Науковий керівник: Професор Володимир Гнатушенко

Підрозділ: Кафедра інформаційних технологій та комп'ютерної інженерії
Коефіцієнт подібності 1:5.5%

Коефіцієнт подібності 2:2.4%

Мікропробіли: 7

Заміна букв: 5

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2021-06-18 15:17:17.0

Після аналізу Звіту подібності констатую наступне:

☒ Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.

☐ Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.

☐ Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.

Обгрунтування:

2021-06-19 Професор Володимир Гнатушенко

Дата експерт