МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютеризованих систем управління

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

_____Литвиненко О.Є.
"_____"_____ 2021 р.

ДИПЛОМНИЙ ПРОЕКТ
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"БАКАЛАВР"

Тема: Програмний засіб перегляду метаданих бази даних комп'ютерних комплектуючих

Виконавець: _____Радченко Р.В.

Керівник: _____ к.т.н., доцент Халімон Н. Ф.

Нормоконтролер: _____Тупота Є.В.

Київ 2021

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра комп'ютеризованих систем управління
Спеціальність 123 «Комп'ютерна інженерія»
(шифр, найменування)

Освітньо-професійна програма «Системне програмування»
Форма навчання заочна

ЗАТВЕРЖУЮ
Завідувач кафедри

Литвиненко О. Є.

« ____ »_____2021 р.

З А В Д А Н Н Я
на виконання дипломної роботи (проекту)

Радченка Романа Валерійовича
(прізвище, ім'я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту): Програмний засіб перегляду метаданих бази даних комп'ютерних комплектуючих

затверджена наказом ректора від "21" грудня 2020 р. № 2523/ст.

2. Термін виконання роботи (проекту): з 11.01.2021 до 28.02.2021

3. Вихідні дані до роботи (проекту): база даних, *Microsoft SQL Server*, інтегроване середовище розробки *Visual Studio*, мова *C#*, мова *SQL*.

4. Зміст пояснювальної записки:

1) Засоби адміністрування баз даних.
2) Проектування програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих.
3) Розробка програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих.

5. Перелік обов'язкового графічного (ілюстративного) матеріалу:

1) Вікно утиліти адміністрування бази даних комп'ютерних комплектуючих.
2) Вікно конструктора схем наборів даних *_Dipl Computer komplektDataSet* для бази даних комп'ютерних комплектуючих.
3) Головне вікно програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих.
4) Вікно форми програмного засобу для пункту меню «Типы данных для таблицы».

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з постановкою задачі дипломного проектування	22.12.2020 – 21.01.2021	
2	Вивчити спеціальну літературу і технічну документацію	22.01.2021 – 23.01.2021	
3	Проаналізувати системи управління базами даних та утиліти для адміністрування	24.01.2021 – 25.01.2021	
4	Написати розділ 1.	26.01.2021 – 27.01.2021	
5	Дослідити засоби адміністрування баз даних	28.01.2021 – 01.02.2021	
6	Написати розділ 2.	02.02.2021 – 03.02.2021	
7	Провести проектування програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих	04.02.2021 – 05.02.2021	
8	Написати розділ 2.	06.02.2021 – 07.02.2021	
9	Провести розробку програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих	08.02.2021 – 15.02.2021	
10	Написати розділ 3.	16.02.2021 – 17.02.2021	
11	Оформити пояснювальну записку	18.02.2021 – 20.02.2021	
12	Підготувати графічний демонстраційний матеріал та доповідь	21.02.2021 – 22.02.2021	

7. Дата видачі завдання: “22” грудня 2020 р.

Керівник дипломної роботи (проекту) _____ Халімон Н.Ф.
(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Радченко Р.В.
(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмний засіб перегляду метаданих бази даних комп'ютерних комплектуючих»: 55 с., 9 рис., 20 літературних джерел.

АДМІНІСТРУВАННЯ БАЗ ДАНИХ, МЕТАДАНІ, ПАРАМЕТРИ СИСТЕМНИХ ПОДАНЬ, ІНФОРМАЦІЙНА СХЕМА.

Об’єкт проектування – адміністрування баз даних СУБД *MS SQL Server*.

Предмет проектування – програмний засіб перегляду метаданих бази даних комп'ютерних комплектуючих.

Метод проектування – застосування засобів адміністрування баз даних для розробки програмного засобу.

Дипломний проект присвячено тематиці моніторингу та налаштування параметрів баз даних.

Програма може бути використана в учбовому процесі НАУ.

ЗМІСТ

ВСТУП

Адміністрування баз даних − управління інформаційними ресурсами, включаючи планування бази даних, розробку і впровадження стандартів, визначення обмежень і процедур, а також концептуальне і логічне проектування баз даних. Адміністратор даних відповідає за корпоративні інформаційні ресурси. На практиці це часто пов'язано з управлінням даними, які є спільно використовуваним ресурсом для різних користувачів і прикладних програм організації. В одних випадках адміністрування даних може бути окремим функціональним завданням, а в інших − поєднуватися з адмініструванням бази даних.

Об’єкт проектування – адміністрування баз даних СУБД *MS SQL Server*.

Предмет проектування – програмний засіб перегляду метаданих бази даних комп'ютерних комплектуючих.

Метод проектування – застосування засобів адміністрування баз даних для розробки програмного засобу.

Для пошуку інформації про метадані в СУБД *MS SQL Server* можна використовувати кілька різних альтернативних способів. Інформаційна схема − стандартне рішення для доступу до метаданих, яке надає загальний інтерфейс не тільки для компонента *Database Engine*, але і для всіх існуючих реляційних систем баз даних. Метадані описують атрибути об'єктів в системі баз даних, тому тема дипломної роботи «Програмний засіб перегляду метаданих бази даних комп'ютерних комплектуючих» є актуальною.

В даний час при обмірковуванні стратегії планування інформаційної системи все більший акцент робиться на важливості адміністрування даних. Організації все більше схильні приділяти увагу значенню даних, що використовуються, або зібраних в їх інформаційній системі, як засобу досягнення більш високої конкурентоспроможності. В результаті виникає обов'язкова вимога злиття стратегії побудови інформаційних систем з бізнес-стратегіями організації. Це дозволяє створити організацію з більш гнучкою структурою, здатну

адаптуватися до різких змін, що має більш творче та інноваційне внутрішнє середовище та забезпечує ефективну перебудову бізнес-процесів в разі потреби. Згадане перенесення акцентів означає, що адміністратор у все більшій мірі повинен розуміти ідеологію розвитку не тільки інформаційних систем, але і бізнес-процесів, і грати ключову роль в розробці стратегії розвитку інформаційної системи, підтримуючи її відповідність діловим стратегіям організації.

При адмініструванні баз даних в СКБД зазвичай використовують системні бази даних, утиліти, системні подання, мову маніпулювання даними *SQL*. Для СКБД *MS SQL Server* основною утилітою адміністрування є *MS SQL Server Managment Studio.*

Подання (*View*) − це віртуальна таблиця, вміст якої визначається запитом. Як і таблиця, подання складається з ряду іменованих стовпців і рядків даних. Поки подання не буде проіндексовано, воно не існує в базі даних як збережена сукупність значень. Рядки і стовпці даних беруться з таблиць, зазначених у запиті, який визначає подання, та динамічно створюваних при зверненнях до подання.

В розробленому додатку для моніторингу параметрів бази даних комп'ютерних комплектуючих використано системні подання. *Microsoft SQL Server* надає наступні колекції системних подань, що містять метадані: подання каталогу, подання інформаційної схеми, подання сумісності, подання реплікації, динамічні адміністративні подання.

Подання каталогу містять інформацію, що використовується ядром бази даних *SQL Server*. Вони надають найбільш загальний інтерфейс метаданих каталогу, самий прямий шлях доступу до цієї інформації і найпростіший спосіб роботи з нею. Всі доступні для користувачів метадані в системному каталозі відображаються через подання каталогу.

Ключовою особливістю такої архітектури є наявність інтегрованого системного каталогу з даними про схеми, користувачів, додатки і т.д. Передбачається, що каталог доступний як користувачам, так і функціям СУБД. Залежно від типу використовуваної СУБД кількість інформації і спосіб її

застосування можуть варіюватися. Зазвичай в системному каталозі зберігаються такі відомості:

- імена, типи і розміри елементів даних;
- імена зв'язків;
- обмеження, які накладаються на дані, для підтримки цілісності;
- імена зареєстрованих користувачів, яким надано право доступу до даних;
- зовнішня, концептуальна і внутрішня схеми, і відображення між ними;
- статистичні дані, наприклад частота транзакцій і лічильники звернень до об'єктів бази даних.

Системний каталог надає певні переваги при адмініструванні. Інформація про дані може бути централізовано зібрана і збережена, що дозволить контролювати доступ до цих даних, як і до будь-якого іншого ресурсу. Можна визначити зміст даних, що допоможе іншим користувачам зрозуміти їх призначення. Спрощується повідомлення, так як зберігаються точні визначення сенсу даних. У системному каталозі також можуть бути вказані один або кілька користувачів, які є власниками даних або мають право доступу до них. Завдяки централізованому зберіганню надмірність і суперечливість опису окремих елементів даних можуть бути легко виявлені. Внесені в базу даних зміни можуть бути запротокольовані. Наслідки будь-яких змін можуть бути визначені ще до їх внесення, оскільки в системному каталозі зафіксовані всі існуючі елементи даних, зв'язки, які встановлені між ними, а також всі їх користувачі. Заходи забезпечення безпеки можуть бути додатково посилені. З'являються нові можливості організації підтримки цілісності даних. Може виконуватися аудит інформації, що зберігається.

Для пошуку інформації про метадані можна використовувати кілька різних альтернативних способів: подання каталогів, динамічні адміністративні подання (*Dynamic Management Views, DMV*) і динамічні адміністративні функції (*Dynamic Management Functions, DMF*), системні збережені процедури, системні функції, функції властивостей і інформаційна схема.

У всіх реляційних системах баз даних системні базові таблиці мають таку ж логічну структуру, як і базові таблиці. Тому інформацію з системних базових таблиць можна отримувати за допомогою таких же інструкцій Transact-SQL, які застосовуються для вилучення інформації з базових таблиць. При цьому до системних базових таблиць можна звертатися безпосередньо. необхідно виконувати запит на інформацію з системного каталогу за допомогою існуючих інтерфейсів. Для пошуку інформації про метадані можна використовувати кілька різних альтернативних способів: подання каталогів, динамічні адміністративні подання (*Dynamic Management Views, DMV*) і динамічні адміністративні функції (*Dynamic Management Functions, DMF*), системні збережені процедури, системні функції, функції властивостей і інформаційна схема.

Інформаційна схема − стандартне рішення для доступу до метаданих, яке надає загальний інтерфейс не тільки для компонента *Database Engine*, але і для всіх існуючих реляційних систем баз даних (за умови, що конкретна система підтримує інформаційну схему).

При проектуванні та розробці програмного засобу моніторингу метаданих бази даних комп'ютерного обладнання використані запити до об'єктів інформаційної схеми. В розробленому додатку для моніторингу використано системні подання: *INFORMATION_SCHEMA.tables, INFORMATION_ SCHEMA.COLUMNS, INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints.*

При розробці програмного засобу реалізовані обробники подій на наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", " Имена ограничений".

РОЗДІЛ 1
ЗАСОБИ АДМІНІСТРУВАННЯ БАЗ ДАНИХ

1.1. Мови адміністрування баз даних

Адміністрування даних − управління інформаційними ресурсами, включаючи планування бази даних, розробку і впровадження стандартів, визначення обмежень і процедур, а також концептуальне і логічне проектування баз даних. Адміністратор даних відповідає за корпоративні інформаційні ресурси. На практиці це часто пов'язано з управлінням даними, які є спільно використовуваним ресурсом для різних користувачів і прикладних програм цієї організації. В одних випадках адміністрування даних може бути окремим функціональним завданням, а в інших − поєднуватися з адмініструванням бази даних [1].

В даний час при обмірковуванні стратегії планування інформаційної системи все більший акцент робиться на важливості адміністрування даних. Організації все більше схильні приділяти увагу значенню даних, що використовуються, або зібраних в їх інформаційній системі, як засобу досягнення більш високої конкурентоспроможності. В результаті виникає обов'язкова вимога злиття стратегії побудови інформаційних систем з бізнес-стратегіями організації. Це дозволяє створити організацію з більш гнучкою структурою, здатну адаптуватися до різких змін, що має більш творче та інноваційне внутрішнє середовище та забезпечує ефективну перебудову бізнес-процесів в разі потреби. Згадане перенесення акцентів означає, що адміністратор у все більшій мірі повинен розуміти ідеологію розвитку не тільки інформаційних систем, але і бізнес-процесів, і грати ключову роль в розробці стратегії розвитку інформаційної

системи, підтримуючи її відповідність діловим стратегіям організації.

СУБД – сукупність мовних та програмних засобів призначених для створення, підтримки і спільного користування бази даних багатьма користувачами. СУБД класифікуються в залежності від моделі на: ієрархічні (*IMS IBM*), мережеві (*dbVista*), реляційні (*Oracle, MySQL, MS SQL Server, SQLite*), об'єктно-орієнтовані (*CouchDB*), об'єктно-реляційні (*PostgreSQL, Oracle Database*), багатовимірні (*Oracle Express*), нереляційні *NoSQL* (*Amazon DynamoDB*).

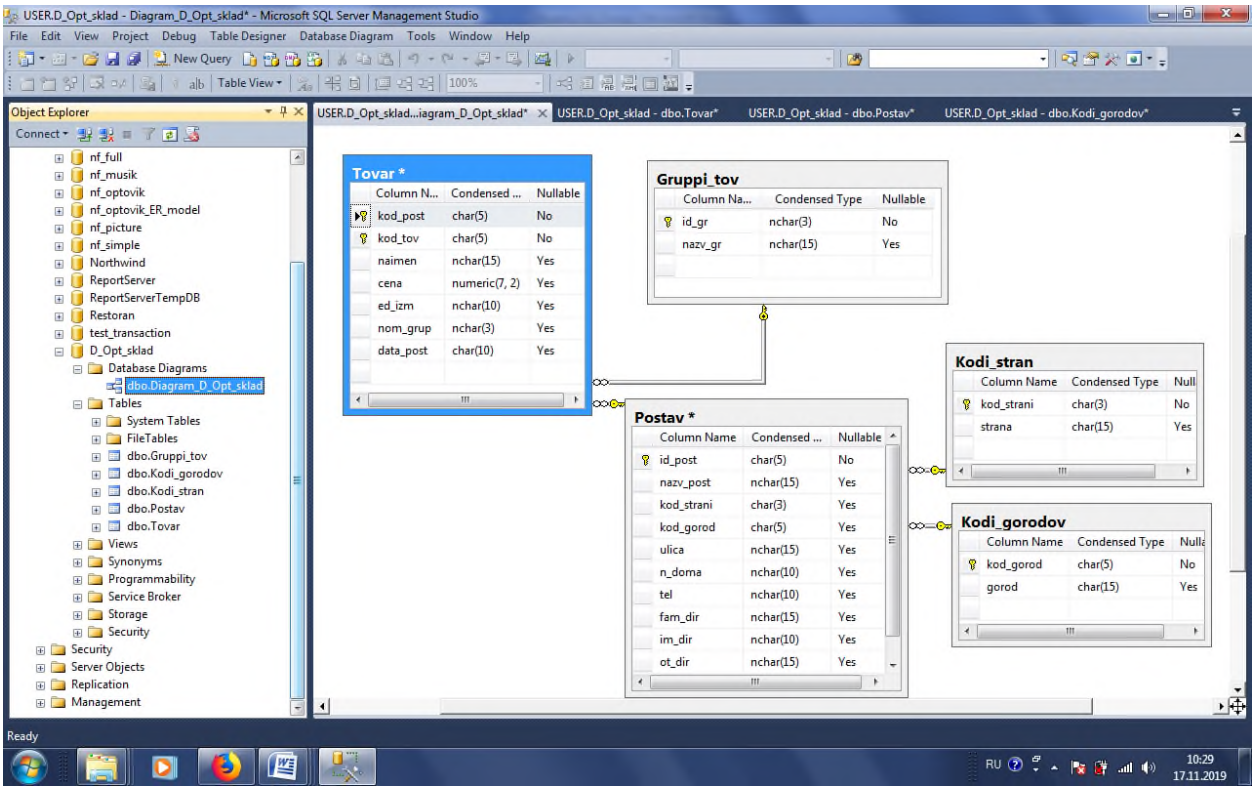

Рис. 1.1. Вікно утиліти адміністрування бази даних комп'ютерних комплектуючих

З огляду на складність і важливість функцій адміністратора баз даних, легко припустити, що для їх успішного виконання у адміністратора повинні бути спеціальні засоби адміністрування. До основних з таких засобів адміністрування можна віднести:

1) мову визначення даних;

2) мову маніпулювання даними;

3) системний каталог (або словник даних).

Для роботи з даними в СУБД передбачено внутрішню мову, що складається з двох частин: мови визначення даних (*Data Definition Language − DDL*) і мови маніпулювання даними (*Data Manipulation Language − DML*). Ці мови ще називаються підмовами даних, тому що в них відсутні конструкції для виконання всіх обчислювальних операцій, зазвичай використовуваних в мовах програмування високого рівня. У багатьох СУБД передбачена можливість впровадження операторів підмови даних в програму на мовах високого рівня. У цьому випадку мову високого рівня прийнято називати базовою.

Мова визначення даних (*DDL*) − формальний закон, який використовується в деякій моделі даних для визначення структури баз даних [2]. Результат компіляції операторів *DDL* − набір таблиць, збережених в особливих файлах, званих словниками даних або системними каталогами. За допомогою *DDL* зазвичай визначаються підрозділи даних, типові структури і правила їх композиції, присвоюються імена даним, визначаються типи елементів даних за допомогою завдання властивих їм властивостей, створюються ключі таблиць бази даних, а також визначаються зв’язки між даними, впорядкованість даних всередині їх сукупностей, правила перевірки достовірності даних і засоби захисту від неправомірного використання їх.

Зазвичай в *DDL* не визначаються техніка запам'ятовування або пошуку даних на фізичних носіях та інші особливості їх фізичної організації, що обумовлено однією з основних концепцій бази даних − незалежністю логічної структури даних від фізичних особливостей їх зберігання. *DDL* зазвичай повністю незалежна від мови маніпулювання даними. Отже, визначення даних в базах даних незалежно від програм обробки є другою важливою концепцією використання баз даних.

Мова маніпулювання даними (*DML*) − сукупність мовних засобів для організації доступу до даних в деякій моделі даних і в відповідних їй СУБД [3]. Може виступати в ролі мови запитів, яка забезпечує інформаційне обслуговування користувачів баз даних, або бути розширенням деякої мови

програмування, званої базовою мовою, з конструкціями і поняттями якої *DML* повинна бути узгодженою. Оператори *DML* дозволяють отримувати дані з баз даних, створювати або модифікувати останні.

До основних операцій маніпулювання даними відносяться:

- вставка в БД нових даних (оператор *INSERT*);
- модифікація даних, що зберігаються в БД (оператор *UPDATE*);
- отримання даних, що містяться в БД (оператор *SELECT*);
- видалення даних на БД (оператор *DELETE*).

DML відрізняються базовими конструкціями маніпулювання даними. Відрізняють два їх типи: а) процедурні *DML*; б) непроцедурні (декларативні) *DML*.

За допомогою процедурної мови користувач (програміст) вказує на те, як можна отримати необхідні дані з певного набору даних. Тобто користувач повинен визначити всі операції доступу до даних, щоб отримати результат. При цьому передбачається знання користувачем деталей внутрішньої організації структур даних в БД. За допомогою непроцедурних мов користувач вказує, які дані йому потрібні без визначення способу їх отримання. Даний підхід звільняє користувача від необхідності знати подробиці внутрішньої організації БД і надає деяку незалежність від даних [4].

У загальному випадку мова запитів − частина *DML*, високорівнева вузькоспеціалізована мова, призначена для задоволення різних вимог по вибірці даних з БД. СУБД повинна мати доступний кінцевим користувачам каталог, в якому зберігається опис елементів даних.

1.2. Системний каталог СУБД *MS SQL SERVER*

Системний каталог або словник даних − спеціальна система в складі БД, що містить інформацію про всі ресурси системи. У словнику даних (системному каталозі) містяться метадані − дані про об'єкти бази даних, що дозволяють

спростити спосіб доступу до них і управління ними. Перед доступом до реальних даних СУБД звертається до системного каталогу.

Ключовою особливістю такої архітектури є наявність інтегрованого системного каталогу з даними про схеми, користувачів, додатки і т.д. Передбачається, що каталог доступний як користувачам, так і функціям СУБД. Залежно від типу використовуваної СУБД кількість інформації і спосіб її застосування можуть варіюватися. Зазвичай в системному каталозі зберігаються такі відомості:

- імена, типи і розміри елементів даних;
- імена зв'язків;
- обмеження, які накладаються на дані, для підтримки цілісності;
- імена зареєстрованих користувачів, яким надано право доступу до даних;
- зовнішня, концептуальна і внутрішня схеми, і відображення між ними;
- статистичні дані, наприклад частота транзакцій і лічильники звернень до об'єктів бази даних.

Системний каталог надає певні переваги при адмініструванні. Інформація про дані може бути централізовано зібрана і збережена, що дозволить контролювати доступ до цих даних, як і до будь-якого іншого ресурсу. Можна визначити зміст даних, що допоможе іншим користувачам зрозуміти їх призначення. Спрощується повідомлення, так як зберігаються точні визначення сенсу даних. У системному каталозі також можуть бути вказані один або кілька користувачів, які є власниками даних або мають право доступу до них. Завдяки централізованому зберіганню надмірність і суперечливість опису окремих елементів даних можуть бути легко виявлені. Внесені в базу даних зміни можуть бути запротокольовані. Наслідки будь-яких змін можуть бути визначені ще до їх внесення, оскільки в системному каталозі зафіксовані всі існуючі елементи даних, зв'язки, які встановлені між ними, а також всі їх користувачі. Заходи забезпечення безпеки можуть бути додатково посилені. З'являються нові можливості організації підтримки цілісності даних. Може виконуватися аудит інформації, що зберігається [5].

Системний каталог СУБД є одним з фундаментальних компонентів системи. Програмні компоненти будуються на використанні даних, що зберігаються в системному каталозі. Наприклад, модуль контролю прав доступу використовує системний каталог для перевірки наявності у користувача повноважень, необхідних для виконання запитаних ним операцій. Для проведення подібної перевірки системний каталог повинен включати наступні компоненти:

- імена користувачів, яким надано дозвіл на доступ до бази даних;
- імена елементів даних в базі даних;
- елементи даних, до яких кожен користувач має право доступу, і дозволені операції доступу до них − для вставки, оновлення, видалення або читання.

Іншим прикладом можуть служити засоби перевірки цілісності даних, які використовують системний каталог для перевірки того, чи задовольняє запитана операція всім встановленим обмеженням підтримки цілісності даних. Для виконання цієї перевірки в системному каталозі повинні зберігатися такі відомості:

- імена елементів даних з бази даних;
- типи і розміри елементів даних;
- обмеження, встановлені для кожного з елементів даних.

Системний каталог складається з таблиць, що описують структуру об'єктів бази даних, таких як базові таблиці, подання, індекси і власне бази даних. Ці таблиці називаються системними базовими таблицями. Компонент *Database Engine* часто звертається до системного каталогу за інформацією, необхідною для правильного функціонування системи.

Компонент *Database Engine* відрізняє системні базові таблиці *master, model, msdb, tempdb* від базових таблиць баз даних користувача. Системні базові таблиці *master, model, msdb, tempdb* належать до системного каталогу, а системні таблиці певної бази даних зберігаються в каталозі цієї бази даних. Тому системні базові таблиці присутні лише в одному екземплярі для всієї системи (якщо вони належать виключно до бази даних *master*), тоді як інші таблиці присутні в одному екземплярі в кожній базі даних, включаючи базу даних *master* [6].

Кожна БД зберігається у двох файлах: безпосередньо файл даних з розширенням імені *.mdf,* та файл журналу транзакцій *.ldf* за шляхом, який визначається адміністратором.

1.3. Висновки до розділу

Адміністрування базами даних передбачає виконання функцій, спрямованих на забезпечення надійного та ефективного функціонування системи баз даних, адекватності змісту бази даних інформаційним потребам користувачів, відображення в базі даних актуального стану предметної області. З огляду на складність і важливість функцій адміністратора баз даних, легко припустити, що для їх успішного виконання у адміністратора повинні бути спеціальні засоби адміністрування. До основних з таких засобів адміністрування можна віднести:

1) мову визначення даних;

2) мову маніпулювання даними;

3) системний каталог (або словник даних).

Для роботи з даними в СУБД передбачено внутрішню мову, що складається з двох частин: мови визначення даних (*Data Definition Language − DDL*) і мови маніпулювання даними (*Data Manipulation Language − DML*).

Словник даних (системний каталог) − спеціальна система в складі БД, що містить інформацію про всі ресурси системи. У словнику даних (системному каталозі) містяться метадані − дані про об'єкти бази даних, що дозволяють спростити спосіб доступу до них і управління ними. Перед доступом до реальних даних СУБД звертається до системного каталогу. Системний каталог СУБД є одним з фундаментальних компонентів системи.

Системний каталог складається з таблиць, що описують структуру об'єктів бази даних, таких як базові таблиці, подання, індекси і власне бази даних. Ці таблиці називаються системними базовими таблицями. Компонент *Database Engine* часто звертається до системного каталогу за інформацією, необхідною для правильного функціонування системи.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ ПЕРЕГЛЯДУ МЕТАДАНИХ БАЗИ ДАНИХ КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ

2.1. Системні інтерфейси для перегляду метаданих

У всіх реляційних системах баз даних системні базові таблиці мають таку ж логічну структуру, як і базові таблиці. Тому інформацію з системних базових таблиць можна отримувати за допомогою таких же інструкцій Transact-SQL, які застосовуються для вилучення інформації з базових таблиць. При цьому до системних базових таблиць можна звертатися безпосередньо. Для цього необхідно виконувати запит на інформацію з системного каталогу за допомогою існуючих інтерфейсів. Для пошуку інформації про метадані можна використовувати кілька різних альтернативних способів: подання каталогів, динамічні адміністративні подання (*Dynamic Management Views, DMV*) і динамічні адміністративні функції (*Dynamic Management Functions, DMF*), системні збережені процедури, системні функції, функції властивостей і інформаційну схему.

Всі ці інтерфейси можна поділити на дві групи: загальні інтерфейси (подання каталогу, динамічні адміністративні подання і функції, інформаційна схема) і спеціалізовані інтерфейси компонента *Database Engine* (системні збережені процедури, системні функції і функції властивостей).

"Загальні" означає, що всі реляційні системи баз даних підтримують такі інтерфейси, але використовують іншу термінологію. Наприклад, в термінології Oracle, подання каталогу і динамічні адміністративні подання називаються "подання словника даних" і "подання V$" відповідно.

До загальних належать такі інтерфейси:

- подання каталогу;
- динамічні адміністративні подання і динамічні адміністративні функції;
- інформаційна схема.

Подання каталогу є найбільш загальним інтерфейсом для метаданих і надають найефективніший спосіб отримання спеціалізованих форм цієї інформації. Подання каталогу належать до схеми *sys*, тому при зверненні до цих об'єктів слід використовувати ім'я цієї схеми.

Подання каталогів є основним інтерфейсом для метаданих, що зберігаються в системних базових таблицях. Метадані описують атрибути об'єктів в системі баз даних. Подання каталогу містять інформацію, що використовується ядром бази даних *SQL Server*. Вони надають найбільш загальний інтерфейс метаданих каталогу, самий прямий шлях доступу до цієї інформації і найпростіший спосіб роботи з нею. Всі доступні для користувачів метадані в системному каталозі відображаються через подання каталогу.

Подання каталогу відповідають загальній концепції ієрархії об'єктів, в якій об'єкти нижчого рівня успадковують атрибути об'єктів більш високого рівня. Деякі подання каталогу успадковують стовпці (оскільки стовпці таблиці − атрибути сутностей, які в ній зберігаються) з інших подань каталогу. Наприклад, подання каталогу *Tables* (Таблиці) успадковує стовпці подання каталогу *Objects* (Об'єкти). Таким чином, до стовпців подання каталогу *Tables* (Таблиці) додаються всі стовпці подання каталогу *Objects* (Об'єкти).

Подання каталогу повертають дані, які використовуються компонентом *SQL Server Database Engine*. Рекомендується, щоб використовувалися подання каталогу, тому що вони мають найбільш універсальний інтерфейс до метаданих каталогу і надають найбільш ефективний спосіб для отримання, перетворення і подання налаштованих форм цих даних. Всі доступні для користувача метадані каталогу надаються через подання каталогу.

Динамічні адміністративні подання і динамічні адміністративні функції зазвичай застосовуються для перегляду активних процесів і вмісту пам'яті.

Динамічні адміністративні подання і динамічні адміністративні функції повертають інформацію про стан сервера, яку можна застосувати для спостереження над активними процесами і, таким чином, для налаштування продуктивності системи або для відстеження дійсного стану системи. На відміну від подань каталогу, *DMV* і *DMF* засновані на внутрішніх структурах системи.

Основна відмінність між поданнями каталогу і *DMV* полягає в їх застосуванні: подання каталогу надають інформацію про метадані, тоді як *DMV* і *DMF* застосовуються для доступу до динамічних властивостей системи. Іншими словами, *DMV* застосовуються для отримання інформації про базу даних, про окремі запити або окремих користувачів. *DMV* і *DMF* належать до схеми *sys*, а їх імена складаються з префікса *dm_* і текстового рядка, що вказує категорію, до якої належить *DMV* або *DMF*. Далі перераховані деякі з цих категорій:

*sys.dm_db_ ** − повертає інформацію про бази даних та їх об'єктах;

*sys.dm_tran_ ** − повертає інформацію, що має відношення до транзакцій;

*sys.dm_io_ ** − повертає інформацію про дії по вводу / виводу;

*sys.dm_exec_ ** − повертає інформацію, пов'язану з виконанням призначеного для користувача коду.

У кожній новій версії *SQL Server* корпорація *Microsoft* безперервно збільшує кількість підтримуваних *DMV*. Так *SQL Server* 2012 містить 20 нових *DMV*, а їх загальна кількість дорівнює 155.

Системні функції і функції властивостей дозволяють отримувати системну інформацію. Основна різниця між цими двома типами функцій полягає в їх структурі. Крім цього, функції властивостей повертають більше інформації, ніж системні функції.

Системні збережені процедури можна використовувати для отримання доступу до вмісту системних баз даних і модифікації цього вмісту.

Інформаційна схема − стандартне рішення для доступу до метаданих, яке надає загальний інтерфейс не тільки для компонента *Database Engine*, але і для всіх існуючих реляційних систем баз даних (за умови, що конкретна система підтримує інформаційну схему).

При проектуванні та розробці програмного засобу моніторингу метаданих бази даних комп'ютерних комплектуючих використані запити до об’єктів саме інформаційної схеми. В розробленому додатку для моніторингу використано системні подання: *INFORMATION_SCHEMA.tables, INFORMATION_ SCHEMA.COLUMNS, INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints* [8].

2.2. Подання інформаційної схеми

Інформаційна схема складається з подань, доступних тільки для читання, які надають інформацію про всі таблиці, подання і стовпчиках компонента, до яких *Database Engine* має доступ. На відміну від системного каталогу, який управляє метаданими стосовно до системи, як єдиним цілим, інформаційна схема в основному управляє середовищем бази даних [9].

Інформаційна схема була вперше представлена в стандарті *SQL92*. Компонент *Database Engine* відображає подання інформаційної схеми, щоб розроблені на інших системах баз даних програми змогли отримати свій системний каталог, не використовуючи його прямим чином. Ці стандартні подання використовують різну термінологію, тому при використанні імен стовпців вважається, що "каталог" є синонімом бази даних, а "домен" − синонімом призначеного для користувача типу даних [10].

При проектуванні програмного засобу визначені наступні функції моніторингу параметрів: перегляд таблиць, шляхів до розташування баз даних комп’ютерних комплектуючих, перегляд імен таблиць, перегляд імен колонок для всіх таблиць, перегляд імен колонок для конкретної таблиці, перегляд типів даних для таблиці «Товар», перегляд первинних та зовнішніх ключів, перегляд метаданих про обмеження. Було використано запити мови *Transact-SQL*, системні подання, функції та змінні.

При проектуванні програмного засобу визначені наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования",

"Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", " Имена ограничений".

Найбільш важливою командою мови маніпулювання даними є команда вибору *SELECT*. Для побудови запитів у роботі використано оператор *SELECT*. У структуру запиту оператора *SELECT* можуть бути включені додаткові оператори: уточнюючі умова вибірки, що виконують угруповання, сортування вихідних значень і т.д. Оператор *SELECT* дозволяє формувати запит до бази даних. В результаті виконання цього оператора СУБД формує результуючий набір даних. Синтаксис команди:

SELECT[ALL | DISTINCT] select_list

[INTO [new_table_name]]

[FROM { table_name | view_name } [(optimizer_hints)]

[[, { table_name2 | view_name2 } [(optimizer_hints)]

[..., table_name16 | view_name16 } [(optimizer_hints)]]]

[WHERE clause]

[GROUP BY clause]

[HAVING clause]

[ORDER BY clause]

ALL – опція за замовчуванням, виводить всі колонки в результаті запиту. *DISTINCT* виводить унікальні колонки в результаті запиту. *select_list* визначає колонки результату запиту. Може бути, наприклад: символ *, який виводить всі колонки в результаті запиту; перелік імен колонок, розділених комами.

INTO new_table_name – виводить результати запиту, *new_table_name* може бути БД, масивом, текстовим файлом, екраном або принтером. Крім цього, інформація може бути переслана в так званий “курсор”. Курсор – це тимчасовий набір даних, який може бути областю пам’яті або тимчасовим файлом і має режим “тільки читання”.

FROM вказує ім’я таблиці *table_name* або подання *view_name*, які використовуються як джерела даних, припускається використання максимум 16 таблиць або подань, перелік яких розділяється комами. Опція (*optimizer_hints*) вказує методи блокування, індексування.

WHERE вказує критерій відбору. Вираз *clause* вказує вираз для відповідного операнда, *clause* може містити логічні вирази *AND, OR, NOT*, знаки відношення =,>,>=,<,<=, оператори *LIKE, BETWEEN, IN*. Вираз *clause* може задавати групування даних, яке задається операндами *GROUP BY, HAVING*.

GROUP BY вказує стовпчики, по яких виконується групування вихідних даних. Усі записи БД, для яких значення стовпчиків збігаються, відображаються у вибірці єдиним рядком. Групування використовують для отримання деяких звітних характеристик (сум, кількості) групи. *HAVING* задає критерій відбору даних у кожну сформовану в процесі вибору групу.

ORDER BY вказує порядок сортування. За замовчуванням сортування відбувається за збільшенням (*ASC*), але може бути виконано і за зменшенням (*DESC*).

При проектуванні програмного засобу використано найбільш важливі подання інформаційної схеми. Подання *information_schema.tables* повертає один рядок для кожної таблиці в поточній базі даних, до якої користувач має доступ. Це подання отримує інформацію з системного каталогу, використовуючи подання каталогу *sys.objects*. У таблиці нижче подано опис чотирьох стовпців цього поданння: *TABLE_CATALOG* тип *nvarchar* (128) − кваліфікатор таблиці; *TABLE_SCHEMA* тип *nvarchar* (128) − ім'я схеми, що містить таблицю; *TABLE_NAME* тип *sysname* − ім'я таблиці або подання; *TABLE_TYPE* тип *varchar* (10) − тип таблиці, може бути *VIEW* або *BASE TABLE* [11].

В роботі для пункту меню "Имена таблиц" спроектовано наступний запит для бази даних комп'ютерних комплектуючих *Dipl_Computer_komplekt*:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_type
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.tables

Подання *information_schema.columns* повертає один рядок для кожного стовпця, доступного активного користувача в поточній базі даних. Це подання отримує інформацію з системного каталогу, використовуючи подання каталогу *sys.columns* і *sys.objects*. Наведено опис стовпців цього подання, які було використано: *TABLE_CATALOG* тип *nvarchar* (128) − ім'я каталогу (бази даних), до якого належить стовпець; *TABLE_SCHEMA* тип *nvarchar* (128) − ім'я схеми, до якої належить стовпець; *TABLE_NAME* тип *nvarchar* (128)− ім'я таблиці, до якої належить стовпець; *COLUMN_NAME* тип *nvarchar*(128)− ім'я стовпця; *ORDINAL_POSITION* тип int –номер по порядку стовпчика; *DATA_TYPE* − тип даних стовпчика; *COLLATION_NAME* тип *nvarchar* (128) − унікальне ім'я для параметрів сортування, якщо стовпець має символьні дані або текстовий тип даних, інакше повертається значення *NULL*; *COLUMN_DEFAULT* тип *nvarchar* (4000). Значення стовпця за замовчуванням; *DATA_TYPE* тип *nvarchar* (128) − тип даних, підтримуваний системою; *CHARACTER_MAXIMUM_LENGTH* тип *int* максимальна довжина в символах для двійкових даних, символьних даних або текстових даних і зображень (-1 для даних типу *XML* і великих значень, інакше повертається значення *NULL*); *NUMERIC_SCALE* тип *int* − допустима розмірність числових даних, точних числових даних, цілих чи грошових даних (загальна кількість цифр форматі десяткового вигляду без коми), іншому випадку повертається NULL; *NUMERIC_PRECISION* тип *tinyint* − точність для числових даних, точних числових даних, цілих чи грошових даних (кількість цифр після коми в форматі десяткового вигляду), в іншому випадку повертається *NULL*. Розмірність (7,2) (*NUMERIC_SCALE* − *7*) означає загальну кількість цифр при введенні – 7, а точність 2 означає кількість цифр (*NUMERIC_PRECISION* − 2) після коми в форматі десяткового вигляду [11].

В роботі для пункту меню "Имена колонок для всех таблиц" спроектовано наступний запит для бази даних комп’ютерного обладнання *Dipl_Computer_komplekt*:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,COLLATION_NAME

FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS

В роботі для пункту меню "Имена колонок для конкретной таблицы" спроектовано наступний запит для бази даних комп'ютерних комплектуючих *Dipl_Computer_komplekt*:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, COLUMN_DEFAULT
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = N'Tovar'

В роботі для пункту меню "Типы данных для таблицы ТОВАР" спроектовано наступний запит для бази даних комп’ютерного обладнання *Dipl_Computer_komplekt*:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION, DATA_TYPE, CHARACTER_MAXIMUM_LENGTH, NUMERIC_SCALE, NUMERIC_PRECISION
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS
WHERE TABLE_NAME = N'Tovar'

Подання *INFORMATION_SCHEMA.key_column_usage* повертає один рядок для кожного стовпця, використаного як обмеження-ключ в поточній базі даних. Це подання інформаційної схеми повертає відомості про об'єкти, на які у поточного користувача є дозволи. Далі наведено опис стовпців цього подання, які було використано: *TABLE_CATALOG* тип *nvarchar* (128) − ім'я каталогу (бази даних), до якого належить стовпець; *TABLE_SCHEMA* тип *nvarchar* (128) − ім'я схеми, до якої належить стовпець; *TABLE_NAME* тип *nvarchar* (128)− ім'я таблиці, до якої належить стовпець; *COLUMN_NAME* тип *nvarchar*(128)− ім'я стовпця; *ORDINAL_POSITION* тип int – номер по порядку стовпчика; *DATA_TYPE* − тип даних стовпчика.

В роботі для пункту меню "Ключи - первичные, внешние" спроектовано наступний запит для бази даних комп’ютерного обладнання *Dipl_Computer_komplekt*:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.key_column_usage
WHERE TABLE_NAME = N'Tovar'

Подання *INFORMATION_SCHEMA.table_constraints* містить по одному рядку для кожного табличного обмеження в поточній базі даних. Це подання інформаційної схеми повертає відомості про об'єкти, на які у поточного користувача є дозволи. Далі наведено опис стовпців цього подання, які було використано: *CONSTRAINT_CATALOG* тип *nvarchar* (128) − кваліфікатор обмеження; *CONSTRAINT_SCHEMA* тип *nvarchar* (128) − ім'я схеми, що містить обмеження; *CONSTRAINT_NAME* тип *sysname* − ім'я обмеження; *TABLE_CATALOG* тип *nvarchar* (128) − кваліфікатор таблиці; *TABLE_SCHEMA* тип *nvarchar* (128) − ім'я схеми, що містить таблицю; *TABLE_NAME* тип *sysname* − ім'я таблиці; *CONSTRAINT_TYPE* тип *varchar* (11) − тип обмеження (*CHECK, UNIQUE, PRIMARY KEY, FOREIGN KEY*).

Обмеження (*Constraints*) використовуються для обмеження типу даних, які можуть задаватися. *NOT NULL* − гарантує, що стовпець не може мати значення *NULL*; *UNIQUE* − гарантує, що всі значення в стовпці різні; *Primary Key* (первинний Ключ) − комбінація NOT NULL і UNIQUE, унікально ідентифікує кожен рядок в таблиці; *FOREIGN KEY* − унікально ідентифікує рядок/запис в іншій таблиці; *CHECK* − гарантує, що всі значення в стовпці задовольняють певній умові; *DEFAULT* − встановлює значення за замовчуванням для стовпця, якщо значення не вказано; *INDEX* − використовується для пришвидшення створення і отримання даних з бази даних.

В роботі для пункту меню "Имена ограничений" спроектовано наступний запит для бази даних комп’ютерного обладнання *Dipl_Computer_komplekt*:

SELECT constraint_CATALOG, constraint_SCHEMA, constraint_NAME, TABLE_NAME, constraint_type
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.table_constraints
WHERE TABLE_NAME = N'Tovar'

СУБД підтримують декілька моделей даних. Основна модель, яка реалізована у багатьох СУБД – реляційна. Проектування бази даних комп'ютерних комплектуючих виконано згідно методиці проектування реляційних моделей – нормалізації відношень. Модель бази даних комп’ютерних комплектуючих складається з наступних таблиць: таблиця постачальників *Postav*, таблиця товарів *Tovar*, таблиця групп товарів *Gruppi*, таблиця кодів стран *Kodi_stran*, таблиця кодів міст *Kodi_gorodov*. Таблиці пов’язані первинними та зовнішніми ключами (рис. 3.1) [12].

База даних міститься в двох файлах: файлі даних (розширення імені фізичного файлу *.mdf* та файлу журналу транзакцій (розширення імені фізичного файлу *.ldf*. Кожна база даних *SQL Server* має журнал транзакцій, в якому фіксуються всі зміни даних, здійснені в кожній з транзакцій. Журнал транзакцій є критичним компонентом бази даних і в разі системного збою використовується для приведення бази даних в узгоджений стан.

2.3. Висновки до розділу

При проектуванні програмного засобу визначені наступні функції моніторингу параметрів: перегляд таблиць, шляхів до розташування баз даних комп’ютерного обладнання, перегляд імен таблиць, перегляд імен колонок для всіх таблиць, перегляд імен колонок для конкретної таблиці, перегляд типів даних для таблиці «Товар», перегляд первинних та зовнішніх ключів, перегляд метаданих про обмеження. Було використано запити мови *Transact-SQL*, системні подання, функції та змінні.

При проектуванні програмного засобу визначені наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена

колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", " Имена ограничений".

При проектуванні програмного засобу моніторингу метаданих бази даних комп’ютерного обладнання використані запити до об’єктів інформаційної схеми. В розробленому додатку для моніторингу використано системні подання інформаційної схеми: *INFORMATION_SCHEMA.tables, INFORMATION_ SCHEMA.COLUMNS, INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints.*

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ ПЕРЕГЛЯДУ МЕТАДАНИХ БАЗИ ДАНИХ КОМП'ЮТЕРНИХ КОМПЛЕКТУЮЧИХ

3.1. Склад файлів рішення. Розробка головної форми

Програмний засіб розроблено в середовищі *Microsoft Visual Studio 2019* на мові програмування *C#* з використанням СУБД *MS SQL Server 2016*. Файл рішення має ім'я *WindowsFormsApplication1*, файл проекту *WindowsFormsApplicationOptSklad* розробленої програми перегляду та налаштувань параметрів журналу транзакцій складається з наступних основних файлів: *FormOptSklad.cs, Dipl_Computer_komplektDataSet.designer.cs, та Program.cs* [13].

Файл *FormOptSklad.cs* містить обробники подій головної форми *Form1*. Файл *FormOptSklad.designer.cs* зберігає код, створений конструктором форм *Windows*. Файл *Dipl_Computer_komplektDataSet.designer.cs* зберігає код, що автоматично створено конструктором запитів до бази даних. Компонент *Program.cs.* зберігає головну точку входу для програми. Створено дві форми: основна *Form1* та про автора *FormAbout.*

При розробці програмного засобу було реалізовані наступні горизонтальні пункти меню: «Таблицы», «Просмотр метаданных», «Об авторе программы», «Выход». Пункт меню «Таблицы» здійснює перегляд таблиць бази даних комп'ютерних комплектуючих. Пункт меню «Таблицы» містить наступні вертикальні підпункти для перегляду таблиць бази даних комп'ютерного обладнання: «Поставщики оборудования», «Товары», «Группы товаров», «Коды стран», «Коды городов» (для перегляду таблиці постачальників *Postav*, таблиці

товарів *Tovar*,таблиці групп товарів *Gruppi*, таблиці кодів стран *Kodi_stran*, таблиці кодів міст *Kodi_gorodov* відповідно).

При розробці програмного засобу реалізовані обробники подій на наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", " Имена ограничений".

При створенні рішення *WindowsFormsApplication1* та проекту *WindowsFormsApplicationOptSklad* виконано наступне. При створенні програми в середовищі *Visual Studio* першим кроком є вибір типу (шаблону) проекту. Для кожного типу проекту *Visual Studio* задає параметри компілятора і автоматично створює стартовий код [14].

У меню середовища *Visual Studio* обрано *File* послідовно обрано пункти *New* і *Project*. У діалоговому вікні *New Project* на лівій панелі розкрито вузол *Installed***,** вузол *Templates*, вузол *Visual C#*, у списку встановлених шаблонів у списку праворуч обрано *Windows Forms Application* – додатки *Windows*. В полі *Name* введено ім'я проекту *WindowsFormsApplicationOptSklad***.** В полі *Location* вказано шлях до файлів рішення (папка *D:\diplom*). Під час створення проекту середовище *Visual Studio* поміщає проект в рішення. За замовчуванням ім'я рішення збігається з ім'ям проекту. Для того, щоб розрізнити імена файлів рішення і проекту за ім'ям, в полі *Solution Name* введено ім'я файлів рішення *WindowsFormsApplication1*.

Всі файли рішення і проекту відобразяться на панелі перегляду елементів *Solution Explorer*. Всі файли рішення і проекту записані в папку *D:\diplom\WindowsFormsApplication1***,** яка містить файл проекту *WindowsFormsApplicationOptSklad.* В папці *D:\diplom* міститься файл рішення *WindowsFormsApplication1.sln*.

При розробці основної форми виконано наступне (рис. 4.1). Встановлено властивості компонента форма *Form*. Обрано конструктор *Windows Forms*, що

відображає форму *Form1* створеного проекту та виконано наступне. Обрано за компонент *Form1*. В групі *Appearance* встановлено властивість *Text* «Просмотр метаданных базы данных компьютерных комплектующих».

В групі *Layout* встановлено властивість *WindowState* в значення *Maximized*; властивість *WindowState* визначає розмір вікна: *Maximized* – максимальний, *Minimized* – зменшений до розмірів заголовка, *Normal* – довільний, визначений у процесі проектування.

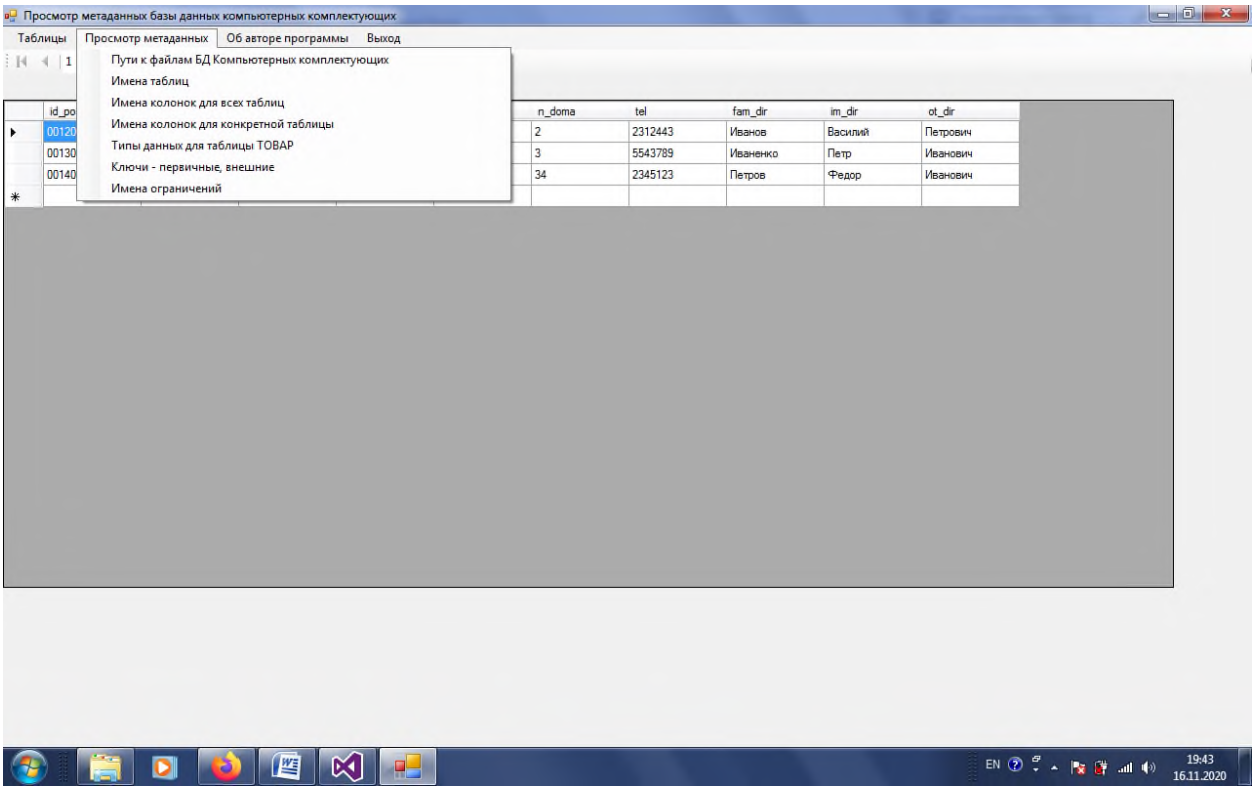

Рис. 3.1. Головне вікно програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих

Для створення меню виконано наступне. Додано до форми компонент *menuStrip* із списку *All Windows* панелі елементів *Toolbox*. Обрано на формі компонент *menuStrip1***.** В групі *Data* розкрито властивість *Items*. У вікні *Items Collection Editor* в полі вводу *Select items and add to list below* обрано *MenuItems, Add*. У списку властивостей в групі *Appearance* встановлено значення властивості *Text* для першого елемента горизонтального меню: «Таблицы».

В групі *Design* встановлено властивість *Name* компонента *toolStripMenuItem1* такою, щоб дорівнювала *toolStripMenuItemTabl* для пункта меню «Таблицы». Для горизонтальних пунктів меню властивості *Name* для компонентів *toolStripMenuItem* встановлено відповідно в: *toolStripMenuItemAdm* для пункта меню «Просмотр метаданных», *toolStripMenuItemObAvt* для пункта меню «Об авторе программы», *toolStripMenuItemVihod* для пункта меню «Выход».

Для додавання вертикальних пунктів меню виконано наступне. Після вибору горизонтального пункту «Просмотр метаданных» *toolStripMenuItemAdm* у вікні *Properties* обрано пункт меню *DropDownItems.* Увікні *Items Collection Editor* в полі вводу *Select items and add to list below* обрано *MenuItems*, потім *Add*. У списку властивостей в групі *Appearance* встановлено значення властивості *Text* для першого елемента вертикального меню: «Просмотр путей к файлам баз данных». Встановлено властивість *Name* в *ToolStripMenuItemProsmPutiCompKompl*. Аналогічно додано всі вертикальні пункти меню. Для вертикальних пунктів меню властивості *Name* для компонентів *toolStripMenuItem* встановлено відповідно в: *toolStripMenuItemProsmTables, ToolStripMenuItemProsmColNameм ToolStripMenuItemColNameTable, ToolStripMenuItemTypeData, ToolStripMenuItemKeys, ToolStripMenuItemConstraints.*

Розроблено інформаційну панель, на якій міститься напис виконуваного процесу. Використано компоненту *Panel,* властивість *Name* встановлено в *panelMenuInf.* Використано компоненту *Label,* властивість *Name* встановлено в *LabelMenuInf,* властивість *Text* для неї встановлено в коді в залежності від назви пункту меню. Для вертикальних пунктів меню властивість *Text* встановлено в: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", " Имена ограничений".

3.2. Розробка модулів моніторингу

3.2.1. Модулі перегляду структури таблиць

Для організації з'єднання з базою даних *SQL Server* виконано наступне. В меню *Project* обрано *Add New Data Source.* У вікні *Data Source Configuration Wizard,* на сторінці *Choose Your Data Connection* у переліку джерел даних *Data source list* обрано *Database,* обрано *Next.*

Для створення підключення до бази даних комп'ютерних комплектуючих *D_Opt_sklad* обрано *New Connection*, у вікні *Add Connection* у випадаючому списку *Server name* обрано ім'я сервера *User_Radchenko*. В групі *Connect to a database* в списку *Select enter a dabase name* обрано ім'я бази даних *D_Opt_sklad,* в якій розташовується таблиця. У вікні *Data Adapter Configuration Wizard* з'явиться ім'я з'єднання, наприклад, *user. D_Opt_sklad.dbo* (рядок *Data Source= User_Radchenko;Initial Catalog=Dipl_Computer_komplekt;Integrated Security=True*), далі обрано *Next, dataset, Next.*

На сторінці *Choose Your Database Objects* обрано вузол *Database*. Автоматично буде згенеровано компоненту *DataSet* та файл *Dipl_Computer_komplektDataSet.xsd.* Компонента *DataSet* використовується як буфер для зберігання таблиць та інших об’єктів з сервера. Клас *DataSet* спеціально сконструйований для доступу до даних незалежно від джерела даних. Тому він може бути використаний з багатьма різними джерелами даних. *DataSet* містить колекцію одного або декількох об'єктів *DataTable*, які складаються з рядків і стовпців даних, а також містить відображення даних про первинний ключ, зовнішній ключ, обмеження та пов'язані відомості про дані в об'єктах *DataTable*. *DataAdapter* поміщає результати запиту у відповідну таблицю *DataTable,* що знаходиться в *DataSet*.

При вирішенні питання про те, використовувати *DataReader* або *DataSet,* слід врахувати, для чого призначений додаток. *DataSet* може бути призначений для виконання наступних завдань: локальне кешування даних в додатку для

подальшої обробки (якщо потрібно тільки зчитувати результати запиту, клас *DataReader* підходить якнайкраще); віддалена взаємодія з даними або веб-службою *XML*; динамічна взаємодія з даними, наприклад, прив'язка до елементу управління *Windows Forms* або комбінування і зв'язування даних з декількох джерел; виконання інтенсивної обробки, яка не вимагає відкритого з'єднання з джерелом даних, що звільняє з'єднання для використання іншими клієнтами.

Схема алгоритму програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих наведено на Рис.3.2.

Command − об'єкт, який представляє один з двох класів: або клас *OleDbCommand*, або клас *SqlCommand*. Основне призначення об'єкта *Command* − виконання різних дій над базою даних (джерелом даних) при використанні відкритого з'єднання. Самі ж дії зазвичай кодуються оператором *SQL* або збереженою процедурою. Закодована інформація фіксується з використанням об'єктів − представників класу *Parameter*, спеціально розроблених для "запису" інформації в команді. Тобто після встановлення з'єднання з базою даних для зміни стану цієї бази може бути створений, відповідним чином налаштований і застосований об'єкт − представник класу *Command*.

Як було зазначено, об'єкт *Command* забезпечує управління джерелом даних, яке полягає: у виконанні *DML (Data Manipulation Language*) запитів − запитів, які не повертають дані (операції *INSERT, UPDATE, DELETE*); у виконанні *DDL (Data Definition Language*) запитів − запитів, які змінюють структуру бази баних (*CREATE*).

Об'єкт класу *Command* представлений двома класами − *SqlCommand* і *OleDb Command*. Дозволяє виконувати команди для бази даних і при цьому використовує встановлене з'єднання. Виконувані команди можуть бути представлені: збереженими процедурами; командами *SQL*; операторами, які повертають цілі таблиці. Об'єкт-представник класу *Command* підтримує два варіанти методів (варіанти методів визначаються базовим класом): *ExecuteNonQuery* − забезпечує виконання команд, які не повертають дані, наприклад *INSERT, UPDATE, DELETE*; *ExecuteScalar* − виконує запити до бази

даних, які повертають єдине значення; *ExecuteReader* − повертає результуючий набір через об'єкт *DataReader*.

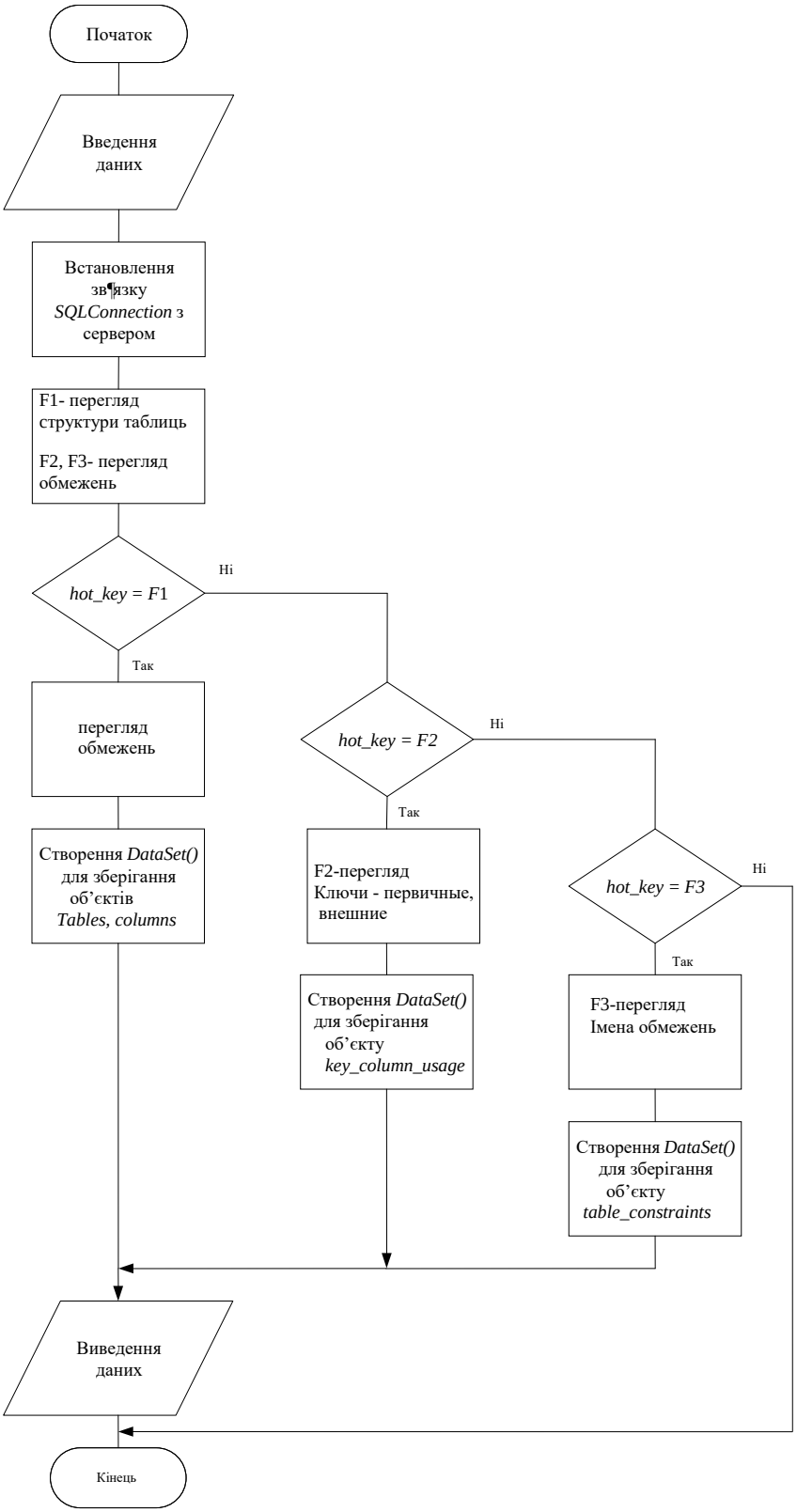

Рис. 3.2. Схема алгоритму програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих

Доступ до даних в *ADO.NET* за допомогою *Data Provider*'а здійснюється наступним чином:

1. Об'єкт-представник класу *Connection* встановлює з'єднання між базою даних та програмою.

2. Це з'єднання стає доступним об'єктам *Command* і *DataAdapter*.

3. При цьому об'єкт *Command* дозволяє виконувати команди безпосередньо над базою даних.

4. Якщо виконується команда, яка повертає кілька значень, *Command* відкриває доступ до них через об'єкт *DataReader*.

5. Результати виконання команди обробляються або безпосередньо, з використанням коду програми, або через об'єкт *DataSet*, який заповнюється за допомогою об'єкта *DataAdapter*.

6. Для поновлення бази даних застосовують також об'єкти *Command* і *DataAdapter*.

Отже, в будь-якому випадку, незалежно від обраного постачальника даних, при роботі з даними в *ADO .NET* використано: *Connection Object* − для встановлення з'єднання з базою даних; *Dataset Object* − для представлення даних на стороні додатку; *Command Object* − для зміни стану бази. Способи створення об'єкта *Command*: з використанням конструкторів і з подальшим налаштуванням об'єкта (рядка запиту і об'єкта *Connection*); виклик методу *CreateCommand* об'єкта *Connection* [15].

Для організації зв’язку форми з даними таблиці *postav* виконано наступне. Обрано пункт меню *View, Other windows, Data Sources.* На панелі *Data Sources* обрано таблицю *postav* та переміщено її на форму. На формі з'являться компоненти *DataGridView* для візуального відображення даних таблиці, компонента *BindingNavigator* для переміщення по записах таблиці. Компонети *D_Opt_skladDataSet, postavBindingSource, postavTableAdapter, tableAdapterManager, postavBindingNavigator* автоматично з'являться на панелі для невидимих компонент. Властивість *DataMember* компоненти

postavBindingSource встановлено автоматично *Postav*. При організації доступу до даних за допомогою меню *Visual Studio* компоненти *DataSet, BindingSource, TableAdapter, tableAdapterManager, BindingNavigator* з'являться автоматично на панелі для невидимих компонент.

Для відображення назви таблиці, яку буде завантажено у вікно, перенесено на форму проекту компоненту *Panel*, в групі *Design* властивість *Name* вказано *panelMenuInf.* Для відображення даних таблиці *Postav* на формі візуальний компонент *dataGridView* налаштовано наступним чином. В групі *Design* властивість *Name* вказано *postavDataGridView,* в групі *Data* властивість *DataSource* вказано *postavBindingSource,* властивість *Visible* вказано *false.* В групі *Layout* обрано властивість *Anchor,* вказано *Top, Left*. Властивість *Dock* вказано *Fill.*

Для зв'язку форми з даними таблиці *postav* компоненту *BindingSource* налаштовано наступним чином. Властивість *Name* вказано *postavBindingSource,* властивість *DataSource* вказано *DataSetD_Opt_sklad,* властивість *DataMember* компоненти *postavBindingSource* встановлено *Postav*.

Для організації зв'язку форми з даними таблицями *tovar, gruppi_tov, kodi_stran, kodi_gorodov* аналогічним чином налаштовано компоненти зв'язку форми з даними *BindingSource.* Властивість *Name* вказано *tovarBindingSource, gruppi_tovBindingSource, kodi_stranBindingSource, kodi_gorodovBindingSource* відповідно. Для налаштування візуальної компоненти для відображення даних на формі використано відповідно компоненти *DataGridView,* властивість *Name* вказано *DataGridViewtovar, DataGridViewgruppi_tov, DataGridViewkodi_stran, DataGridViewkodi_gorodov* відповідно. Властивість *DataSource* вказано *tovarBindingSource, gruppi_tovBindingSource, kodi_stran BindingSource, kodi_gorodovBindingSource* відповідно. При розробці доступу до таблиці компоненти *tovarTableAdapter, gruppi_tovTableAdapter, kodi_stranTableAdapter, kodi_gorodovTableAdapter* автоматично з'являться на панелі для невидимих компонент.

Для створення оброблювача події для відображення даних таблиці *Postav* на пункт меню «Поставщики оборудования» виконано наступне. У вікні *Object Inspector,* на панелі *Properties* обрано компонент *toolStripMenuItemPostav*, далі на вкладці *Events* обрано подію *Click* відповідним чином. Після чого згенеровано заголовок оброблювача події, в тіло якого введено код. Для відображення інформації про виконуваний пункт меню властивість *Text* компоненти *labelMenuInf* встановлено в "Поставщики", для підключення навігації по таблиці властивість *BindingSource* компоненти *BindingNavigator* встановлено в *postavBindingSource*. Для компоненту *postavTableAdapter* використано метод *Update* з аргументом, який вказує таблицю *Postav* з компонентом для роботи з від'єднаними даними *DataSetD_Opt_sklad*. Виклик методу *Update* адаптера таблиці виконує інструкції, створені під час першого налаштування адаптера, метод *Update* дозволяє оновити зроблені в проекті зміни в самій базі даних.

Для того, щоб зробити невидимими компоненти *dataGridViewSysaltfiles, dataGridViewIS_Tables,, dataGridViewColName, dataGridViewColNameTable, dataGridViewTypeData, dataGridViewKeys, dataGridViewConstraints* властивість *Visible* для них у відповідних оброблювачах подій встановлено в *false*. Для виведення інформації властивість *Visible* у відповідних оброблювачах подій встановлено в *true*.

Далі для компоненту *postavTableAdapter* використано метод *Fill* з аргументом, який вказує таблицю *Postav* з компоненти для роботи з від'єднаними даними *Dipl_Computer_komplektDataSet*. Виклик методу *Fill* адаптера таблиці заповнює значеннями елемент *DataSet*, який використовується для кешування даних. Для візуального відображення властивість *Visible* компоненти *dataGridViewPostav* встановлено в *true*.

Оброблювачі подій аналогічно створені для таблиці товарів *Tovar*, таблиці групп товарів *Gruppi*, таблиці кодів стран *Kodi_stran*, таблиці кодів міст *Kodi_gorodov*.

Для створення оброблювачів подій на пункти меню "Просмотр метаданных", з вертикальними пунктами меню "Пути к файлам БД

Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", "Имена ограничений" було налаштовано компоненти *sqlConnectionCompKompl*, компоненту *DataSet*, властивість *Name* якої встановлено в *DataSetdipl_Computer_komplekt*, властивість *DataSetName* якої встановлено в *Dipl_Computer_komplektDataSet*. Схема *DataSetdipl_Computer_komplekt* містить компоненти *TableAdapter* з запитами до системних подань *INFORMATION_SCHEMA.tables*, *INFORMATION_SCHEMA.COLUMNS*, *INFORMATION_SCHEMA.key_column_usage*, *INFORMATION_SCHEMA.table_ constraints* (рис. 3.1).

В модулі перегляду структури таблиць задіяно наступні подання інформаційної схеми *INFORMATION_SCHEMA.tables*, *INFORMATION_SCHEMA.COLUMNS*.

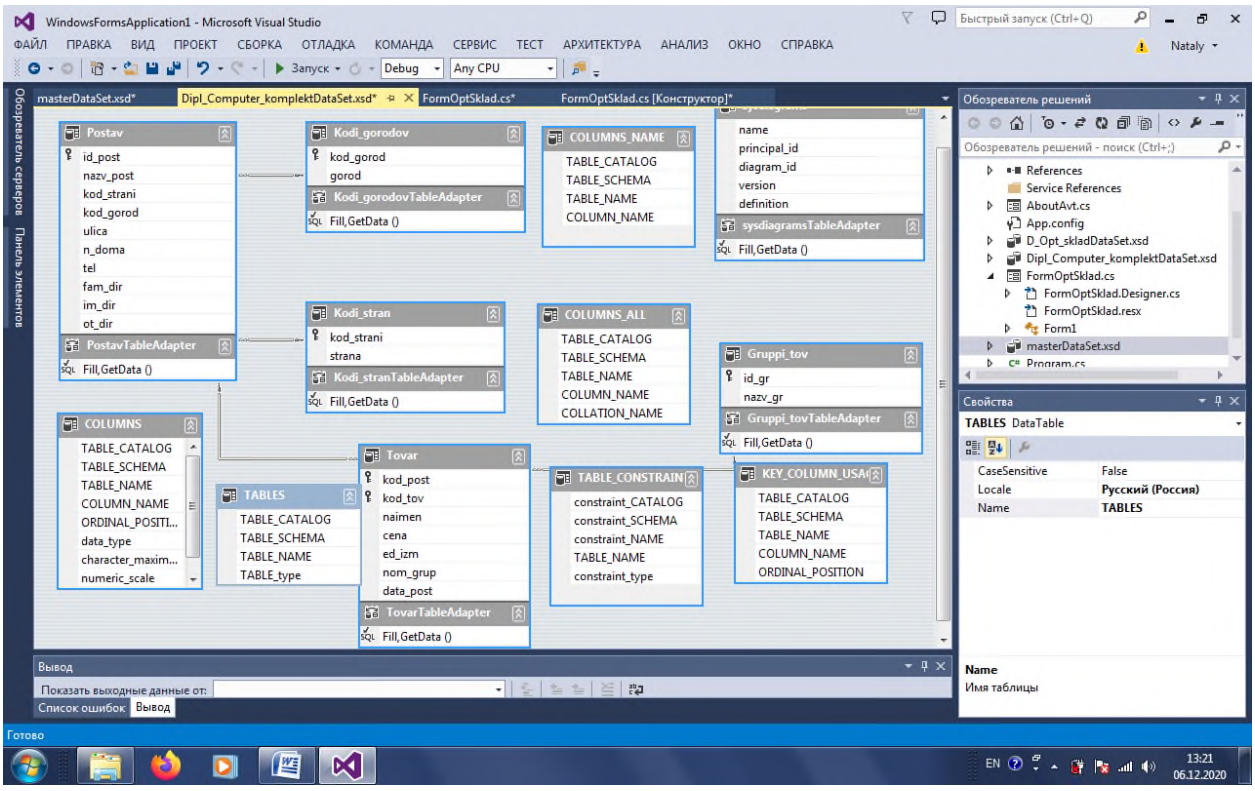

Рис. 3.3. Вікно конструктора схем наборів даних *_Dipl_Computer_komplektDataSet* для бази даних комп'ютерних комплектуючих

Розроблені компоненти *TableAdapter* конструктора схем наборів даних *_Dipl_Computer_komplektDataSet* мають значення властивості *Name: TABLES, COLUMNS_ALL, COLUMNS_NAME, COLUMNS*.

В модулі перегляду структури таблиць для побудови запитів на формі використано компоненти *sqlDataAdapter*. Для вертикальних пунктів меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР" налаштовано відповідно наступні компоненти: *sqlDataAdapterIS_Tables, sqlDataAdapterColName, sqlDataAdapterColNameTable, sqlDataAdapterTypeData.*

Для компоненту *sqlDataAdapterIS_Tables*, пункт меню "Имена таблиц" (Рис.3.2), властивість *CommandText* містить розроблений запит:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_type
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.tables

Для компоненту *sqlDataAdapterColName*, пункт меню "Имена колонок для всех таблиц" (Рис.3.3), властивість *CommandText* містить розроблений запит:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,COLLATION_NAME
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS

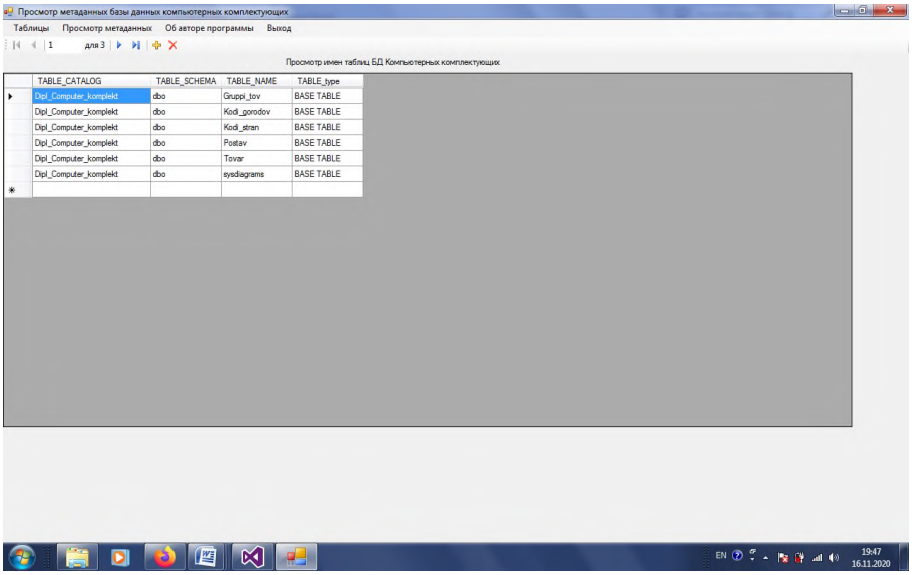

Рис. 3.4. Вікно форми програмного засобу для пункту меню «Имена таблиц»

Для компоненти *sqlDataAdapterTypeData*, пункт меню"Типы данных для таблицы ТОВАР" (Рис.3.4), властивість *CommandText* містить розроблений запит:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION, data_type, character_maximum_length, numeric_scale, NUMERIC_PRECISION

FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS

WHERE TABLE_NAME = N'Tovar'

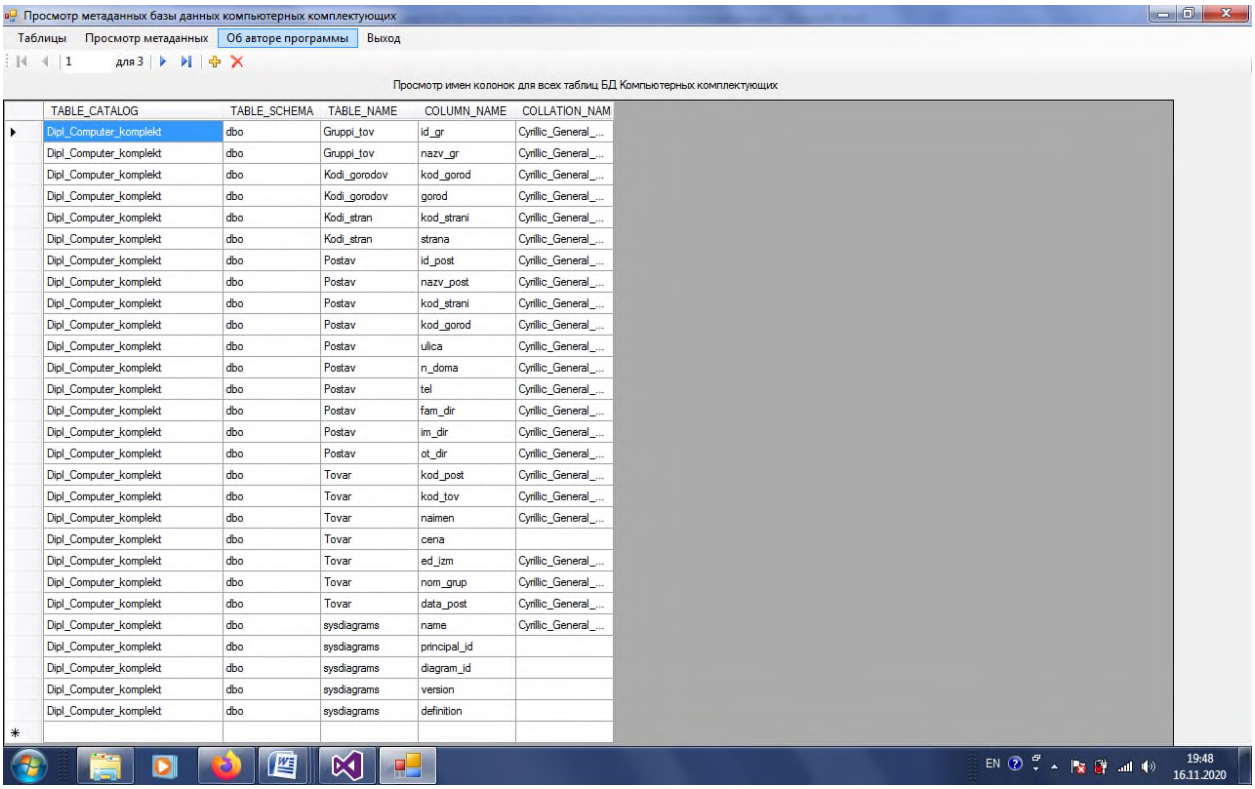

Рис. 3.5. Вікно форми програмного засобу для пункту меню «Имена колонок для всех таблиц»

Для створення оброблювача події на пункт меню «Просмотр путей к файлам баз данных» було виконано наступне. При налаштуванні компоненти *sqlDataAdapter_sysalytfiles* було вказано запит до системного подання *sysalytfiles: select fileid, groupid, size, maxsize, growth, dbid, name, filename from sys.sysaltfiles.*

Використано компоненту *LabelMenuInf,* властивість *Text* для неї встановлено в коді "Просмотр путей к файлам баз данных". Для компоненти *sqlDataAdapter_sysalytfiles* використано метод *Fill* з аргументом. Метод *Fill* об'єкта *DataAdapter* служить для заповнення набору даних *DataSet* результатами

виконання методу *SelectCommand* об'єкта *DataAdapter*. Для того, щоб зробити невидимими компоненти *DataGridViewPostav, DataGridViewTovar* та компоненти *dataGridView* для інших таблиць, *dataGridViewModel, dataGridViewLog_space_usage, dataGridViewtovar, listBox1* властивість *Visible* для них встановлено в *false*. Властивість *ReadOnly* компоненти *dataGridViewSysaltfiles* встановлено в *true* для того, щоб унеможливити редагування цієї компоненти. Для візуального відображення властивість *Visible* компоненти *dataGridViewPostav* встановлено в *true*.

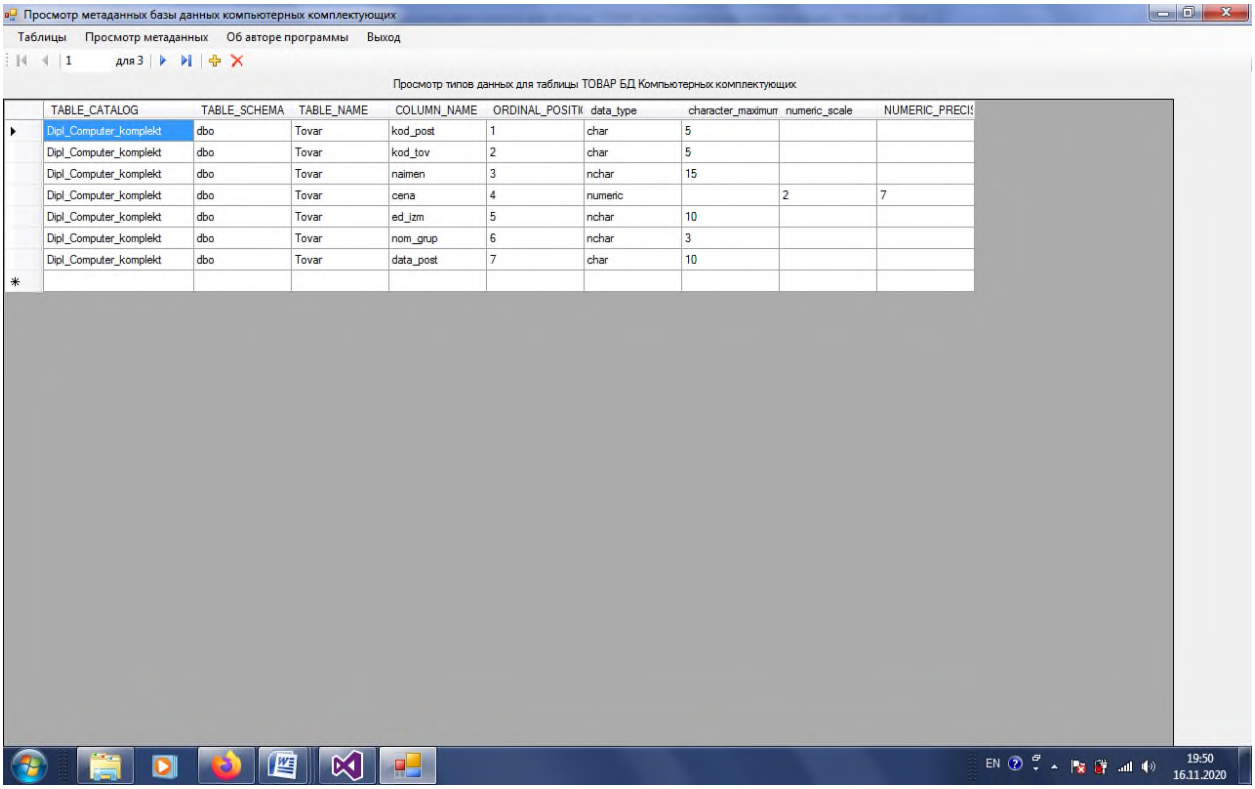

Рис. 3.6. Вікно форми програмного засобу для пункту меню «Типы данных для таблицы TOBAP»

3.2.3. Модулі перегляду обмежень

В модулі перегляду обмежень задіяно наступні подання інформаційної схеми *INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints.*

Компоненти *TableAdapter* конструктора схем наборів даних *_Dipl_Computer_komplektDataSet* мають значення властивості *Name: KEY_COLUMN_USAGE, TABLE_CONSTRAINTS.*

В модулі перегляду обмежень для побудови запитів на формі використано компоненти *sqlDataAdapter*. Для вертикальних пунктів меню "Ключи - первичные, внешние", " Имена ограничений" налаштовано відповідно наступні компоненти: *sqlDataAdapterKeys*, *sqlDataAdapterConstraints.*

Для компоненти *sqlDataAdapterKeys*, пункт меню "Ключи - первичные, внешние" (Рис.3.5), властивість *CommandText* містить розроблений запит:

SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME, ORDINAL_POSITION
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.key_column_usage
WHERE TABLE_NAME = N'Tovar'

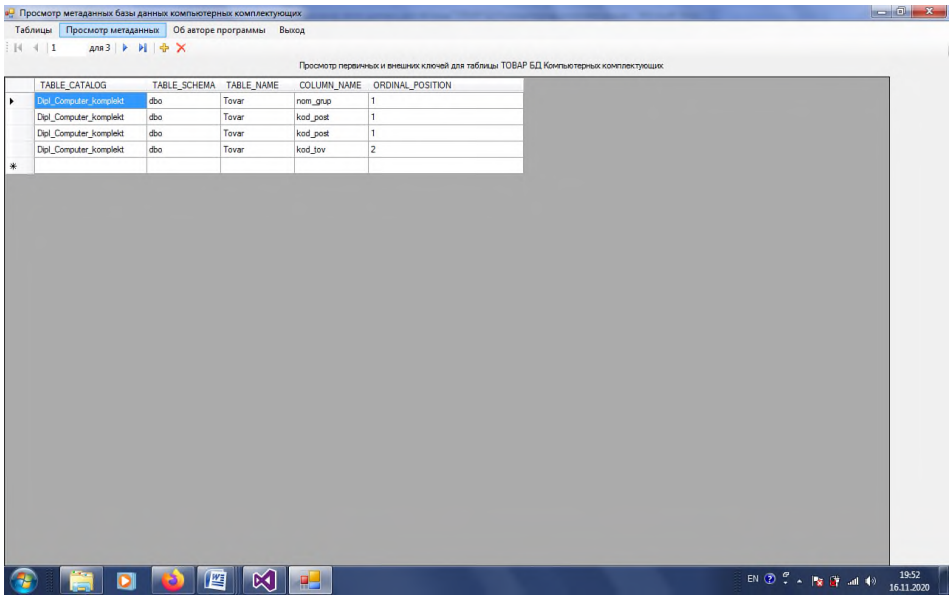

Рис. 3.7. Вікно форми програмного засобу для пункту меню «Ключи - первичные, внешние»

В модулі перегляду обмежень для побудови запитів на формі використано компоненти *sqlDataAdapter*. Для вертикальних пунктів меню "Ключи - первичные, внешние", "Имена ограничений" налаштовано відповідно наступні компоненти: *sqlDataAdapterKeys.*

Для компоненти *sqlDataAdapterConstraints*, пункт меню "Имена ограничений" (Рис.3.6), властивість *CommandText* містить розроблений запит:

SELECT constraint_CATALOG, constraint_SCHEMA, constraint_NAME, TABLE_NAME, constraint_type
FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.table_constraints
--WHERE TABLE_NAME = N'Tovar'

Для налаштування розмірів компонент *dataGridView* для властивості *Location* використано функцію *Point(0, 75)*, для властивості *Size* використано функцію *Size(1200, 800*) простору імен *System.Drawing:*

dataGridViewConstraints.Location = new System.Drawing.Point(0, 75);
dataGridViewConstraints.Size = new System.Drawing.Size(1200, 800);

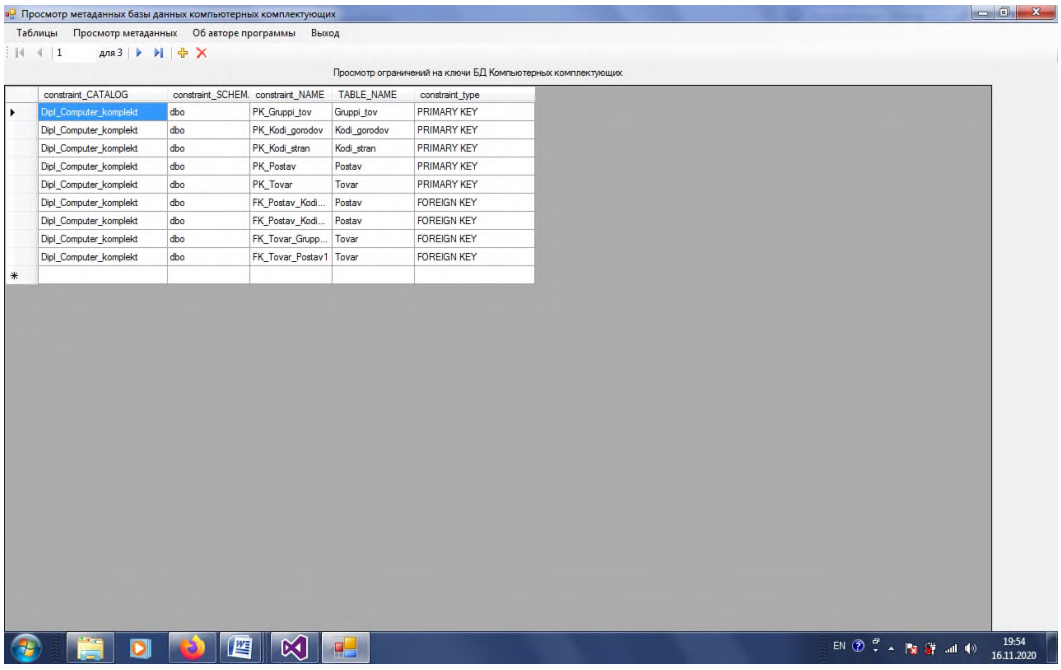

Рис. 3.8. Вікно форми програмного засобу для пункту меню «Имена ограничений»

При розробці програмного засобу реалізовані обробники подій наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран". Для пункту меню "Просмотр метаданных" реалізовано обробники подій для вертикальних пунктів меню: "Пути к файлам БД Компьютерных комплектующих" (оброблювач події *ToolStripMenuItemProsmLogFiles_Click*), "Имена таблиц"

(оброблювач події *ToolStripMenuItemProsmColName_Click*), "Имена колонок для всех таблиц" (оброблювач події *toolStripMenuItemProsmTables_Click*), "Имена колонок для конкретной таблицы" (оброблювач події *ToolStripMenuItemColNameTable_Click*), "Типы данных для таблицы ТОВАР" (оброблювач події *ToolStripMenuItemTypeData_Click*), "Ключи - первичные, внешние" (оброблювач події *ToolStripMenuItemKeys_Click*), " Имена ограничений" (оброблювач події *ToolStripMenuItemConstraints_Click*).

3.3. Висновки до розділу

Для розробки проекту було використано *Visual Studio 2019*. Програмний засіб був написаний на мові програмування *C#* з використанням *MS SQL Server* і запитів на мові *Transact-SQL*.

При розробці програмного засобу реалізовані відповідні оператори мови *SQL*, системні подання, функції та змінні, використано компоненти середовища розробки *Visual Studio*. Для доступу до даних використано компоненти *sqlDataAdapter, DataSet, dataGridView*. Використано методи відповідних класів. Наведено схему алгоритму програмного засобу перегляду метаданих бази даних комп'ютерних комплектуючих.

При розробці програмного засобу реалізовані обробники подій наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих" (оброблювач події *ToolStripMenuItemProsmLogFiles_Click*), "Имена таблиц" (оброблювач події *ToolStripMenuItemProsmColName_Click*), "Имена колонок для всех таблиц" (оброблювач події *toolStripMenuItemProsmTables_Click*), "Имена колонок для конкретной таблицы" (оброблювач події *ToolStripMenuItemColNameTable_Click*), "Типы данных для таблицы ТОВАР" (оброблювач події *ToolStripMenuItemTypeData_Click*), "Ключи - первичные, внешние" (оброблювач

події *ToolStripMenuItemKeys_Click*), " Имена ограничений" (оброблювач події *ToolStripMenuItemConstraints_Click*).

В модулі перегляду структури таблиць задіяно наступні подання інформаційної схеми *INFORMATION_SCHEMA.tables, INFORMATION_ SCHEMA.COLUMNS.* В модулі перегляду обмежень задіяно наступні подання інформаційної схеми *INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints.*

ВИСНОВКИ

Адміністрування базами даних передбачає виконання функцій, спрямованих на забезпечення надійного та ефективного функціонування системи баз даних, адекватності змісту бази даних інформаційним потребам користувачів, відображення в базі даних актуального стану предметної області.

Словник даних (системний каталог) − спеціальна система в складі БД, що містить інформацію про всі ресурси системи. У словнику даних (системному каталозі) містяться метадані − дані про об'єкти бази даних, що дозволяють спростити спосіб доступу до них і управління ними. Перед доступом до реальних даних СУБД звертається до системного каталогу. Системний каталог СУБД є одним з фундаментальних компонентів системи.

Системний каталог складається з таблиць, що описують структуру об'єктів бази даних, таких як базові таблиці, подання, індекси і власне бази даних. Ці таблиці називаються системними базовими таблицями. Компонент *Database Engine* часто звертається до системного каталогу за інформацією, необхідною для правильного функціонування системи.

Для адміністрування баз даних використовують системні подання, мову структурованих запитів *SQL*. Подання – об'єкт бази даних, що є результатом виконання запиту до бази даних, визначеного за допомогою оператора *SELECT*, в момент звернення до подання.

Для роботи з даними в СУБД передбачено внутрішню мову, що складається з двох частин: мови визначення даних (*Data Definition Language − DDL*) і мови маніпулювання даними (*Data Manipulation Language − DML*). Ці мови ще називаються підмовами даних, тому що в них відсутні конструкції для виконання всіх обчислювальних операцій, зазвичай використовуваних в мовах програмування високого рівня. У багатьох СУБД передбачена можливість впровадження операторів підмови даних в програму на мовах високого рівня. У цьому випадку мову високого рівня прийнято називати базовою. Мова визначення

даних (*DDL*) − формальний закон, який використовується в деякій моделі даних для визначення структури баз даних.

Найбільш важливою командою мови маніпулювання даними є команда *SELECT*. У структуру запиту оператора *SELECT* можуть бути включені багато додаткові оператори: уточнюючі умови вибірки, що виробляють угруповання, сортування вихідних значень. Оператор *SELECT* дозволяє формувати запит до бази даних. В результаті виконання цього оператора СУБД формує результуючий набір даних.

При проектуванні програмного засобу визначені наступні функції моніторингу параметрів: перегляд таблиць, шляхів до розташування баз даних комп'ютерних комплектуючих, перегляд імен таблиць, перегляд імен колонок для всіх таблиць, перегляд імен колонок для конкретної таблиці, перегляд типів даних для таблиці «Товар», перегляд первинних та зовнішніх ключів, перегляд метаданих про обмеження. Було використано запити мови *Transact-SQL*, системні подання, функції та змінні.

При проектуванні програмного засобу визначені наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих", "Имена таблиц", "Имена колонок для всех таблиц", "Имена колонок для конкретной таблицы", "Типы данных для таблицы ТОВАР", "Ключи - первичные, внешние", " Имена ограничений".

При проектуванні програмного засобу моніторингу метаданих бази даних комп'ютерного обладнання використані запити до об'єктів інформаційної схеми. В розробленому додатку для моніторингу використано системні подання: *INFORMATION_SCHEMA.tables, INFORMATION_ SCHEMA.COLUMNS, INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints*. Описано спроектовані команди мови *SQL*.

В модулі перегляду структури таблиць задіяно наступні подання інформаційної схеми *INFORMATION_SCHEMA.tables, INFORMATION_*

SCHEMA.COLUMNS. В модулі перегляду обмежень задіяно наступні подання інформаційної схеми *INFORMATION_SCHEMA.key_column_usage, INFORMATION_SCHEMA.table_ constraints.*

При розробці програмного засобу реалізовані обробники подій наступні основні пункти меню: "Таблицы" з вертикальними пунктами меню "Поставщики оборудования", "Товары", "Группы товаров", "Коды городов", "Коды стран"; "Просмотр метаданных" з вертикальними пунктами меню "Пути к файлам БД Компьютерных комплектующих" (оброблювач події *ToolStripMenuItemProsmLogFiles_Click*), "Имена таблиц" (оброблювач події *ToolStripMenuItemProsmColName_Click*), "Имена колонок для всех таблиц" (оброблювач події *toolStripMenuItemProsmTables_Click*), "Имена колонок для конкретной таблицы" (оброблювач події *ToolStripMenuItemColNameTable_Click*), "Типы данных для таблицы ТОВАР" (оброблювач події *ToolStripMenuItemTypeData_Click*), "Ключи - первичные, внешние" (оброблювач події *ToolStripMenuItemKeys_Click*), " Имена ограничений" (оброблювач події *ToolStripMenuItemConstraints_Click*).

Для розробки проекту було використано середовище *Visual Studio 2013* [16]. Програмний засіб був написаний на мові програмування *C#* [17] з використанням *MS SQL Server* і запитів на мові *Transact-SQL* [18], при оформленні використано відповідні нормативні документи [19] та положення [20].

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Проект компанії *Microsoft*, призначений для технічних фахівців з адміністрування програмного забезпечення [Електронний ресурс] – Режим доступу: *https://technet.microsoft.com/ru-ru/library/ms179472(v=sql.105).aspx* (дата звернення 10.01.2021).
2. Офіційний сайт компанії *Microsoft* [Електронний ресурс] – Режим доступу: *https://msdn.microsoft.com/ru-ru/library/ms178028(v=sql.90).aspx* (дата звернення 10.01.2021).
3. Рикарди Г., Системы баз данных. СПб.: «Вильямс», 2001. – 201с.
4. Бондарь А. *Microsoft SQL Server* 2014. СПб.: «БХВ-Петербург », 2015. – 592 с.
5. Джеймс Р. Грофф, Пол Н. Вайнберг. *SQL*: Полное руководство. – 3-е изд.: Пер.с англ. – М.: Издательский дом "Диалектика-Вильямс", 2012. – 960с.
6. Фейерштейн, С. *Oracle PL/SQL* для профессионалов; СПб: Питер, 2012. – 540с.
7. Система керування базами даних *MS SQL SERVER*: Лабораторний практикум з дисципліни “Організація баз даних”. Модуль № 1 /Уклад.: Н. Ф. Халімон. –К.: НАУ, 2014. – 74с.
8. Роберт Виейра. Програмування баз даних *Microsoft SQL Server 2005*. Базовий курс = Программирование баз данных *Microsoft SQL Server 2005*. Базовый курс. – М.: «Діалектика», 2007. – 832с.
9. Рудикова Л. Базы даных, Разработка приложений СПб.: «БХВ-Петербург», 2006. – 487с.
10. Джоунс Е., Функции *SQL* Справочник программиста, М.: «Диалектика» 2007. – 767с.
11. Конноли, Т. Базы данных. Проектирование, реализация и сопровождение. – 3-е изд. – М.: Изд. Дом «Вильямс» 2003. – 1440с.
12. *Microsoft® SQL Server™ 2005*.Реализация и обслуживание. Учебный курс *Microsoft* (Экзамен 70-431). – М.: «Питер», 2007. –767с.

13. Розробка додатків на платформі *.Net Framework* в інтегрованому середовищі *Visual Studio 2013* на мові *C#*: Лабораторний практикум до дисципліни “Організація баз даних” /Уклад.: Н.Ф.Халімон, І.М.Сябрук, І.Ф. Кашкевич. –К.: НАУ, 2016. – 68 с.
14. Майо Д. Самоучитель *Microsoft Visual Studio 2010 – Microsoft Visual Studio 2010: A Beginner's Guide (A Beginners Guide)*. – С.: «БХВ-Петербург», 2010. – 464с.
15. Тернстрем Т. *Microsoft SQL SERVER 2008* разработка баз данных – М. «Русская редакция», 2010 – 494с.
16. Эндрю Троелсен. Язык программирования *C#* 5.0 и платформа *.NET* 4.5, 6-е издание = *Pro C#* 5.0 *and the .NET 4.5 Framework, 6th edition*. − М.: «Вильямс», 2013. − 1312с.
17. Джон Скит. C# для профессионалов: тонкости программирования, 3-е издание, новый перевод = *C# in Delphi, 3rd edition*. − М.: «Вильямс», 2014. − 608с.
18. Кириллов, В.В. Введение в реляционные базы данных. Введение в реляционные базы данных / В.В. Кириллов, Г.Ю. Громов. — СПб.: БХВ-Петербург, 2012. − 464 с.
19. Про прийняття національних стандартів України, змін та поправок до національних стандартів України, гармонізованих з міжнародними нормативними документами [Електронний ресурс] – Режим доступу: *https://zakon.rada.gov.ua/rada/show/v0305774-16* (дата звернення 10.01.2021).
20. Слободян О. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: Видавництво НАУ, 2017. – 63с.

Додаток А

FormOptSklad.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void postavBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            //this.postavBindingSource.EndEdit();
           // this.tableAdapterManager.UpdateAll(this.nf_optovikDataSet);

        }

        private void Form1_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "DataSetdipl_Computer_komplekt.Tovar". При необходимости она может быть перемещена или удалена.
            this.TableAdapterTovarCK.Fill(this.DataSetdipl_Computer_komplekt.Tovar);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "DataSetdipl_Computer_komplekt.Postav". При необходимости она может быть перемещена или удалена.
            this.TableAdapterPostavCK.Fill(this.DataSetdipl_Computer_komplekt.Postav);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "DataSetD_Opt_sklad.Tovar". При необходимости она может быть перемещена или удалена.
            this.tovarTableAdapter.Fill(this.DataSetD_Opt_sklad.Tovar);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "d_Opt_skladDataSet.Postav". При необходимости она может быть перемещена или удалена.
            this.postavTableAdapter.Fill(this.DataSetD_Opt_sklad.Postav);
        }
```

```
private void toolStripMenuItemPostav_Click(object sender, EventArgs e)
{
   labelMenuInf.Text = "Поставщики";
   BindingNavigator.BindingSource = postavBindingSource;
   postavTableAdapter.Update(DataSetD_Opt_sklad.Postav);
   dataGridViewSysaltfiles.Visible = false;
   dataGridViewModel.Visible = false;
   dataGridViewLog_space_usage.Visible = false;
   dataGridViewtovar.Visible = false;
   listBox1.Visible = false;
   dataGridViewtovar.Visible = false;
   dataGridViewIS_Tables.Visible = false;
   dataGridViewColName.Visible = false;
   dataGridViewColNameTable.Visible = false;
   dataGridViewTypeData.Visible = false;
   dataGridViewKeys.Visible = false;
   postavTableAdapter.Fill(DataSetD_Opt_sklad.Postav);
   //DataGridViewPostav.Dock = DockStyle.Fill;
   DataGridViewPostav.Location = new System.Drawing.Point(0, 75);
   DataGridViewPostav.Size = new System.Drawing.Size(1200, 500);
   DataGridViewPostav.Visible = true;
}

private void ToolStripMenuItemTovar_Click(object sender, EventArgs e)
{
   labelMenuInf.Text = "Товары";
   BindingNavigator.BindingSource = tovarBindingSource;
   tovarTableAdapter.Update(DataSetD_Opt_sklad.Tovar);
   dataGridViewSysaltfiles.Visible = false;
   dataGridViewModel.Visible = false;
   dataGridViewLog_space_usage.Visible = false;
   listBox1.Visible = false;
   dataGridViewIS_Tables.Visible = false;
   dataGridViewColName.Visible = false;
   dataGridViewColNameTable.Visible = false;
   dataGridViewTypeData.Visible = false;
   dataGridViewKeys.Visible = false;
   DataGridViewPostav.Visible = false;
   listBox1.Visible = false;
   tovarTableAdapter.Fill(DataSetD_Opt_sklad.Tovar);
   //dataGridViewtovar.Dock = DockStyle.Fill;
   dataGridViewtovar.Location = new System.Drawing.Point(0, 75);
   dataGridViewtovar.Size = new System.Drawing.Size(1200, 500);
   dataGridViewtovar.Visible = true;
}
private void toolStripMenuItem1_Click(object sender, EventArgs e)
{
   Close();
}

private void toolStripMenuItemObAvt_Click(object sender, EventArgs e)
{
```

```
        AboutAvt FormAboutAvt = new AboutAvt();
        FormAboutAvt.ShowDialog(this);
    }

    private void ToolStripMenuItemProsmPutiCompKompl_Click(object sender, EventArgs e)
    {
        //выполняется в dataGridViewSysaltfiles
        //SELECT fileid, groupid, size, maxsize, growth, dbid, name, filename
        //FROM sys.sysaltfiles WHERE (name = 'Dipl_Computer_komplekt') ORDER BY name
        labelMenuInf.Text = "Просмотр путей к файлам БД Компьютерных
комплектующих";
        masterDataSet1.sysaltfiles.Clear();
        this.sqlDataAdapterpPutiCK.Fill(this.masterDataSet1.sysaltfiles);
        DataGridViewPostav.Visible = false;
        dataGridViewModel.Visible = false;
        dataGridViewLog_space_usage.Visible = false;
        dataGridViewtovar.Visible = false;
        listBox1.Visible = false;
        dataGridViewIS_Tables.Visible = false;
        dataGridViewColName.Visible = false;
        dataGridViewColNameTable.Visible = false;
        dataGridViewTypeData.Visible = false;
        dataGridViewKeys.Visible = false;
        dataGridViewConstraints.Visible = false;
        dataGridViewSysaltfiles.ReadOnly = true;
        //dataGridViewSysaltfiles.Dock = DockStyle.Fill;
        dataGridViewSysaltfiles.Location = new System.Drawing.Point(0, 75);
        dataGridViewSysaltfiles.Size = new System.Drawing.Size(1300, 500);
        dataGridViewSysaltfiles.Visible = true;
    }

    private void toolStripMenuItemProsmTables_Click(object sender, EventArgs e)
    {
        //выполняется в masterDataSet
        //SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, TABLE_type
        //FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.tables
        labelMenuInf.Text = "Просмотр имен таблиц БД Компьютерных комплектующих";
        masterDataSet1.TABLES.Clear();
        this.sqlDataAdapterIS_Tables.Fill(this.masterDataSet1.TABLES);
        DataGridViewPostav.Visible = false;
        dataGridViewModel.Visible = false;
        dataGridViewLog_space_usage.Visible = false;
        dataGridViewtovar.Visible = false;
        listBox1.Visible = false;
        dataGridViewSysaltfiles.Visible = false;
        dataGridViewColName.Visible = false;
        dataGridViewColName.Visible = false;
        dataGridViewColNameTable.Visible = false;
        dataGridViewTypeData.Visible = false;
        dataGridViewKeys.Visible = false;
        dataGridViewConstraints.Visible = false;
        // ОГРАНИЧЕННОЕ ПРОСТРАНТВО
```

```
            //dataGridViewSysaltfiles.Dock = DockStyle.Fill;
            dataGridViewIS_Tables.Location = new System.Drawing.Point(0, 75);
            dataGridViewIS_Tables.Size = new System.Drawing.Size(1200, 500);
            dataGridViewIS_Tables.ReadOnly = true;
            dataGridViewIS_Tables.Visible = true;
        }

        private void ToolStripMenuItemProsmColName_Click(object sender, EventArgs e)
        {
            //выполняется в Dipl_Computer_komplektDataSet
            //SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
COLUMN_DEFAULT, COLLATION_NAME
            //FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS
            labelMenuInf.Text = "Просмотр имен колонок для всех таблиц БД Компьютерных
комплектующих";
            DataSetdipl_Computer_komplekt.COLUMNS_ALL.Clear();

this.sqlDataAdapterColName.Fill(this.DataSetdipl_Computer_komplekt.COLUMNS_ALL);
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            dataGridViewtovar.Visible = false;
            listBox1.Visible = false;
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewIS_Tables.Visible = false;
            dataGridViewColNameTable.Visible = false;
            dataGridViewTypeData.Visible = false;
            dataGridViewKeys.Visible = false;
            dataGridViewConstraints.Visible = false;
            dataGridViewColName.Location = new System.Drawing.Point(0, 75);
            dataGridViewColName.Size = new System.Drawing.Size(1200, 800);
            dataGridViewColName.ReadOnly = true;
            dataGridViewColName.Visible = true;
        }

        private void ToolStripMenuItemColNameTable_Click(object sender, EventArgs e)
        {
            //выполняется в Dipl_Computer_komplektDataSet
            //SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
COLUMN_DEFAULT, COLLATION_NAME
            //FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS
            labelMenuInf.Text = "Просмотр имен колонок для таблицы ТОВАР БД
Компьютерных комплектующих";
            DataSetdipl_Computer_komplekt.COLUMNS_NAME.Clear();

this.sqlDataAdapterColNameTable.Fill(this.DataSetdipl_Computer_komplekt.COLUMNS_NAM
E);
            DataGridViewPostav.Visible = false;
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            dataGridViewtovar.Visible = false;
            listBox1.Visible = false;
```

```
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewIS_Tables.Visible = false;
            dataGridViewColName.Visible = false;
            dataGridViewTypeData.Visible = false;
            dataGridViewKeys.Visible = false;
            dataGridViewConstraints.Visible = false;
            dataGridViewColNameTable.Location = new System.Drawing.Point(0, 75);
            dataGridViewColNameTable.Size = new System.Drawing.Size(1200, 800);
            dataGridViewColNameTable.ReadOnly = true;
            dataGridViewColNameTable.Visible = true;

        }

        private void ToolStripMenuItemTypeData_Click(object sender, EventArgs e)
        {
            //выполняется в Dipl_Computer_komplektDataSet
            //SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
ORDINAL_POSITION, data_type, character_maximum_length, numeric_scale,
NUMERIC_PRECISION
            //FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS
            //WHERE TABLE_NAME = N'Tovar'
            labelMenuInf.Text = "Просмотр типов данных для таблицы ТОВАР БД
Компьютерных комплектующих";
            DataSetdipl_Computer_komplekt.COLUMNS.Clear();
            this.sqlDataAdapterTypeData.Fill(this.DataSetdipl_Computer_komplekt.COLUMNS);
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            dataGridViewtovar.Visible = false;
            listBox1.Visible = false;
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewIS_Tables.Visible = false;
            dataGridViewColName.Visible = false;
            dataGridViewColNameTable.Visible = false;
            dataGridViewKeys.Visible = false;
            dataGridViewConstraints.Visible = false;
            dataGridViewTypeData.Location = new System.Drawing.Point(0, 75);
            dataGridViewTypeData.Size = new System.Drawing.Size(1200, 800);
            dataGridViewTypeData.ReadOnly = true;
            dataGridViewTypeData.Visible = true;
        }

        private void ToolStripMenuItemKeys_Click(object sender, EventArgs e)
        {
            //выполняется в Dipl_Computer_komplektDataSet
            //SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
ORDINAL_POSITION, data_type, character_maximum_length, numeric_scale,
NUMERIC_PRECISION
            //FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.COLUMNS
            //WHERE TABLE_NAME = N'Tovar'
            labelMenuInf.Text = "Просмотр первичных и внешних ключей для таблицы ТОВАР
БД Компьютерных комплектующих";
            DataSetdipl_Computer_komplekt.KEY_COLUMN_USAGE.Clear();
```

```
this.sqlDataAdapterKeys.Fill(this.DataSetdipl_Computer_komplekt.KEY_COLUMN_USAGE);
            dataGridViewModel.Visible = false;
            dataGridViewLog_space_usage.Visible = false;
            dataGridViewtovar.Visible = false;
            listBox1.Visible = false;
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewIS_Tables.Visible = false;
            dataGridViewColName.Visible = false;
            dataGridViewColNameTable.Visible = false;
            dataGridViewTypeData.Visible = false;
            dataGridViewConstraints.Visible = false;
            dataGridViewKeys.Location = new System.Drawing.Point(0, 75);
            dataGridViewKeys.Size = new System.Drawing.Size(1200, 800);
            dataGridViewKeys.ReadOnly = true;
            dataGridViewKeys.Visible = true;
        }

        private void ToolStripMenuItemConstraints_Click(object sender, EventArgs e)
        {
            //выполняется в Dipl_Computer_komplektDataSet
            //SELECT TABLE_CATALOG, TABLE_SCHEMA, TABLE_NAME, COLUMN_NAME,
ORDINAL_POSITION, data_type, character_maximum_length, numeric_scale,
NUMERIC_PRECISION
            //имена ограничений - первичный и внешний ключи
            //SELECT constraint_CATALOG, constraint_SCHEMA, constraint_NAME,
TABLE_NAME, constraint_type
            //FROM Dipl_Computer_komplekt.INFORMATION_SCHEMA.table_constraints
            //--WHERE TABLE_NAME = N'Tovar'
            labelMenuInf.Text = "Просмотр ограничений на ключи БД Компьютерных
комплектующих";
            DataSetdipl_Computer_komplekt.TABLE_CONSTRAINTS.Clear();

this.sqlDataAdapterConstraints.Fill(this.DataSetdipl_Computer_komplekt.TABLE_CONSTRAIN
TS);
            dataGridViewLog_space_usage.Visible = false;
            dataGridViewtovar.Visible = false;
            listBox1.Visible = false;
            dataGridViewSysaltfiles.Visible = false;
            dataGridViewIS_Tables.Visible = false;
            dataGridViewColName.Visible = false;
            dataGridViewColNameTable.Visible = false;
            dataGridViewTypeData.Visible = false;
            dataGridViewKeys.Visible = false;
            dataGridViewConstraints.Location = new System.Drawing.Point(0, 75);
            dataGridViewConstraints.Size = new System.Drawing.Size(1200, 800);
            dataGridViewConstraints.ReadOnly = true;
            dataGridViewConstraints.Visible = true;
        }

    }
}
```