МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
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 адміністрування бази даних банківських операцій_____

_____

_____

Виконавець: _____Савченко В.О.

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 денна

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____Литвиненко О.Є.
“_____”_____ 2021 р.

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

Савченка Віталія Олександровича
(прізвище, ім’я, по батькові випускника в родовому відмінку)

1. Тема дипломної роботи (проекту): Програмний засіб адміністрування бази даних банківських операцій

затверджена наказом ректора від "04" лютого 2021 року № 135/ст.

2. Термін виконання роботи (проекту): з 17.05.2021 до 20.06.2021

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

4. Зміст пояснювальної записки:
1) Автоматизація банківської діяльності.
2) Проектування програмного засобу адміністрування бази даних банківських операцій.
3) Розробка програмного засобу адміністрування бази даних банківських операцій

5. Перелік обов’язкового графічного (ілюстративного) матеріалу:
1) Вікно конструктора схеми наборів даних «Банк»
2) Головне вікно «адміністрування первинної файлової групи бази даних банківських операцій»
3) Вікно перегляду зовнішніх ключів бази даних банківських операцій
4) Вікно перегляду первинних ключів бази даних банківських операцій

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1	Ознайомитись з постановкою задачі дипломного проектування.	17.05.2021 – 18.05.2021	
2	Вивчити спеціальну літературу і технічну документацію.	18.05.2021 – 20.05.2021	
3	Проаналізувати системи управління базами даних та утиліти для адміністрування.	20.05.2021 – 21.05.2021	
4	Написати розділ 1.	20.05.2021 – 21.05.2021	
5	Провести проектування програмного засобу адміністрування бази даних банківських операцій.	21.05.2021 – 22.05.2021	
6	Написати розділ 2.	21.05.2021 – 22.05.2021	
7	Провести розробку програмного засобу адміністрування бази даних банківських операцій.	23.05.2021 – 25.05.2021	
8	Написати розділ 3.	26.05.2021 – 28.05.2021	
9	Оформити пояснювальну записку.	28.05.2021 – 30.05.2021	
10	Підготувати графічний демонстраційний матеріал та доповідь.	08.06.2021 – 10.06.2021	

7. Дата видачі завдання: "17" травня 2021 р.

Керівник дипломної роботи (проекту) _____ Халімон Н.Ф.

(підпис керівника) (П.І.Б.)

Завдання прийняв до виконання _____ Савченко В.О.

(підпис випускника) (П.І.Б.)

РЕФЕРАТ

Пояснювальна записка до дипломного проекту «Програмний засіб адміністрування»: 54 с., 12 рис., 14 літературних джерел.

АДМІНІСТРУВАННЯ БАЗ ДАНИХ, ПАРАМЕТРИ СИСТЕМНИХ ПОДАНЬ, ФАЙЛИ ПЕРВИННОЇ ГРУПИ, ЖУРНАЛ ТРАНЗАКЦІЙ.

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

Предмет проектування – програмний засіб адміністрування бази даних банківських операцій.

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

Дипломний проект присвячено актуальній тематиці адміністрування баз даних.

ЗМІСТ

ВСТУП 6
РОЗДІЛ 1 АВТОМАТИЗАЦІЯ БАНКІВСЬКОЇ ДІЯЛЬНОСТІ 9
1.1. Огляд існуючих рішень автоматизації 9
1.2. Адміністрування баз даних в банківських установах 12
1.3. Висновки до розділу 14
РОЗДІЛ 2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ АДМІНІСТРУВАННЯ БАЗИ ДАНИХ БАНКІВСЬКИХ ОПЕРАЦІЙ 16
2.1. Засоби адміністрування бази даних банківських операцій 16
2.2. Системні подання відображення параметрів адміністрування бази даних банківських операцій 24
2.3. Опис первинних і зовнішніх ключів 29
2.4. Висновки до розділу 32
РОЗДІЛ 3 РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ АДМІНІСТРУВАННЯ БАЗИ ДАНИХ БАНКІВСЬКИХ ОПЕРАЦІЙ 33
3.1. Склад файлів проекту 33
3.2. Розробка модулів програмного засобу адміністрування бази даних 34
3.3. Висновки до розділу 49
ВИСНОВКИ 51
СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ 53
ДОДАТОК А ..... **Ошибка! Закладка не определена.**

ВСТУП

На сьогоднішній день складно уявити більш сприятливу сферу для запровадження інформаційних технологій, ніж банківська діяльність. Інформаційні технології відіграють важливу роль в процесі автоматизації банківських установ. Дуже велика кількість *IT*-компаній постійно займається розробкою більш сучасних програмних засобів, які б дали можливість швидко обробляти великі обсяги інформації, яка зберігається в базах даних банків. Зумовлено це тим, що розробка таких програм займає великий обсяг часу, сил і знань, відповідно плата за такі послуги буде більшою. Крім того банківські установи мають достатні фінансові можливості для придбання програмних засобів, які можуть значно поліпшити їхню роботу.

Завдання дипломного проекту − розробити програмний засіб адміністрування бази даних банківських операцій, який дасть змогу оброблювати великий обсяг інформації.

Об’єкт проектування − адміністрування баз даних, предметом проектування є програмний засіб адміністрування бази даних банківських операцій.

У першому розділі було розглянуто існуючі рішення автоматизації банківських систем та адміністрування баз даних банківських операцій. На ринку існує багато готових банківських систем. Основним завдання, яке стоїть перед службами автоматизації банків, є вибір найкращого рішення та підтримка ефективності обраної системи. Коли в Україні з’явилась банківська галузь, було приділенно недостатньо уваги питанню автоматизації. Українські банки пішли шляхом створення власних банківських систем. Такий шлях, мав як переваги так і недоліки.

Вкладаючи кошти в програмне забезпечення, комп’ютери, телекомунікаційне обладнання та створення бази для переходу до нових обчислювальних платформ, банківські установи в основному прагнуть зменшити витрати, пришвидшити щоденну роботу та виграти конкуренцію в інших банків.

Банківські системи, як правило, реалізуються на модульній основі. Дуже часто використовуються спеціальні універсальні і потужні комп'ютери, що об'єднані локальною мережею *(LAN)*. Банківські системи використовують віддалений доступ до ресурсів офісу центрального банку для здійснення електронних платежів. Банківська система повинна мати засоби для адаптації до конкретних умов використання. Для підтримки операцій банку система повинна запускати інтернет-оброку транзакцій в режимі реального часу.

Функції банківської системи:

– Автоматизація внутрішньобанківських операцій, їх облік та підготовка підсумкових звітів;

– Автоматизація зв'язку з філіями та відділеннями банку;

– Забезпечення зв'язку з клієнтами;

– Автоматизація роздрібних операцій;

– Автоматизація міжбанківських розрахунків;

– Автоматизація банківської діяльності на ринку цінних паперів;

– Автоматизація процесів швидкого отримання інформації, від якої залежить фінансова ситуація.

В базі даних банківської установи зберігаються дані по всіх операціях, які були проведені даним банком. Зберігатися така база даних повинна на електронних носіях, резервні копії якої за вимогою можуть бути передані в центральний банк.

До інформації яка зберігається в базах даних банку відноситься:

– Інформація про клієнтів;

– Дані про депозити та кредити фізичних і юридичних осіб;

– Інформація про всі банківські операції.

У другому розділі було спроектовано програмний засіб адміністрування бази даних банківських операцій. *Microsoft SQL Server Management Studio* є повнофункціональним інтегрованим адміністративним клієнтом, розробленим для вирішення завдань адміністратора сервера *SQL Server*. У середовищі *Management Studio* завдання адміністрування виконуються за допомогою оглядача об'єктів,

який дозволяє підключитися до будь-якого сервера сімейства *SQL Server* і переглядати його вміст за допомогою графічних засобів.

Мова *SQL (Structured query language)* – це мова програмування, яка застосовується для створення, модифікації та управління даними в реляційній базі даних, керованою відповідною системою управління базами даних.

Для проектування програмного засобу адміністрування бази даних банківських операцій було використано такий перелік системних подань як: *sys.objects, sys.foreign_keys, sys.key_constrains, sys.dm_db_lod_space_usage* і *sys.sysaltfiles*.

У третьому розділі відображено склад файлів проекту та розробку модулів програмного засобу адміністрування бази даних банківських операцій. При розробці програмного засобу адміністрування бази даних банківських операцій було використано середовище *Microsoft Visual Studio* і шаблон додатку *Windows Forms (.NET Framework)*.

Основним завданням дипломного проекту була розробка модулів для адміністрування бази даних. В результаті виконання завдання було розроблено наступні модулі: перегляд розмірів файлів журналів транзакцій бази даних банківських операцій, додавання файлу даних в первинну файлову групу, додавання журналу транзакцій, перегляд параметрів таблиці «Банк», перегляд параметрів таблиці «Коди операцій», перегляд зовнішніх ключів бази даних банківських операцій, перегляд первинних ключів бази даних банківських операцій, перегляд шляхів бази даних банківських операцій. Кожен з цих модулів, допомагає адміністратору бази даних виконувати свою роботу.

Банківські установи завжди прагнули використовувати новітні рішення в галузі інформаційних технологій. Автоматизація роботи банків покращує ефективність роботи підприємців різних рівнів бізнесу. Але існуючі банківські системи пропонують комплексні рішення, які є громіздкими і складними в освоєнні, роботі та проведенні елементарних операцій, тому тема дипломного проекту – «Програмний засіб адміністрування бази даних банківських операцій» є актуальною.

РОЗДІЛ 1
АВТОМАТИЗАЦІЯ БАНКІВСЬКОЇ ДІЯЛЬНОСТІ

Автоматизація – це один із напрямів науково-технічного прогресу, який спрямований на застосування саморегульованих технічних засобів, економіко-математичних методів і систем керування, що звільняють людину від участі у процесах отримання, перетворення, передавання і використання енергії, матеріалів чи інформації, істотно зменшують міру цієї участі чи трудомісткість виконуваних операцій [1].

Банківські установи завжди прагнули використовувати новітні рішення в галузі інформаційних технологій. Автоматизація роботи банків покращує ефективність роботи підприємців різних рівнів бізнесу. Але існуючі банківські системи пропонують комплексні рішення, які є громіздкими і складними в освоєнні, роботі та проведенні елементарних операцій.

1.1. Огляд існуючих рішень автоматизації

Автоматизована Банківська Система – загальновживаний термін, що зазвичай означає комплекс комп'ютерних програм або програмну систему, яка призначена для комплексної автоматизації банківської діяльності, є повноцінним інструментом ведення банківського бізнесу та дозволяє автоматизувати широкий спектр бізнес-процесів і фінансових інструментів банку [2].

Останнім часом банківська система (БС) моєї країни дуже швидко розвивається. Не зважаючи на недоліки нагляду за банківською діяльністю згідно українського законодавства, ситуація поступово покращується. У наш час все

більша кількість банків покладається на професіоналізм працівників та сучасні технології.

Важко уявити собі більш сприятливий ґрунт для запровадження комп'ютерних технологій, ніж банківська діяльність. Майже всі завдання банківському процесі потребують автоматизації. Швидка і безперебійна обробка великих обсягів інформації – одне із головних завдань будь-якої великої фінансової організації. Тому необхідно мати комп'ютерну мережу, яка дозволяє обробляти постійно зростаючий обсяг інформації. Крім того, банківські установи мають достатні фінансові можливості для використання новітніх технологій.

Однак не слід вважати, що звичайні банки готові витрачати великі кошти на комп'ютеризацію. Банк – це перш за все фінансова установа, яка має на меті отримання прибутку, тому вартість модернізації повинна бути співставлена з очікуваною вигодою від її проведення. Відповідно до світової практики, середні витрати на комп'ютеризацію банків становлять щонайменше 17% від загальної суми річних витрат.

Зацікавленість у розвитку комп'ютеризованих банківських систем головним чином визначається стратегічними інтересами. На практиці, інвестиції в такі проекти починають приносити прибуток лише через певний проміжок часу, необхідний для підготовки персоналу та адаптації системи до конкретних умов. Вкладаючи кошти в програмне забезпечення, комп'ютери, телекомунікаційне обладнання та створення бази для переходу до нових обчислювальних платформ, банки в основному прагнуть зменшити витрати, пришвидшити щоденну роботу та виграти конкуренцію.

Нові технології допомагають банкам, інвестиційним та страховим компаніям змінити відносини з клієнтами та знайти нові способи для отримання прибутку. Аналітики сходяться на думці, що нові технології найактивніше впроваджуються інвестиційними компаніями, за ними йдуть банки, і тільки тоді страхові компанії. Завдання, що стоїть перед фінансовими установами, однакове. Це інтеграція нових систем у розподілену архітектуру локальних та глобальних мереж.

На сьогоднішній день комп’ютерна банківська система є найбільш розвиненою серед галузей прикладного мережевого програмного забезпечення. Не можна не відзначити, що банківська система є дуже вигідним ринком для будь-яких виробників комп’ютерів та розробників програмного забезпечення.

Банківська система, на сьогоднішній день, дозволяє автоматизувати майже всю банківську діяльність. На основі використання сучасних мережевих технологій в основній функції банківської системи слід згадати: базу даних на основі «Клієнт-сервер», систему електронної пошти, віддалений доступ до мережевих ресурсів системи, віддалений доступ до мережевих ресурсів системи.

На ринку існує багато готових банківських систем. Основним завдання, яке стоїть перед службами автоматизації банків, є вибір найкращого рішення та підтримка ефективності обраної системи. Коли в Україні з’явилась банківська галузь, було приділенно недостатньо уваги питанню автоматизації. Українські банки пішли шляхом створення власних банківських систем [3]. Цей спосіб має як переваги, так і недоліки.

Переваги створення власної банківської системи:

– Не потрібно вкладувати багато коштів на придбання системи;

– Адаптація системи до умов експлуатації;

– Можливість постійної та ефективної модернізації системи.

Недоліки створення власної банківської системи:

– Необхідно мати багато професіоналів, які будуть підтримувати та модернізувати систему;

– Несумісність систем різних банків;

– Відставання від сучасних тенденцій розвитку.

Сьогодні найпопулярнішими є змішані рішення, де деякі модулі банківської системи розробляються комп'ютерним відділом банку, а інші купуються у незалежних виробників.

Банківські системи, як правило, реалізуються на модульній основі. Дуже часто використовуються спеціальні універсальні і потужні комп’ютери, що об’єднані локальною мережею *(LAN)*. Банківські системи використовують

віддалений доступ до ресурсів офісу центрального банку для здійснення електронних платежів. Банківська система повинна мати засоби для адаптації до конкретних умов використання. Для підтримки операцій банку система повинна запускати інтернет-оброку транзакцій в режимі реального часу.

Функції банківської системи реалізовані у вигляді незалежних модулів:

– Автоматизація внутрішньобанківських операцій, їх облік та підготовку підсумкових звітів;

– Автоматизація зв’язку з філіями та відділеннями банку;

– Забезпечення зв’язку з клієнтами;

– Аналіз усієї банківської діяльності;

– Автоматизація роздрібних операцій;

– Автоматизація міжбанківських розрахунків;

– Автоматизація банківської діяльності на ринку цінних паперів, таких як акції, облігації;

– Автоматизація процесів швидкого отримання інформації, від якої залежить фінансова ситуація.

1.2. Адміністрування баз даних в банківських установах

База даних – це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, великих обсягів [4].

В базі даних банківської установи зберігаються дані по всім операціям, які були проведені даним банком. Наявність такого реєстру даних є обов’язковим для всіх банківських установ. Зберігатися така база даних повинна на електронних носіях, резервні копії якої за вимогою можуть бути передані в центральний банк.

До інформації яка зберігається в базах даних банку відноситься:

– Інформація про клієнтів, а саме паспортні дані, ідентифікаційні коди, відомості про права власності на майно, наявність страхування, кредитів;

– Дані про депозити та кредити фізичних і юридичних осіб;

– Інформація про всі банківські операції, в тому числі міжнародні.

Розміри і час обробки операцій бази даних банку, це лише фактори для сортування.

Також можна відмітити те що бази даних банківських установ не є ізольованими, але доступ до них мають не всі, а тільки співробітники, які мають необхідний рівень акредитації та центральний банк. Звичайний клієнт, має право отримати довідку тільки власного профілю, але за попереднім запитом. Окремі дані, наприклад, за станом платежів, на його рахунок, можуть бути надані безпосередньо при зверненні в банк або через індивідуальний онлайн профіль в додатку або на офіційному сайті цього банку.

Надійність таких баз даних банківських установ ставиться під сумнів, тому що черех їх низький рівень захисту системи, бази даних дуже часто зламують.

Адміністрування баз даних – це функція управління та підтримки програмного забезпечення систем управління базами даних (СУБД) [5]. Основна роль адміністрування баз даних полягає в забезпеченні максимального часу роботи бази даних, щоб вона завжди була доступна при необхідності. Зазвичай це передбачає активний періодичний моніторинг та усунення несправностей. Це в свою чергу тягне за собою деякі технічні навички з боку адміністратора бази даних. На додаток до поглиблених знань про цю базу даних, адміністратор бази даних також потребує знань та навчання на платформі, на якій працює база даних.

Адміністратор бази даних, як правило, також відповідає за інші вторинні, але все ще критично важливі завдання та ролі. Деякі з них включають:

– Безпеку бази даних, тобто забезпечення доступу до бази даних лише уповноваженим користувачам та зміцнення її проти будь-якого зовнішнього, несанкціонованого доступу;

– Налаштування бази даних, тобто налаштування будь-якого з кількох параметрів для оптимізації продуктивності, таких як розподіл пам’яті сервера, фрагментація файлів та використання диска;

– Резервне копіювання та відновлення;

– Створення звітів із запитів.

Підсумовуючи все вище сказане, можна сказати, що функція адміністрування баз даних потребує технічної підготовки та багаторічного досвіду. Деякі компанії, що пропонують продукти комерційної бази даних, такі як *Oracle DB* та *Microsoft SQL Server*, також пропонують сертифікати на свої конкретні продукти. Ці галузеві сертифікати, такі як *Oracle Certified Professional (OCP)* та *Microsoft Certified Administrator Database Administrator (MCDBA)*, проходять довгий шлях до того, щоб переконати організації, що адміністратор бази даних справді проходить повну підготовку щодо відповідного продукту. Оскільки сьогодні більшість продуктів реляційних баз даних використовують мову *SQL*, знання команд і синтаксису *SQL* також є цінним надбанням для сучасних.

1.3. Висновки до розділу

Автоматизована банківська система – призначена для комплексної автоматизації банківської діяльності, є повноцінним інструментом для ведення банківського бізнесу та дозволяє автоматизувати широкий спектр бізнес-процесів і фінансових інструментів банку. Останнім часом банківська система (БС) моєї країни дуже швидко розвивається. У наш час все більша кількість банків покладається на професіоналізм працівників та сучасні технології.

Важко уявити собі більш сприятливий ґрунт для запровадження комп’ютерних технологій, ніж банківська діяльність. Майже всі завдання банківському процесі потребують автоматизації. Швидка і безперебійна обробка великих обсягів інформації – одне із головних завдань будь-якої великої фінансової організації.

Банківські системи, як правило, реалізуються на модульній основі. Дуже часто використовуються спеціальні універсальні і потужні комп’ютери, що об’єднані локальною мережею *(LAN)*.

В базі даних банківської установи зберігаються дані по всім операціям, які були проведені даним банком: а саме інформація про клієнтів, дані про депозити та кредити, інформація про банківські платежі.

Основна роль адміністрування баз даних полягає в забезпеченні максимального часу роботи бази даних, щоб вона завжди була доступна при необхідності. Функція адміністрування баз даних потребує технічної підготовки та багаторічного досвіду.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ АДМІНІСТРУВАННЯ БАЗИ ДАНИХ БАНКІВСЬКИХ ОПЕРАЦІЙ

2.1. Засоби адміністрування бази даних банківських операцій

Microsoft SQL Server Management Studio (рис.2.1) є повнофункціональним інтегрованим адміністративним клієнтом, розробленим для вирішення завдань адміністратора сервера *SQL Server* і бази даних *SQL Azure* [6]. У середовищі *Management Studio* завдання адміністрування виконуються за допомогою оглядача об'єктів, який дозволяє підключитися до будь-якого сервера сімейства *SQL Server* і переглядати його вміст за допомогою графічних засобів. Сервер може бути екземпляром компоненти *Database Engine*, служби *Analysis Services*, служби *Reporting Services*, служби *Integration Services* або бази даних *SQL Azure*.

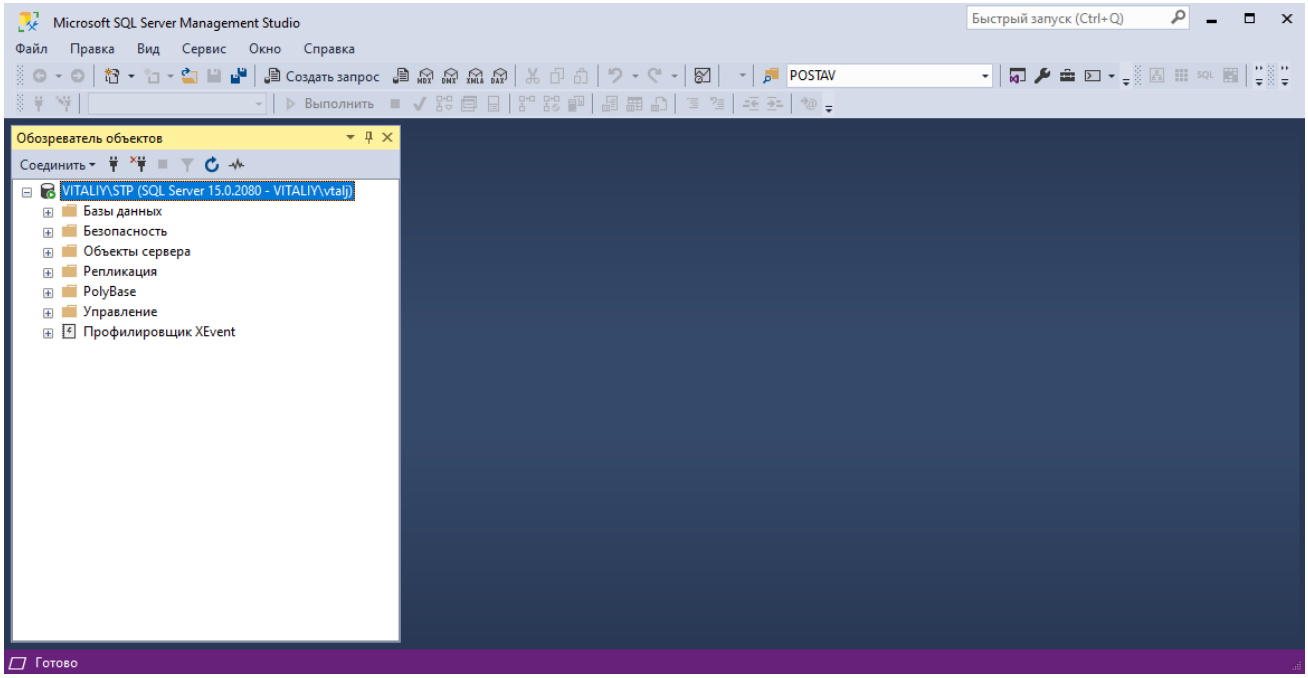


Рис 2.1. Вікно середовища *Microsoft SQL Server Management Studio*

Кафедра КСУ				НАУ 21 17 24 000 ПЗ			
Виконав	Савченко В.О.			Проектування програмного засобу адміністрування бази даних банківських операцій	Літера	Аркуш	Аркушів
Керівник	Халімон Н.Ф.					16	54
Консульт.					СП-436 123		
Норм. контр.	Тупота Є.В.						
Зав. Каф.	Литвиненко О.Є.						

У число засобів середовища *Management Studio* входять зареєстровані сервери, оглядач об'єктів, оглядач рішень, оглядач шаблонів, сторінка зведення і вікно документа. Щоб відобразити засіб, в меню «Вид» виберіть його назву. Для відображення редактора запитів натисніть кнопку «Створити запит» на панелі інструментів.

Середовище *Management Studio* використовується для виконання наступних дій:

– Реєстрація серверів;

– З'єднання з екземпляром компоненти *Database Engine, SSAS,* служб *Reporting Services, Reporting Services* або бази даних *SQL Azure;*

– Налаштування властивостей сервера;

– Управління об'єктами бази даних та службами *SSAS;*

– Створення таких об'єктів, як бази данних, таблиці;

– Управління файлами і групами файлів;

– Приєднання та від'єднання баз даних;

– Запуск середовищ для роботи зі сценаріями;

– Управління безпекою;

– Перегляд системних журналів;

– Контроль поточної активності;

– Налаштування реплікацій;

– Управління повнотекстовими індексами.

Компонент *Database Engine* входить до складу *SQL Server* і є основною службою, призначеною для зберігання, обробки та забезпечення безпеки даних. Також компонент *Database Engine* забезпечує керований доступ до ресурсів і швидку обробку транзакцій, що дозволяє використовувати його навіть з самими вимогливими додатками по обробці даних на підприємстві. *SQL Server* підтримує до 50 екземплярів компонентів *Database Engine* на одному комп'ютері [7].

Службами *Analysis Services* передбачено резервне копіювання і відновлення бази даних і її об'єктів на певний момент часу. Створення резервних копій та

відновлення підходить для перенесення баз даних на модернізовані сервери, між серверами або для розгортання бази даних на робочому сервері. Якщо є цінні дані, але поки немає плану резервного копіювання, то необхідно якомога швидше розробити і реалізувати такий план на випадок подальшого відновлення даних.

Служби *Reporting Services* – включає в себе цілий контейнер серверних і клієнтських додатків для створення, доступу до управління і розгортання матриць, таблиць, графічних звітів, крім того дозволяє створювати звіти в довільній формі. *Reporting Services* – це розширюваний компонент, який можна використовувати для розробки додатків звітів.

Служби *Integration Services* – це платформа для побудови рішень по інтеграції і перетворення даних рівня підприємства. Служби *Integration Services* можна використовувати при вирішенні складних бізнес-задач шляхом копіювання та завантаження файлів, завантаження сховищ даних, очищення та інтелектуального аналізу даних, а також управління об'єктами і даними *SQL Server*.

Мова *SQL* (*Structured query language*) – це мова програмування, яка застосовується для створення, модифікації та управління даними в реляційній базі даних, керованою відповідною системою управління базами даних. Також обов'язково варто відзначити і те, що база даних, і зокрема реляційна модель, заснована на теорії множин, яка має на увазі об'єднання різних об'єктів в одне ціле, під одним цілим в базі даних як раз і мається на увазі таблиця. Це важливо, тому що мова *SQL* працює саме з множиною, з набором даних, тобто з таблицями.

У випадку використання *SQL* людина формулює запит на витяг чи внесення змін до даних, а алгоритм його виконання майже повністю лягає на плечі конкретної систему управління базами даних. Хоча якщо один і той же результат може бути отриманий за допомогою різних запитів, програмісту краще вибрати той, який створить менше навантаження на систему управління базами даних. Тобто програмісту бажано мати уявлення про те, як працюють такі системи.

Запит здійснюється до таблиць бази даних, результатом обробки запиту також є таблиця, яку при бажанні можна зберегти. Мова *SQL* призначена для

створення і зміни реляційних баз даних, а також вилучення з них даних. Іншими словами, *SQL* – це інструмент, за допомогою якого людина керує базою даних. При цьому ключовими операціями є створення таблиць, додавання записів в таблиці, зміна і видалення записів, вибірка записів з таблиць, зміна структури таблиць.

Однак в процесі розвитку мови *SQL* в ньому з'явилися нові засоби. Стало можливо описувати і зберігати такі об'єкти як індекси, подання, тригери і процедури. Тобто в сучасних діалектах *SQL* є елементи процедурних мов. Мову *SQL* і системи управління базами даних зазвичай не використовуються самі по собі, а виконують функцію проміжного вбудованого компонента, що забезпечує зв'язок між прикладним програмним забезпеченням або програмою, яку пише програміст, і базою даних.

З точки зору реалізації мови *SQL* представляє собою набір операторів, які діляться на певні групи і у кожної групи є своє призначення. Групи операторів *SQL* поділяються на:

- *DDL*;
- *DML*;
- *DCL*;
- *TCL*.

Data Definition Language (DDL) – це група операторів визначення даних. Іншими словами, за допомогою операторів, що входять в цю групу, визначається структура бази даних і виконується робота з об'єктами цієї бази, тобто їх створення, зміна і видалення. В цю групу входять наступні оператори:

- *CREATE* – використовується для створення об’єктів бази даних;
- *ALTER* – використовується для зміни об’єктів бази даних;
- *DROP* – використовується для видалення об’єктів бази даних.

Data Manipulation Language (*DML*) - це група операторів для маніпуляції даними. За допомогою цих операторів можна додавати, змінювати, видаляти і вивантажувати дані з бази, тобто маніпулювати ними. В цю групу входять наступні оператори:

– *SELECT* – виконує вибірку даних;

– *INSERT* – додає нові дані*;*

– *UPDATE* – змінює існуючі дані;

– *DELETE* – видаляє дані.

Data Control Language (DCL) – група операторів визначення доступу до даних. Іншими словами, це оператори для управління доступом, за допомогою них можна надавати або обмежувати доступ до виконання певних операцій над об'єктами бази даних. В цю групу входять оператори:

– *GRANT* – надає користувачеві доступ на певні операції з об’єктом;

– *REVOKE* – відкликає виданий доступ;

– *DENY* – обмежує доступ.

Transaction Control Language (TCL) – група операторів для управління транзакціями. Транзакція - це команда або блок команд (інструкцій), які успішно завершуються як єдине ціле, при цьому в базі даних всі внесені зміни фіксуються на постійній основі або скасовуються, тобто всі зміни, внесені будь-якою командою, що входить в транзакцію, будуть скасовані. Група операторів *TCL* призначена для реалізації і керування транзакціями. Сюди можна віднести:

– *BEGIN* – використовується для визначення початку транзакції;

– *COMMIT* – виконує транзакцію;

– *ROLLBACK* – виконує відкат всіх змін, які були зроблені в контексті поточної транзакції;

– *SAVE* – встановлює проміжну точку збереження в транзакції.

Data Manipulation Language (DML) – це сімейство комп'ютерних мов, які використовуються в комп'ютерних програмах або користувачами баз даних для отримання, вставки, видалення або зміни даних в базах даних. Ця мова включає в себе набір операторів для підтримки основних операцій над даними, що містяться в базі. До операцій маніпулювання даними відносяться:

– вставка в базу даних нових відомостей;

– модифікація відомостей, що зберігаються в базі даних;

– витяг відомостей, що містяться в базі даних;

– видалення відомостей з бази даних.

Таким чином, одна з основних функцій СУБД полягає в підтримці мови маніпулювання даними, за допомогою якої користувач може створювати вирази для виконання перерахованих вище операцій з даними.

Поняття маніпулювання даними може бути застосовано як до зовнішнього і концептуального рівнів, так і до внутрішнього рівня. Однак на внутрішньому рівні для цього доводиться визначати дуже складні низькорівневі процедури, що дозволяють виконувати доступ до даних з високою ефективністю. На більш високих рівнях, навпаки, наголос робиться на велику простоту використання, і основні зусилля пов'язані із забезпеченням ефективної взаємодії користувача з системою.

Мови *DML* мають різні базові конструкції вилучення даних. Існують два типи мов *DML*: процедурна і непроцедурна. Основна відмінність між ними полягає в тому, що процедурні мови вказують на те, як можна отримати необхідний результат, тоді як непроцедурні мови описують те, який результат потрібно отримати. Як правило, в процедурних мовах записи розглядаються окремо, тоді як непроцедурні мови оперують з цілими наборами записів.

Частина непроцедурної мови *DML*, яка відповідає за вилучення даних, називається мовою запитів даних. Мову запитів даних можна визначити як високорівневу вузькоспеціалізована мова, призначену для задоволення різних вимог по вибірці інформації з бази даних. У цьому сенсі термін «запит» більше підходить для позначення оператора вилучення даних, вираженого за допомогою мови запитів даних.

Процедурна мова *DML* повідомляє системі про те, які дані необхідні, і точно вказує, як їх можна витягти. За допомогою процедурної мови *DML* користувач, вказує, які дані йому потрібні і як їх можна отримати. Це означає, що користувач повинен визначити всі операції доступу до даних (здійснювані за допомогою виклику відповідних процедур), які повинні бути виконані для отримання необхідної інформації. Зазвичай така процедурна мова *DML* дозволяє витягти

запис, обробити його і, в залежності від отриманих результатів, витягти інший запис, який повинен бути підданий аналогічній обробці, і т.д. Подібний процес вилучення даних триває до тих пір, поки не будуть витягнуті всі дані запиту. Зазвичай оператори процедурної мови *DML* вбудовуються в програму на мові програмування високого рівня, яка містить конструкції для забезпечення циклічної обробки і переходу до інших ділянок коду. Мови *DML* мережевих і ієрархічних СУБД зазвичай є процедурними.

Непроцедурна мова *DML* дозволяє вказати лише те, які дані потрібні, але не те, як їх слід витягувати. Непроцедурні мови *DML* задають весь набір необхідних даних за допомогою одного оператора вибірки або поновлення. За допомогою непроцедурних мов *DML* користувач вказує, які дані йому потрібні, без визначення способу їх отримання. СУБД транслює вираз на мові *DML* в процедуру (або набір процедур), яка забезпечує маніпулювання зазначеним набором записів. Такий підхід звільняє користувача від необхідності знати подробиці внутрішньої реалізації структур даних і особливості алгоритмів, що використовуються для отримання і можливого перетворення даних. В результаті робота користувача стає в певній мірі незалежною відданих.

Непроцедурну мову часто також називають декларативною мовою. Реляційні СУБД в тій чи іншій формі забезпечують підтримку непроцедурної мови маніпулювання даними, а саме структуровану мову запитів *SQL*.

Data Definition Language (DDL) – це сімейство комп'ютерних мов, які використовуються в комп'ютерних програмах для опису структури баз даних. Мова опису даних дозволяє адміністратору баз даних або користувачеві задавати і іменувати сутності й атрибути, які необхідні для роботи деякого додатка, а також зв'язки, наявні між різними сутностями, вказуючи, крім того, обмеження цілісності і захисту.

Схема бази даних складається з набору визначеннь, виражених мовою визначення даних. Мова *DDL* використовується як для визначення новой схеми, так и для модифікації вже існуючої.

Результатом компіляції *DDL*-операторів є набір таблиць, що зберігається в особливих файлах, так званих системних каталогах. У системному каталозі інтегровані метадані, тобто дані, які описують об’єкти бази даних, а також дозволяють спростити спосіб доступу до них і управління ними. Метадані включають визначення записів, елементів даних, а також інших об'єктів, що становлять інтерес для користувачів або необхідних для роботи СУБД. Перед доступом до реальних даних СУБД зазвичай звертається до системного каталогу.

У мові запитів *SQL* значне місце займає оператор *SELECT*, так як за допомогою нього можна повертати рядки та здійснювати вибірку одного або декількох стовпців таблиці в базі даних. Оператор відноситься до групи операторів *DML*.

Докладніші відомості про процес демонструють логічний порядок обробки або порядок прив'язки інструкції *SELECT*. Цей порядок визначає, коли об'єкти, визначені в одному кроці, стають доступними для пропозицій в наступних кроках. Наприклад, якщо обробник запитів можна прив'язати (для доступу) до таблиць або подань, визначеним у пропозиції *FROM*, то ці об'єкти і їх стовпці стають доступними для всіх наступних кроків. Фактично фізичне виконання інструкції визначається обробником запитів і порядок з цього списку може значно відрізнятися:

- *FROM*;
- *ON*;
- *JOIN*;
- *WHERE*;
- *GROUP BY*;
- *WITH CUBE*;
- *HAVING*;
- *SELECT*;
- *DISTINCT*;
- *ORDER BY*.

Простий приклад використання оператору *SELECT*.

*SELECT * FROM Table*

де, * – означає переглянути всі дані, *Table* – таблиця із якої беруться ці дані.

Але на практиці, часто потрібні не всі дані із таблиці, а тільки деякі стовпці, для цього потрібно вказати замість * назву потрібного стовпця або декількох стовпців.

SELECT id, name

FROM Table

де, *id, name* – це колонки із таблиці *Table*.

В процесі вибірки достатньо часто потрібно фільтрувати дані по конкретній умові, тобто не всі дані, а тільки ті які відповідають умові, в конструкції *SELECT* для цього можна використати оператор W*HERE* в поєднанні з оператором *LIKE*.

SELECT id

FROM Table

WHERE id LIKE '111'

В даному випадку було відображено тільки ті рядки, які відповідають нашій умові, а саме коли *id* = 111. Оператор *LIKE* було використано для прикладу, щоб показати синтаксис вибірки, тому що цей оператор було використано при проектуванні програмного засобу адміністрування бази даних банківських операцій.

2.2. Системні подання відображення параметрів адміністрування бази даних банківських операцій

Подання – це віртуальна таблиця, зміст якої обирається з інших таблиць за допомогою виконання запиту, причому у разі зміни значень в цих таблицях дані автоматично змінюються і в поданні [8]. Подання фактично являє собою збережені результати *SQL*-запиту. Подання використовують для фільтрації даних перед доступом користувачів до цих даних.

Подання виконує функцію фільтрації базових таблиць, на які вона посилається. Запит, який визначає подання, може бути ініційований в одній або

кількох таблицях або в інших поданнях поточної або інших баз даних. Крім того, для визначення подання з даними з декількох різних джерел можна використовувати розподілені запити. Це корисно, наприклад, якщо потрібно об'єднати структуровані таким чином дані, що відносяться до різних серверів, кожен з яких зберігає дані конкретного відділу організації.

Подання зазвичай використовуються для направлення, спрощення та налаштування сприйняття кожним користувачем інформації бази даних. Подання можуть використовуватися як механізми безпеки, даючи можливість користувачам отримувати доступ до даних через подання, але не надаючи їм дозволів на безпосередній доступ до базових таблиць, які лежать в основі цих поданнів. Подання можуть використовуватися для забезпечення інтерфейсу зворотної сумісності, що моделює таблицю, яка існує, але схема якої змінилася. Подання можуть також використовуватися при прямому і зворотному копіюванні даних в *SQL Server* для підвищення продуктивності і секціонування даних.

Крім основних визначених користувачем подань, що виконують стандартні ролі, в *SQL Server* передбачені наступні типи подань, які відповідають спеціальним призначенням в базі даних:

- індексовані подання;
- секціоновані подання;
- системні подання.

Індексоване подання – це подання, яке було матеріалізовано. Це означає, що визначення подання було обчислене, а отримані дані збережені так само, як і таблиця. Індексувати подання можна, створивши для нього унікальний кластерний індекс. Індексовані подання можуть значно покращити ефективність деяких типів запитів. Індексовані подання найкраще працюють для запитів, які об’єднують багато рядків. Вони не підходять для базових наборів даних, які часто оновлюються.

Секціоноване подання об’єднує дані з горизонтальним секціонуванням із набору таблиць елементів на одному або декількох серверах. При цьому дані виглядають так, як ніби знаходяться в одній таблиці. Подання, яке об’єднує

таблиці-елементи в одному екземплярі *SQL Server,* називається локальним секціонованим поданням.

SQL Server надає набір системних подань для доступу до метаданих про серверне середовище і його об'єктів бази даних. Існують подання каталогу, подання інформаційної схеми, подання динамічного управління і кілька інших типів подань.

Системні подання розділені на категорії, кожна з яких служить визначеній меті. Найбільша категорія – це та, яка містить подання каталогу. Подання каталогу дозволяють отримувати інформацію про широкий спектр компонентів системи і бази даних, від стовпців таблиць і типів даних до конфігурацій на рівні сервера.

Подання інформаційної схеми схожі на деякі подання каталогу в тому, що вони надають доступ до метаданих, які описують об'єкти бази даних, такі як таблиці, стовпці, домени і перевірочні обмеження. Подання динамічного управління повертають дані про стан сервера, їх можна використовувати для відстеження і точного налаштування примірника *SQL Server* і його баз даних. Як і подання каталогу, динамічні подання управління характерні тільки для *SQL Server*.

В дипломному проекті при проектуванні та розробці було використано:

– системні подання каталогу, такі як *sys.objects, sys.foreign_keys, sys.key_constrains* і *sys.sysaltfiles*;

– системне подання динамічного управління *sys.dm_db_lod_space_usage.*

Всі ці подання знаходяться системній базі даних *master*.

Системне подання *sys.objects* містить по одному рядку для кожного визначеного користувачем об'єкту області схеми, який створюється в базі даних, включаючи скомпільовану в своєму нинішньому вигляді скалярну визначену функцію. Для перегляду параметрів таблиці «Банк» було використано системне подання *sys.objects* і був створений такий запит з даного подання:

Select name, object_id, type, type_desc, create_date

From sys.objects

Where name like ‘%bank%’

В цьому запиті використовуються такі елементи як:

– *name* – тип даних *sysname*, ім’я об’єкту;

– *object_id* – тип даних int, унікальний ідентифікаційний номер об’єкту в базі дани;

– *type* – тип даних *char*(2), тип об’єкту;

– *type_desc* – тип даних *nvarchar*; опис типу об’єкту;

– *create_date* – тип даних *datetime*, дата створення об’єкту.

Оператор *where* за допомогою предикату *like* дозволяє знайти всі значення параметру *name*, де ім’я має таку частину «*bank*».

Для відображення параметрів таблиці «Коди операцій» також було використане подання *sys.objects* і був створений наступний запис:

Select name, object_id, type, type_desc, create_date

From sys.objects

Where name like ‘%grup_oper%’

В цьому запиті використовуються такі параметри як:

– *name* – тип даних *sysname*, ім’я об’єкту;

– *object_id* – тип даних *int*, унікальний ідентифікаційний номер об’єкту в базі дани;

– *type* – тип даних *char*(2), тип об’єкту;

– *type_desc* – тип даних *nvarchar*; опис типу об’єкту;

– *create_date* – тип даних *datetime*, дата створення об’єкту.

За допомогою оператору *where* і предикату *like* було знайдено всі значення параметру *name*, де ім'я мало частину *«grup_oper»*.

Системне подання *sys.foreign_keys* містить в собі по одному рядку для кожного об’єкту, який є обмеженням для зовнішнього ключа. Для перегляду зовнішніх ключів бази даних банківських операцій було використане подання *sys.foreign_keys* та створено новий запит:

Select name, object_id, type, type_desc, create_date

From sys. foreign_keys

В цьому запиті використовуються такі елементи як:

– *name* – тип даних *sysname*, ім’я об’єкту;

– *object_id* – тип даних *int*, унікальний ідентифікаційний номер об’єкту в базі дани;

– *type* – тип даних *char*(2), тип об’єкту;

– *type_desc* – тип даних *nvarchar*; опис типу об’єкту;

– *create_date* – тип даних *datetime*, дата створення об’єкту.

Системне подання *sys.key_constrains* містить в собі по одному рядку для кожного об’єкту, який є обмеженням первинного ключа або обмеженням унікальності. Дане подання було використано для перегляду первинних ключів бази даних банківських операцій і був створений новий запит:

*Select **

From sys.key_constrains

Перелік основних елементів які були використані в цьому запиті:

– *object_id* – тип даних *int*, унікальний ідентифікаційний номер об’єкту в базі дани;

– *type* – тип даних *char*(2), тип об’єкту;

– *type_desc* – тип даних *nvarchar*; опис типу об’єкту;

– *unique_index_id* – тип даних *int*, ідентифікатор реально існуючого унікального індекса в наслідуваному об’єкті.

Системне подання *sys.dm_db_log_space_usage* повертає відомості про використання простору для журналу транзакцій. Це подання було використано для перегляду параметрів журналу транзакцій бази даних банківських операцій. Запит цього подання виглядає так:

*Select **

From from sys.dm_db_log_space_usage

Перелік всіх існуючих елементів, які є в цьому запиті:

– *database_id* – тип даних *smallint*, ідентифікатор бази даних;

– *total_log_size_in_bytes* – тип даних *bigint*, розмір журналу;

– *used_log_space_in_bytes* – тип даних *bigint*, об’єм заповненого журналу;

– *used_log_space_in_percent* – тип даних *real*, об’єм заповненого журналу у відсотках від загального розміру журналу;

– *log_space_in_bytes_since_last_backup* – тип даних *bigint*, об’єм простору, який використовується з моменту створення резервної копії журналу.

Останнє системне подання в проекті це *sys.sysaltfiles*, використовується для перегляду шляхів баз даних банківських операцій і реалізується запитом:

Select fileid, groupid, size, maxsize, growth, dbid, name, filename

From sys.sysaltfiles

Where name like'%bank%'

В даному запиті були використані такі елементи:

– *fileid* – тип даних *smallint*, ідентифікаційний номер файла, для кожної бази він актуальний;

– *groupid* – тип даних *smallint*, ідентифікаційний номер файлової групи;

– *size* – тип даних *int*, розмір файла, в сторінках по 8 кілобайтів;

– *maxsize* – тип даних *in*t, максимальний розмір файла, в сторінках по 8 кілобайтів;

– *growth* – тип даних *int*, граничний розмір бази даних

– *dbid* – тип даних *smallint*, ідентифікаційний номер бази даних, якій належить даний файл;

– *name* – тип даних *sysname*, логічне ім’я файлу;

– *filename* – тип даних *nvarvhar*(260), ім’я фізичного пристрою, яке включає в себе повних шлях до файлу бази даних.

2.3. Опис первинних і зовнішніх ключів

Первинні і зовнішні ключі являють собою два типи обмежень, які можуть використовуватися для забезпечення цілісності даних в таблицях *SQL Server*. Це важливі об'єкти бази даних.

Зазвичай в таблиці є стовпець або поєднання стовпців, що містять значення, унікально визначають кожен рядок таблиці. Цей стовпець, або стовпці,

називаються первинним ключем (*primary key, PK*) таблиці і забезпечують цілісність суті таблиці [9]. Обмеження первинного ключа часто визначаються в стовпці ідентифікаторів, оскільки гарантують унікальність даних.

При заданні обмеження первинного ключа для таблиці компонента *Database Engine* гарантує унікальність даних шляхом автоматичного створення унікального індексу для первинних ключових стовпців. Цей індекс також забезпечує швидкий доступ до даних при використанні первинного ключа в запитах. Якщо обмеження первинного ключа задано більш ніж для одного стовпчика, то значення можуть дублюватися в межах одного стовпчика, але кожне поєднання значень всіх стовпців у визначенні обмеження первинного ключа повинно бути унікальним.

Обмеження первинного ключа [10]:

– В таблиці можлива наявність тільки одного обмеження по первинному ключу;

– Первинний ключ не може включати більше 16 стовпців, а загальна довжина ключа не може перевищувати 900 байт;

– Індекс, що формується обмеженням первинного ключа не може спричинити за собою вихід кількості індексів в таблиці за межі в 999 некластеризованих індексів і 1 кластеризований;

– Якщо для обмеження первинного ключа не вказано, чи є індекс кластеризованим або некластеризованим, то за замовчуванням створюється кластеризований індекс, якщо такий відсутній в таблиці;

– Всі стовпчики з обмеженням первинного ключа повинні бути визначені як ті, що не допускають значення *NULL*. Якщо допустимість значення *NULL* не вказана, то всі стовпці с обмеженням первинного ключа встановлюються як ті, що не допускають значення *NULL*.

Зовнішній ключ (*foreign key, FK*) – це стовпець або поєднання стовпців, яке застосовується для примусового встановлення зв'язку між даними в двох таблицях з метою контролю даних, які можуть зберігатися в таблиці зовнішнього ключа [11]. Якщо один або декілька стовпців, в яких знаходиться первинний ключ для однієї таблиці, згадується в одному або декількох стовпцях іншої таблиці, то в

посиланні зовнішнього ключа створюється зв'язок між двома таблицями. Цей стовпець стає зовнішнім ключем в другій таблиці.

Максимальна кількість таблиць і стовпців, на які може посилатися таблиця в якості зовнішніх ключів (вихідних посилань), дорівнює 253. *SQL Server* збільшує обмеження на кількість інших таблиць і стовпців, які можуть посилатися на стовпці в одній таблиці (вхідні посилання), з 253 до 10 000. Збільшення має наступні обмеження:

– Перевищення 253 посилань на зовнішні ключі підтримується лише за допомогою операцій *DML DELETE*. Операції *UPDATE* і *MERGE* не підтримуються;

– Таблиця з посиланням зовнішнього ключа на саму себе, як і раніше обмежена 253 посиланнями на зовнішні ключі;

– Перевищення 253 посилань на зовнішні ключі в даний час недоступно для індексів columnstore, оптимізованих для пам'яті таблиць, бази даних *Stretch* або секціонованих таблиць зовнішнього ключа.

На відміну від обмежень первинного ключа, при створенні обмеження зовнішнього ключа відповідний індекс автоматично не створюється [12]. Проте, часто виникає необхідність створення індексу для зовнішнього ключа вручну з наступних причин. Стовпці зовнішнього ключа часто використовуються в критеріях з'єднання при спільному застосуванні в запитах даних зі зв'язаних таблиць. Це реалізується шляхом зіставлення стовпця або стовпців в обмеженні зовнішнього ключа в одній таблиці з одним або декількома стовпцями первинного або унікального ключа в іншій таблиці. Індекс дозволяє компоненту *Database Engine* швидко знаходити пов'язані дані в таблиці зовнішніх ключів. В тім, створення індексу не є обов'язковим. Дані з двох зв'язаних таблиць можна комбінувати, навіть якщо між таблицями не визначені обмеження первинного ключа або зовнішнього ключа, але зв'язок по зовнішньому ключу між двома таблицями показує, що ці дві таблиці оптимізовані для спільного застосування в запиті, де ключі використовуються в якості критерії. За допомогою обмежень

зовнішнього ключа в зв'язаних таблицях перевіряються зміни обмежень первинного ключа.

2.4. Висновки до розділу

У даному розділі було спроектовано програмний засіб адміністрування бази даних банківських операцій. Визначено основні засоби адміністрування: середовище *Microsoft SQL Server Management Studio*, мову запитів *SQL* та групи запитів, які відносяться до мови маніпулювання даними *DML* і мови опису даних *DDL*. Для проектування програмного засобу адміністрування бази даних банківських операцій було використано основні параметри компоненти *Database Engine*.

Визначено та описано категорії системних подань, які були використані при проектування програмного засобу адміністрування: системні подання каталогу *sys.objects, sys.key_constrains, sys.sysaltfiles* та системне подання динамічного типу управління *sys.dm_db_log_space_usage*. Всі вони входять до системної бази даних *master*. Системні подання каталогу *sys.objects* було використано для відображення всіх таблиць бази даних банківських операцій, системне подання *sys.key_constrains* – для перегляду всіх зовнішніх ключів, а системне подання *sys.sysaltfiles* – для відображення шляхів баз даних банківських операцій. Системне подання динамічного типу управління *sys.dm_db_log_space_usage* було використано для відображення відомості про використання простору для журналу транзакцій. Також було досліджено та реалізовано використання первинних *(PK)* і зовнішніх ключів *(FK)*.

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНОГО ЗАСОБУ АДМІНІСТРУВАННЯ БАЗИ ДАНИХ БАНКІВСЬКИХ ОПЕРАЦІЙ

3.1. Склад файлів проекту

При розробці програмного засобу адміністрування бази даних банківських операцій було використано середовище *Microsoft Visual Studio* і шаблон додатку *Windows Forms (.NET Framework)*.

Проект програмний засіб адміністрування бази даних банківських операцій містить такі файли:

– *AssemblyInfo.cs* – це файл, який відображає інформацію про створений додаток, а саме основний номер версії, додатковий номер версії, номер збірки і редакцію;

– *Resources.Designer.cs* – це файл, що поміщає елементи, які були додані в проект, через конструктор ресурсів, в папку ресурсів проекту;

– *Settings.Designer.cs* – це файл, який містить в собі автоматично згенерований код, який генерується після того, як будуть внесені зміни до редактора налаштувань *Settings.Settings*;

– *App.config* – це *XML*-файл, що містить будь-яку змінну конфігурацію додатку, це центральне місце для розміщення: рядків підключення до бази даних; деталей підключення до зовнішніх сервісів, налаштувань додатків; інших відомостей про те, як додаток повинен запускатися і розміщуватися;

– *Form1.cs* – це файл коду програмної частини форми; це файл класу віконної форми, в якому записано необхідні методи, функції, а також методи і коди, керовані подіями.

– *Form1.Designer.cs* – це файл дизайнера, в якому ініціалізовано елементи форми. Якщо будь який елемент перетягується у вікно форми, то цей елемент буде автоматично ініціалізованим в цьому класі.

– *Diplom_bankDataSet.xsd* – це файл, який створює схему *XML* за допомогою використання DataSet: його структуру; порядок елементів; правила, яким має відповідати кожен документ. В схемі визначаються: елементи; атрибути, які будуть присутні в документі; типи даних елементів і атрибутів; значення за замовчуванням або фіксовані значення.

– *Program.cs* – перед запуском програми шукає статичний клас, який містить викликаний статичний метод і починає виконання програми в цій області. В додатку *Windows Forms* код створює форму і відкриває її, для того щоб користувач міг використовувати додаток.

3.2. Розробка модулів програмного засобу адміністрування бази даних

Розроблено наступні модулі: перегляд розмірів файлів журналів транзакцій бази даних банківських операцій, додавання файлу даних в первинну файлову групу, додавання журналу транзакцій, перегляд параметрів таблиці «Банк», перегляд параметрів таблиці «Коди операцій», перегляд зовнішніх ключів бази даних банківських операцій, перегляд первинних ключів бази даних банківських операцій, перегляд шляхів бази даних банківських операцій. При розробці модулів програмного засобу адміністрування бази даних банківських операцій було використано простори імен. Програми на *C#* організовані за допомогою просторів імен. Простори імен використовуються як «внутрішні» системи організації для програми, а в якості «зовнішньої» системи організації − способом подання програмних елементів, що надаються іншим програмам.

Код команд *SQL* для адміністрування бази даних банківських операцій реалізовано з модулів в *Visual Studio* на мові *C#*. Компоненти для побудови запитів відображено в конструкторі схем наборів даних (рис. 3.1)

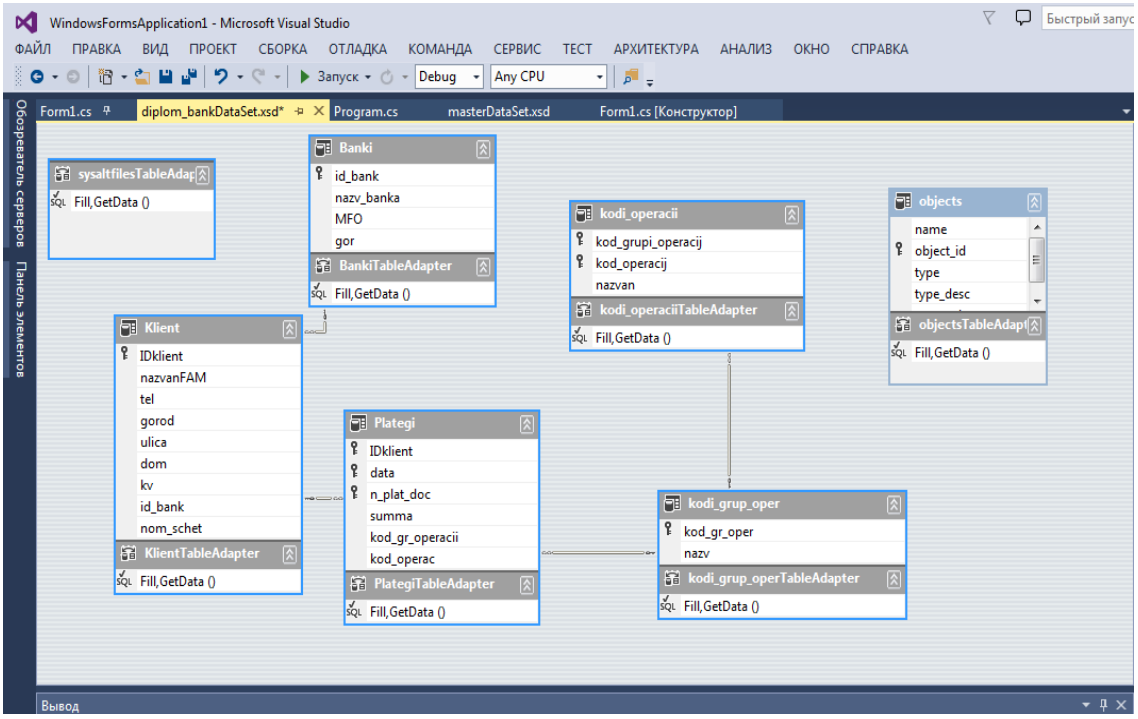


Рис. 3.1. Вікно конструктора схеми наборів даних «Банк»

Ключове слово *using* має три основних застосування:

– Інструкція *using* визначає область, по завершенні якої об'єкт видаляється;

– Директива using створює псевдонім для простору імен або імпортує типи, визначені в інших просторах імен;

– Статична директива *using static* імпортує елементи з одного класу.

Для спрощення використання просторів імен надаються директиви *using*. Перелік просторів імен які були використані при розробці проекту:

– *System.Data* – надає доступ до класів, які представляють архітектуру *ADO.NET*. *ADO.NET* дозволяє створювати компоненти, які ефективно керують даними з декількох джерел даних;

– *System.Drawing* – надає доступ до основних графічних функцій *GDI+*. Простори імен *System.Drawing.Drawing2D*, *System.Drawing.Imaging* і *System.Drawing.Text* забезпечують додаткові функціональні можливості;

– *System.Linq* – надає класи та інтерфейси, що підтримують запити з використанням *LINQ*;

– *System.Text* – містить класи, які представляють кодування *ASCII* і Юникоду; абстрактні базові класи для перетворення блоків знаків в блоки байтів і навпаки; допоміжний клас, який обробляє і форматує об'єкти *String*, не створюючи проміжні екземпляри *String*;

– *System.Threading.Tasks* – надає типи, які спрощують роботу з написання паралельного і асинхронного коду.

– *System.Windows.Forms* – містить класи для створення додатків *Windows*, які дозволяють найбільш ефективно використовувати розширені можливості призначеного для користувача інтерфейсу, доступні в операційній системі *Microsoft Windows*;

– *System.Data.SqlClient* – надає інструкцію *Transact-SQL* або збережену процедуру, яка виконується над базою даних *SQL Server*.

Для програмного засобу адміністрування бази даних банківських операцій в середовищій *Visual Studio* було розроблено головне вікно «Адміністрування первинної файлової групи бази даних банківських операцій» (рис. 3.2). Це вікно має такі пункті меню:

– Таблиці;

– Адміністрування;

– Про автора програми;

– Вихід.

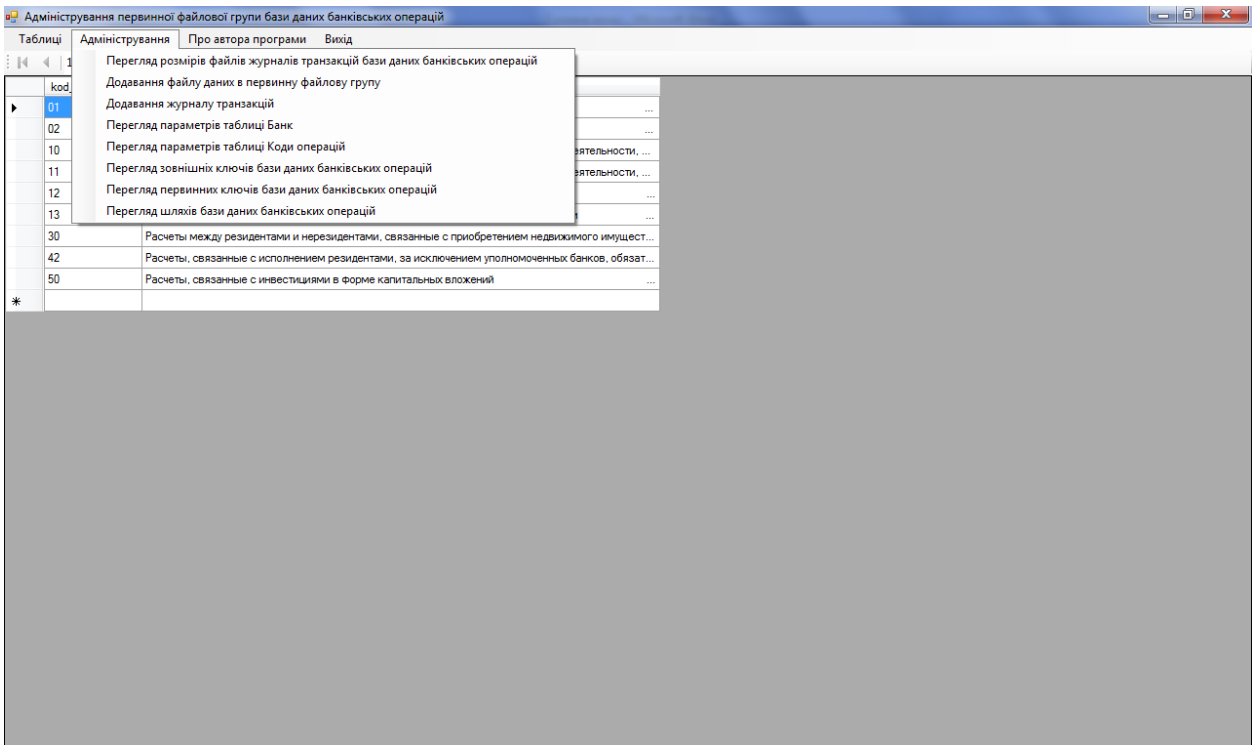


Рис. 3.2. Головне вікно «Адміністрування первинної файлової групи бази даних банківських операцій»

Випадаючий пункт меню «Адміністрування» містить такі підпункти:

– Перегляд розмірів файлів журналів транзакцій бази даних банківських операцій;

– Додавання файлу даних в первинну файлову групу;

– Додавання журналу транзакцій;

– Перегляд параметрів таблиці Банк;

– Перегляд параметрів таблиці Коди операцій;

– Перегляд зовнішніх ключів бази даних банківських операцій;

– Перегляд первинних ключів бази даних банківських операцій;

– Перегляд шляхів бази даних банківських операцій.

Вікно «перегляду таблиці бази даних банківських операцій» зображено на рис. 3.3.

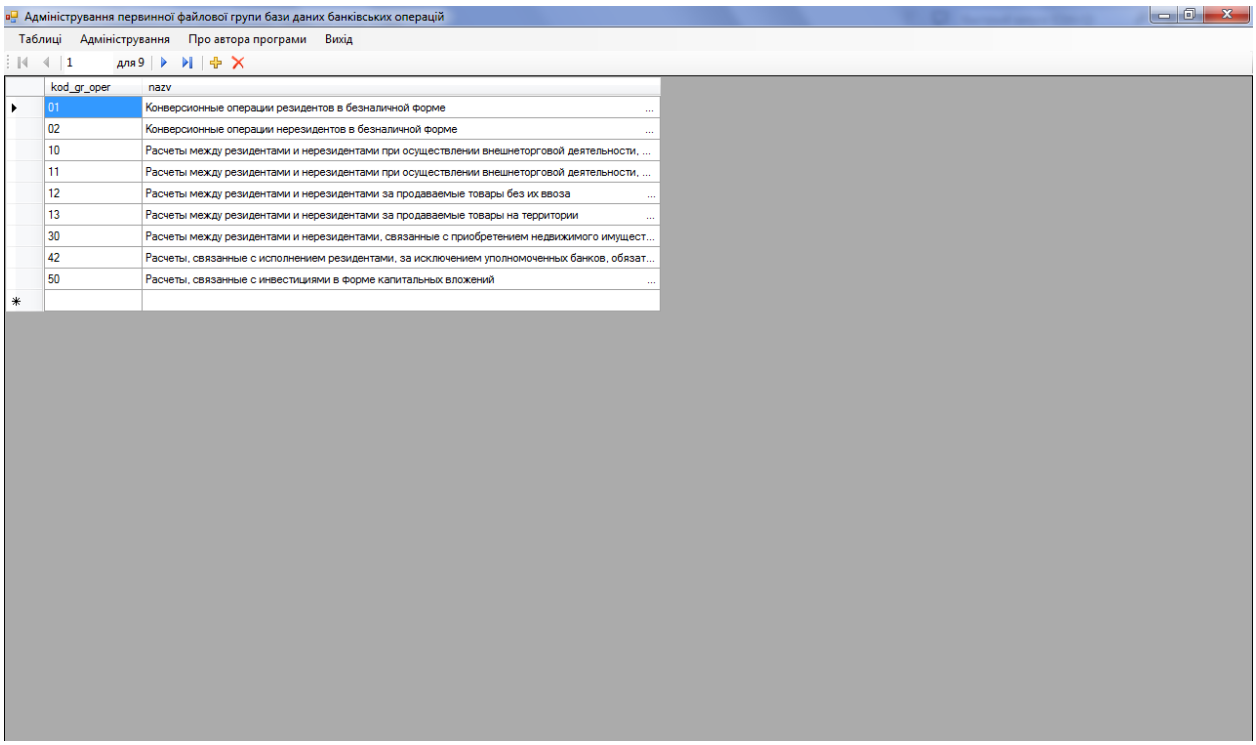


Рис. 3.3. Вікно перегляду таблиці бази даних банківських операцій

3.2.1. Модуль перегляду розмірів файлів журналів транзакцій бази даних банківських операцій

Для модулю перегляду розмірів файлів журналів транзакцій бази даних банківських операцій у оброблювач події *private void toolStripMenuItemProsmLogSpace_Click (object sender, EventArgs e)* в файлі *Form1.cs* було введено код для пункту меню «Перегляд розмірів файлів журналу транзакцій бази даних банківських операцій». При розробці використано наступні компоненти, функції та подання.

Клас *DataSet* – розташований в оперативній пам’яті набір даних, що забезпечує узгоджену реляційну програмну модель незалежно від джерела даних. Набір даних *DataSet* – показує повну сукупність даних, яка включає таблиці, які містять, обмежують і впорядковують дані, а також зв’язки між таблицями.

Клас *DataAdapter* представляє набір команд *SQL* і підключення до бази даних, які використовуються для заповнення *DataSet* і оновлення джерела даних.

Системне подання *dm_db_log_space_usage* – повертає відомості про використання простору для журналу транзакцій бази даних банківських операцій.

Функція *Fill()* заповняє об'єкти класу *DataSet* відомостями про використання простору для журналу транзакцій.

Для компонентів *dataGridViewSysaltfiles, dataGridViewFK, dataGridViewPK* властивість *Visible* встановлено *false*, це зроблено, для того, щоб вони не відображалися. Для компоненти *dataGridViewLog_space_usage* властивість *ReadOnly* встановлено *true*, це зроблено, для того, щоб заборонити зміну значень. Для компоненти *dataGridViewLog_space_usage* властивість *Visible* встановлено *true*, це зроблено, для того, щоб він відображав значення. Для елементу *listBox1* властивість *Visible* встановлена *false*, для того щоб він не відображався. Результат роботи зображено на рис. 3.4.

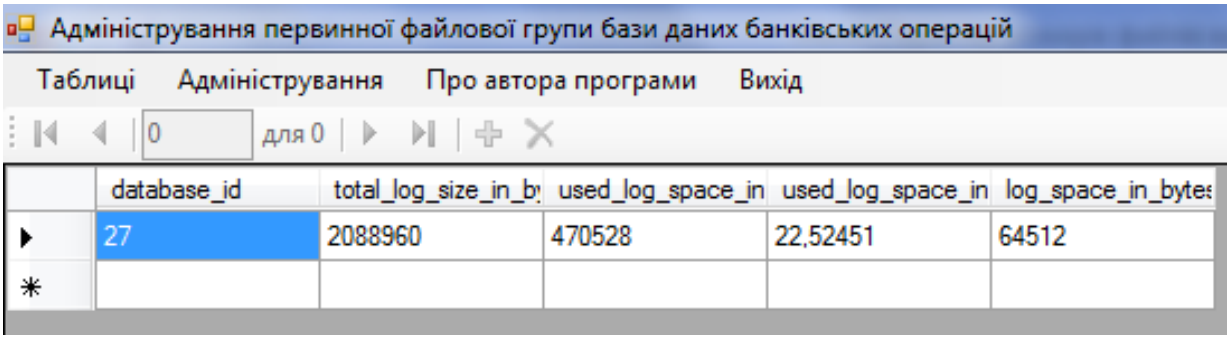


Рис. 3.4. Вікно перегляд розмірів файлів журналів транзакцій бази даних банківських операцій

3.2.2. Модуль додавання файлу в первинну файлову групу

Для модулю додавання файлу в первинну файлову групу у оброблювач події *private void toolStripMenuItemAdd_file_gr_Click(object sender, EventArgs e)* в файлі *Form1.cs* було введено код для пункту меню «додавання файлу даних в первинну файлову групу». При розробці використано наступні компоненти, функції та подання.

Клас *SqlCommand* простору імен *System.Data.SqlClient*, представляє інструкцію *Transact-SQL* або збережену процедуру, яка виконується над базою даних *SQL Server*.

Команда *CommandType* – задає значення, що визначає, як буде інтерпретуватися властивість *CommandText*.

Команда *CommandText* задає інструкцію *Transact-SQL*, складовими якої є: оператор *USE*, який змінює контекст бази даних на вказану системне базу даних *Master*; оператор *ALTER DATABASE*, що змінює параметри конфігурації бази даних *diplom_bank*; оператор *ADD FILE*, що додає файл до бази даних, з такими параметрами, ім'я файлу *«bank_log_2»*, початковий розмір створеного файлу 5 мегабайтів, крок автоматичного збільшення розміру файлу 5 мегабайтів, максимальний розмір файлу 100 мегабайтів (це значення не може бути перевищене), та шлях де буде зберігатися цей файл *«D:\\436_Savchenko\\diplom_bank_2.ndf»*.

Для компонентів *dataGridViewSysaltfiles, dataGridViewLog_space_usage, dataGridViewKodi_Gr_oper, dataGridViewLog_space_usage, dataGridViewObjects, dataGridViewFK* властивість *Visible* встановлена *false*, це зроблено, для того, щоб вони були невидимі. Для компоненту *dataGridViewPerPuti* властивість *Visible* встановлено *true*, це означає те, що вона видима. Для компоненту *dataGridViewSysaltfiles* властивість *ReadOnly* встановлено *true*, це означає що її не можна змінювати.

Для елементу *listBox1* властивість *Visible* встановлено *true*, це означає те що компонент видимий. Результат роботи зображено на рис. 3.5. У вікні пункту меню відображений елемент *listBox1*, в ньому знаходиться повідомлення про те, що відбулось додавання файлу даних в первинну файлову групу та шлях, де цей файл збережений.

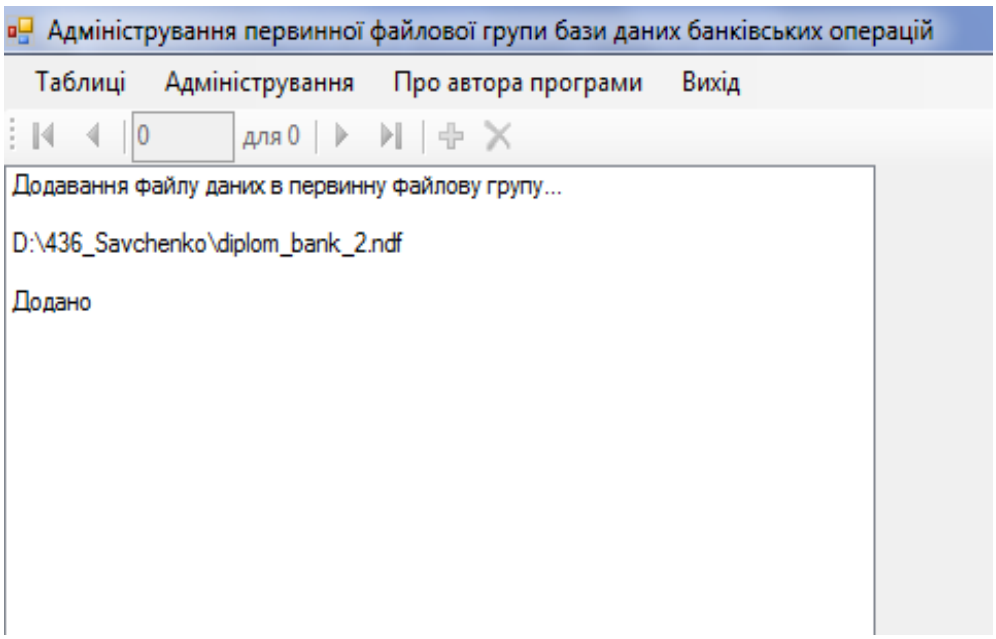


Рис. 3.5. Вікно перегляд додавання файлу в первинну файлову групу

3.2.3. Модулі перегляду параметрів таблиці «Коди операцій» і таблиці «Банк»

Для модулю перегляду параметрів таблиці «Коди операцій» у оброблювач події *private void ToolStripMenuItemKodi_grup_oper_Click (object sender, EventArgs e)* в файлі *Form1.cs* було введено код для пункту меню «перегляд параметрів таблиці Коди операцій». При розробці використано наступні компоненти, функції та подання.

Компонент *BindingSource* призначений для спрощення процесу прив'язки елементів управління до джерела даних. Компонент *BindingSource* виступає в якості джерела і каналу передачі даних для прив'язки інших елементів управління. Він реалізує абстракцію підключення даних форми, перенаправляючи команди до базового списку даних. Крім того, можна додавати дані безпосередньо в нього, тому сам компонент виступає в якості джерела даних.

Функція *Update()* змінює існуючі дані в таблиці або поданні в *SQL Server*.

Функція *Fill()* заповняє об’єкт *DataSet* відомостями про використання простору для перегляду файлів даних таблиці коди операцій.

Компонент *TableAdapter* заповнює набір даних, даними з бази даних на основі одного або декількох зазначених запитів або збережених процедур. *TableAdapter* також може виконувати операції додавання, оновлення та видалення в базі даних для збереження змін, внесених в набір даних.

Компонент *TableAdapter* – адаптер таблиць забезпечує взаємодію між додатком і базою даних. Він підключається до бази даних, виконує запити або збережені процедури і повертає нову таблицю даних або заповнює існуючі *DataTable* за допомогою повернутих даних. Також адаптер таблиць може передавати оновлені дані з додатку назад в базу даних.

Для елементу *listBox1* та компонентів *dataGridViewObjects, dataGridViewSysaltfiles, dataGridViewLog_space_usage, dataGridViewPerPuti, dataGridViewPK, dataGridViewFK* властивість *Visible* встановлено *false*, це зроблено, для того, щоб вони були невидимі. Для компоненту *dataGridViewKodi_Gr_oper* властивість *Visible* встановлено *true*, це зроблено, для того щоб компонента була видима.

Результат роботи зображено на рис. 3.6. У вікні пункту меню відображений компонент *dataGridViewKodi_Gr_oper*, в ньому міститься інформація про параметри таблиці «Коди операцій», а саме ім’я, ідентифікатор об’єкту, тип, опис типу (*FOREIGN_KEY_CONSTRAINT* – обмеження зовнішнього ключа, *USER_TABLE* – таблицю користувачів та *PRIMARY_KEY_CONSTRAINT* – обмеження первинного ключа) і дата створення таблиці.

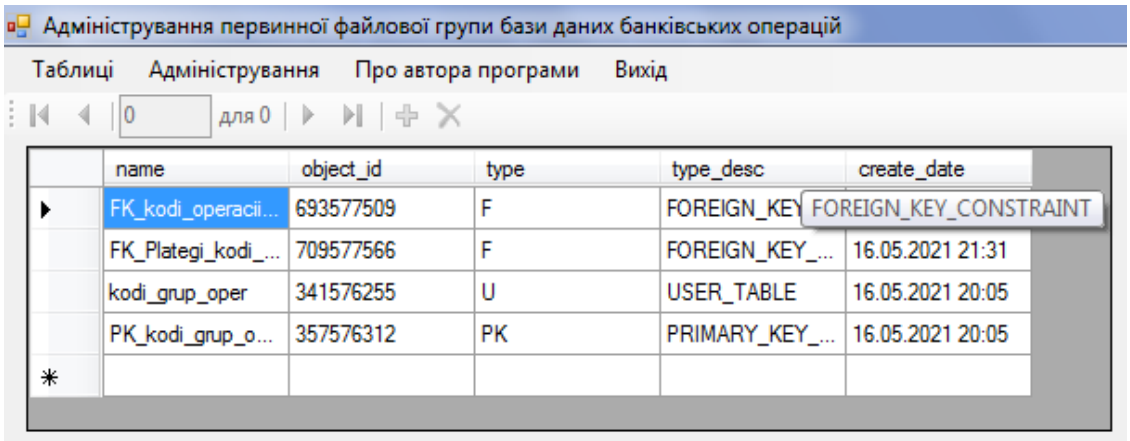


Рис. 3.6. Вікно перегляд параметрів таблиці «Коди операцій»

Для модулю перегляд параметрів таблиці «Банк» у оброблювач події *private void переглядПараметрівТаблиціBankToolStripMenuItem_Click (object sender, EventArgs e)* в файлі *Form1.cs* було введено код для пункту меню «Перегляд параметрів таблиці банк». При розробці використано наступні компоненти, функції та подання.

Клас *DataSet* – розташований в оперативній пам'яті набір даних, що забезпечує узгоджену реляційну програмну модель незалежно від джерела даних. Набір даних *DataSet* – показує повну сукупність даних, яка включає таблиці, які містять, обмежують і впорядковують дані, а також зв'язки між таблицями.

Клас *DataAdapter* представляє набір даних *SQL* і підключення до бази даних, які використовуються для заповнення *DataSet* і оновлення бази даних *Sql Server*.

Функція *Fill()* – додає або оновлює рядки в *DataSet* для отримання відповідності рядків в джерелі даних.

Для компонентів *dataGridViewSysaltfiles, dataGridViewKodi_Gr_oper, dataGridViewLog_space_usage, dataGridViewFK, dataGridViewPK* властивість *Visible* встановлено *false*, це реалізовано для того, щоб вони не відображалися на формі. Для компоненту *dataGridViewObjects* властивості *Visible* і *ReadOnly* встановлені *true*, це зроблено, для того щоб вона була видима і заборонити зміну компоненти.

Результат роботи зображено на рис. 3.7. У вікні пункту меню відображений компонент *dataGridViewObjects*, в ньому міститься інформація про параметри таблиці «Банк», а саме ім'я, ідентифікатор об'єкту, тип, опис типу (*FOREIGN_KEY_CONSTRAINT* – обмеження зовнішнього ключа, *USER_TABLE* – таблицю користувачів та *PRIMARY_KEY_CONSTRAINT* – обмеження первинного ключа) і дата створення таблиці.

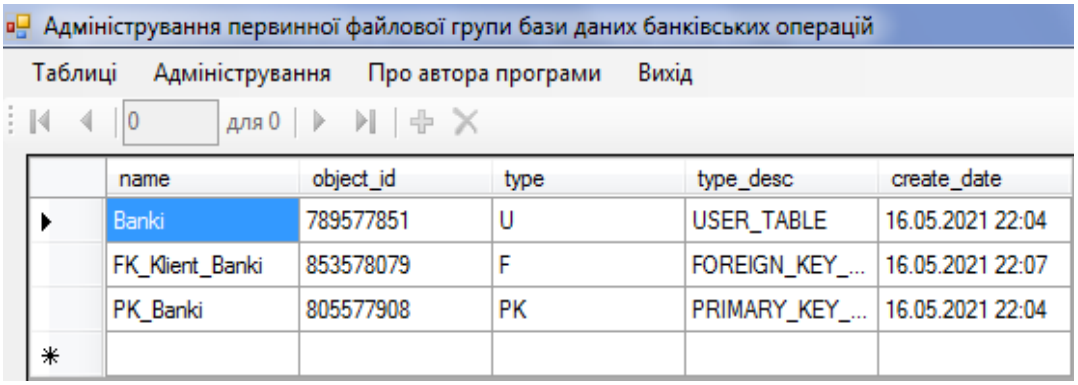


Рис. 3.7. Вікно перегляд параметрів таблиці «Банк»

3.2.4. Модулі перегляду ключів бази даних банківських операцій

Для модулю перегляд зовнішніх ключів бази даних банківських операцій у оброблювач події *private void Per_FK_ToolStripMenuItem_Click(object sender, EventArgs e)* в файлі *Form1.cs* було введено код для пункту меню «Перегляд зовнішніх ключів бази даних банківських операцій». При розробці використано наступні компоненти, функції та подання.

Клас *DataSet* – розташований в оперативній пам'яті набір даних, що забезпечує узгоджену реляційну програмну модель незалежно від джерела даних. Набір даних *DataSet* – показує повну сукупність даних, яка включає таблиці, які містять, обмежують і впорядковують дані, а також зв'язки між таблицями.

Клас *DataAdapter* представляє набір даних *SQL* і підключення до бази даних, які використовуються для заповнення *DataSet* і оновлення джерела даних.

Функція *Fill()* – додає або оновлює рядки в *DataSet* для отримання відповідності рядків в джерелі даних.

Для компонентів *dataGridViewSysaltfiles, dataGridViewKodi_Gr_oper, dataGridViewLog_space_usage, dataGridViewObjects, dataGridViewPK* властивість *Visible* встановлено *false*, це зроблено, для того, щоб вони були невидимі. Для компоненту *dataGridViewFK* властивості *Visible* і *ReadOnly* встановлені *true*, це зроблено, для того щоб він був видимий і його не можна було змінювати.

Результат роботи зображено на рис. 3.8. У вікні пункту меню відображений компонент *dataGridViewFK*, в ньому міститься інформація про те які є зовнішні ключі в базі даних банківських операцій, їх ідентифікатори, тип та опис типу (*FOREIGN_KEY_CONSTRAINT* – обмеження зовнішнього ключа).

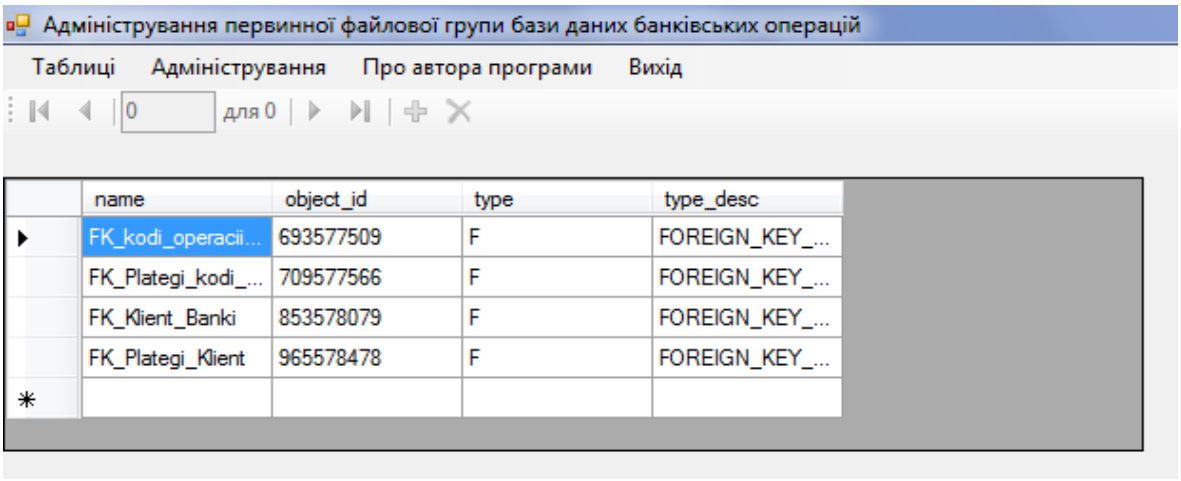


Рис. 3.8. Вікно перегляду зовнішніх ключів бази даних банківських операцій

Для модулю перегляду первинних ключів бази даних банківських операцій у оброблювач події *private void Per_PK_ToolStripMenuItem_Click(object sender, EventArgs e)* в файлі *Form1.cs* було введено код для пункту меню «Перегляд первинних ключів бази даних банківських операцій». При розробці використано наступні компоненти, функції та подання.

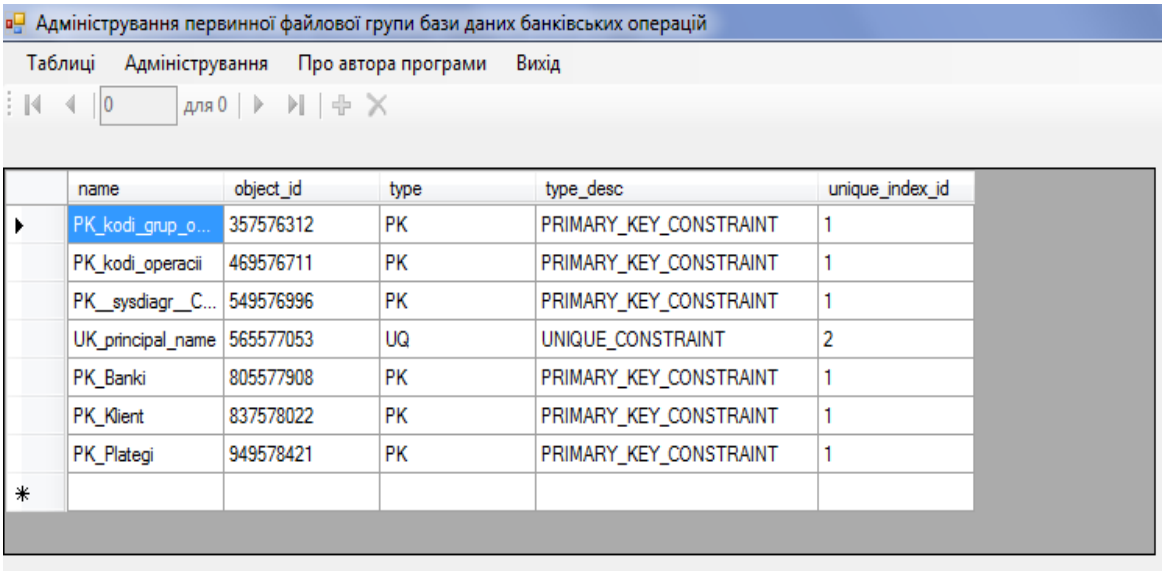
Клас *DataSet* – розташований в оперативній пам’яті набір даних, що забезпечує узгоджену реляційну програмну модель незалежно від джерела даних. Набір даних *DataSet* – показує повну сукупність даних, яка включає таблиці, які містять, обмежують і впорядковують дані, а також зв’язки між таблицями.

Клас *DataAdapter* представляє набір даних *SQL* і підключення до бази даних, які використовуються для заповнення *DataSet* і оновлення джерела даних.

Функція *Fill()* – додає або оновлює рядки в *DataSet* для отримання відповідності рядків в джерелі даних.

Для компонентів *dataGridViewSysaltfiles*, *dataGridViewLog_space_usage*, *dataGridViewKodi_Gr_oper*, *dataGridViewPerPuti*, *dataGridViewFK* та елементу *listBox1* властивість *Visible* було встановлено *false*, для того, щоб вони були невидимі. Для компоненту *dataGridViewPK* властивість *Visible* та *ReadOnly* було встановлено *true*, для того щоб він відображався та його не можна було редагувати.

Результат роботи зображено на рис. 3.9. У вікні пункту меню відображений компонент *dataGridViewPK*, в ньому міститься інформація про те які є первинні ключі в базі даних банківських операцій, їх ідентифікатори, тип, опис типу (*PRIMARY_KEY_CONSTRAINT* – обмеження первинного ключа, *UNIQUE_CONSTRAINT* – унікальне обмеження) та ідентифікатор реально існуючого унікального індекса в наслідуваному об’єкті.



name	object_id	type	type_desc	unique_index_id
PK_kodi_grup_o...	357576312	PK	PRIMARY_KEY_CONSTRAINT	1
PK_kodi_operacii	469576711	PK	PRIMARY_KEY_CONSTRAINT	1
PK__sysdiagr__C...	549576996	PK	PRIMARY_KEY_CONSTRAINT	1
UK_principal_name	565577053	UQ	UNIQUE_CONSTRAINT	2
PK_Banki	805577908	PK	PRIMARY_KEY_CONSTRAINT	1
PK_Klient	837578022	PK	PRIMARY_KEY_CONSTRAINT	1
PK_Plategi	949578421	PK	PRIMARY_KEY_CONSTRAINT	1

Рис. 3.9. Вікно перегляду первинних ключів бази даних банківських операцій

3.2.5. Модуль додавання журналу транзакцій

Для розробки модулю додавання журналу транзакцій в оброблювач подій *private void toolStripMenuItemAdd_flog_Click(object sender, EventArgs e)* в файлі *Form1.cs*, було введено код для пункту меню «додавання журналу транзакцій». При розробці використано наступні компоненти, функції та подання.

Клас *SqlCommand* належить простору імен *System.Data.SqlClient* і представляє інструкцію *Transact-SQL*, або збережену процедуру, яка виконується над базою даних *SQL Server.*

CommandType – це команда, що задає значення, що визначає, як буде інтерпретуватися властивість *CommandText*.

CommandText – це команда яка задає інструкцію *Transact-SQL*, складовими якої є: оператор *USE*, який змінює контекст бази даних на вказану системне базу даних *Master*; оператор *ALTER DATABASE*, що змінює параметри конфігурації бази даних *diplom_bank*; оператор *ADD LOG FILE*, що додає файл до бази даних, з такими параметрами, ім'я файлу *«bank_log3»*, початковий розмір створеного файлу 5 мегабайтів, крок автоматичного збільшення розміру файлу 5 мегабайтів, максимальний розмір файлу 100 мегабайтів (це значення не може бути перевищене), та шлях де буде зберігатися цей файл *«D:\\436_Savchenko\\bank_log3.ldf»*.

Клас *SqlDataReader* простору імен *System.Data.SqlClient*, надає спосіб читання потоку рядків послідовного доступу з бази даних *SQL Server*. *ExecuteReader()* – це метод, що відправляє команду *CommandText* в *Connection* і створює *SqlDataReader*. *Connection* – команда, яка створює з'єднання з джерелом даних.

Функція *SqlConnectionMaster.Open()* – відкриває підключення до системної бази даних *Master*.

Функція *SqlConnectionMaster.Close()* – закриває підключення до системної бази даних *Master*.

Для компонентів *dataGridViewSysaltfiles, dataGridViewLog_space_usage, dataGridViewPerPuti* властивість *Visible* встановлена *false*, це реалізовано для того, щоб ці компоненти були невидимі. Для елементу *listBox1* властивість *Visible* встановлена *true*, це реалізовано, для того елемент був видимий.

Результат роботи зображено на рис. 3.10. У вікні пункту меню відображений елемент *listBox1*, в ньому знаходиться повідомлення про те, що відбулось додавання журналу транзакцій та шлях, де цей файл збережений.

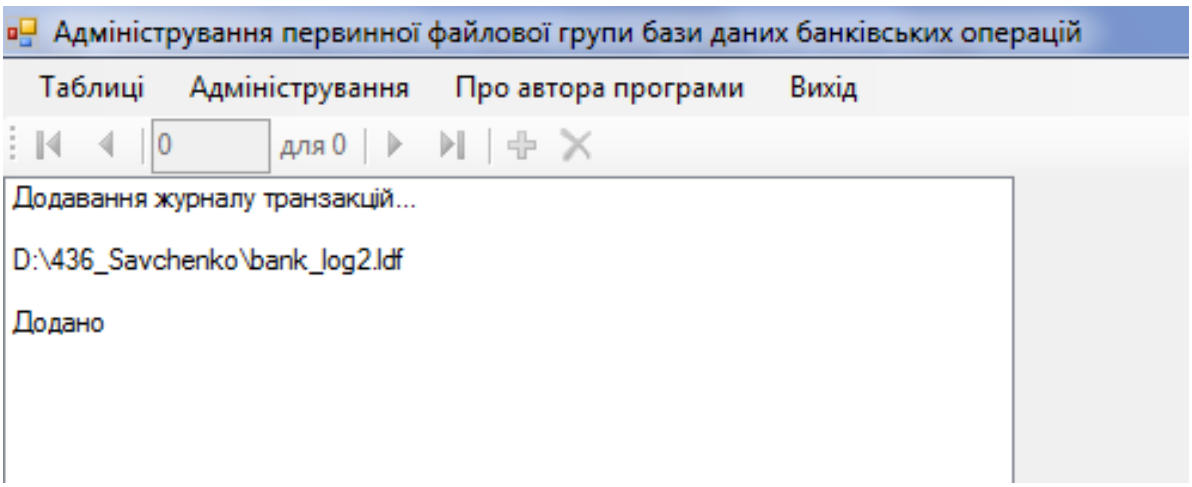


Рис. 3.10. Вікно додавання журналу транзакцій

Для модулю перегляду шляхів бази даних банківських операцій у оброблювач події *private void PerPutiToolStripMenuItem_Click(object sender, EventArgs e)* в файлі *Form1.cs*, було введено код для пункту меню «Перегляд шляхів бази даних банківських операцій». При розробці використано наступні компоненти, функції та подання.

Клас *DataAdapter* представляє набір даних *SQL* і підключення до бази даних, які використовуються для заповнення *DataSet* і оновлення джерела даних.

Клас *DataAdapter* контролює взаємодію з існуючими джерелами даних. *DataAdapter* використаний для заповнення таблиць у наборі даних і отримання даних із джерела даних.

Для компонентів *dataGridViewLog_space_usage, dataGridViewKodi_Gr_oper, dataGridViewLog_space_usage, dataGridViewObjects, dataGridViewFK* та для елементу *listBox1*, властивість *Visible* було встановлено *false*, для того, щоб вони були невидимі.

Функція *Fill()* – додає рядки в об’єкт *diplom_bankDataSet* для отримання відповідності рядків в джерелі даних. Було виконано такий запит: *select, fileid, groupid, size, maxsize, growth, dbid, name, filename from sys.sysaltfiles where name like '%bank%'*. Результат роботи запиту зображено на рис. 3.11.

На цій панелі відображено:

– *fileid* – ідентифікаційний номер, який унікальний для кожної бази даних;

– *groupid* – ідентифікаційний номер файлової групи;

– *size* – розмір файлу, в сторінках по 8 кілобайтів;

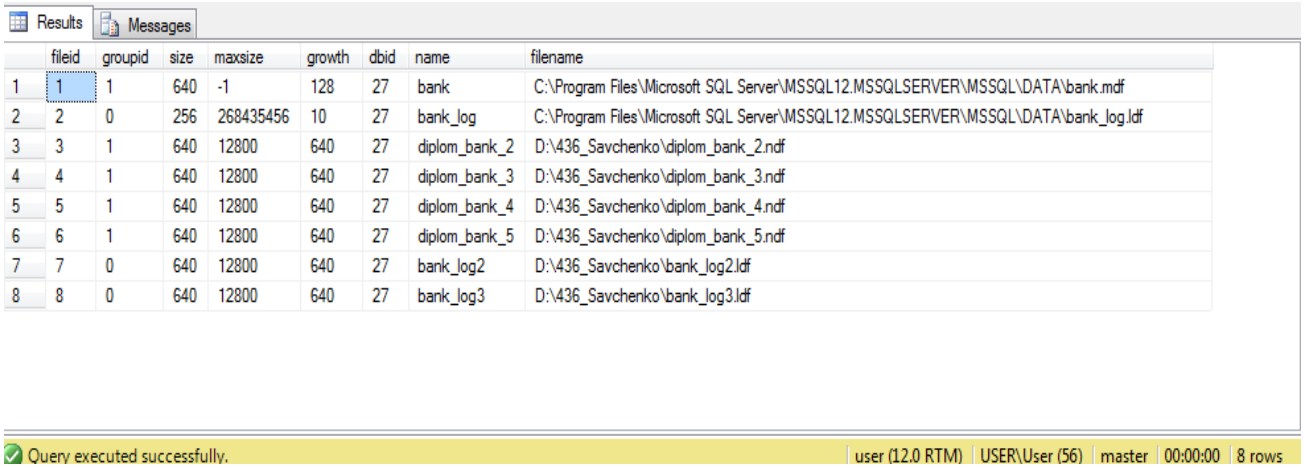
– *maxsize* – максимально можливий розмір файлу, в сторінках по 8 кілобайтів (для першої бази даних стоїть значення «-1», яке означає що розмір файлу може збільшуватись до повного заповнення диску);

– *growth* – максимальний розмір бази даних;

– *dbid* – ідентифікаційний номер бази даних, якій належить даний файл;

– *name* – логічне ім'я файла;

– *file name* – ім'я фізичного пристрою, яке включає повний шлях до файлу.


Results Messages

	fileid	groupid	size	maxsize	growth	dbid	name	filename
1	1	1	640	-1	128	27	bank	C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\bank.mdf
2	2	0	256	268435456	10	27	bank_log	C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\bank_log.ldf
3	3	1	640	12800	640	27	diplom_bank_2	D:\436_Savchenko\diplom_bank_2.ndf
4	4	1	640	12800	640	27	diplom_bank_3	D:\436_Savchenko\diplom_bank_3.ndf
5	5	1	640	12800	640	27	diplom_bank_4	D:\436_Savchenko\diplom_bank_4.ndf
6	6	1	640	12800	640	27	diplom_bank_5	D:\436_Savchenko\diplom_bank_5.ndf
7	7	0	640	12800	640	27	bank_log2	D:\436_Savchenko\bank_log2.ldf
8	8	0	640	12800	640	27	bank_log3	D:\436_Savchenko\bank_log3.ldf

Рис. 3.11. Панель результатів перегляду шляхів бази даних банківських операцій

3.3. Висновки до розділу

В цьому розділі було описано склад файлів розробленого проекту *AssemblyInfo.cs, Resources.Designers.cs, Settings.Designers.cs, App.config, Form1.cs, Form1.Designers, Diplom_bankDataSet.xsd* та *Program.cs*.

При розробці програмного засобу адміністрування бази даних банківських операцій було використано середовище *Microsoft Visual Studio* і шаблон додатку *Windows Forms (.NET Framework)*.

Також в даному розділі було описано розробку модулів програмного засобу адміністрування бази даних банківських операцій, а саме: модуль перегляду розмірів файлів журналів транзакцій бази даних банківських операцій, модуль додавання файлу в первинну файлову групу, модуль перегляду параметрів таблиці «Банк», модуль перегляду параметрів таблиці «Коди операцій», модуль перегляду зовнішніх ключів бази даних банківських операцій, модуль перегляду первинних ключів бази даних банківських операцій, модуль додавання журналу транзакцій. Головним завданням розроблених модулів являється адміністрування бази даних банківських операцій.

ВИСНОВКИ

Під час виконання дипломного проекту було проаналізовано такі теми: автоматизація банківських установ, банківські системи, адміністрування бази даних банківських операцій.

Останнім часом банківська система моєї країни дуже швидко розвивається. У наш час все більша кількість банків покладається на професіоналізм працівників та сучасні технології.

Важко уявити собі більш сприятливий ґрунт для запровадження комп'ютерних технологій, ніж банківська діяльність. Майже всі завдання банківському процесі потребують автоматизації. Швидка і безперебійна обробка великих обсягів інформації – одне із головних завдань будь-якої великої фінансової організації.

Банківські системи, як правило, реалізуються на модульній основі. Дуже часто використовуються спеціальні універсальні і потужні комп'ютери, що об'єднані локальною мережею *(LAN)*.

В базі даних банківської установи зберігаються дані по всім операціям, які були проведені даним банком: а саме інформація про клієнтів, дані про депозити та кредити, інформація про банківські платежі.

Основна роль адміністрування баз даних полягає в забезпеченні максимального часу роботи бази даних, щоб вона завжди була доступна при необхідності. Функція адміністрування баз даних потребує технічної підготовки та багаторічного досвіду.

У дипломному проекті було спроектовано програмний засіб адміністрування бази даних банківських операцій. Визначено основні засоби адміністрування: середовище *Microsoft SQL Server Management Studio*, мову запитів *SQL* та групи запитів, які відносяться до мови маніпулювання даними *DML* і мови опису даних *DDL*.

Визначено та описано категорії системних подань, які були використані при проектування програмного засобу адміністрування: системні подання каталогу

sys.objects, sys.key_constrains, sys.sysaltfiles та системне подання динамічного типу управління *sys.dm_db_log_space_usage*. Всі вони входять до системної бази даних *master*. Системні подання каталогу *sys.objects* було використано для відображення всіх таблиць бази даних банківських операцій, системне подання *sys.key_constrains* – для перегляду всіх зовнішніх ключів, а системне подання *sys.sysaltfiles* – для відображення шляхів баз даних банківських операцій. Системне подання динамічного типу управління *sys.dm_db_log_space_usage* було використано для відображення відомості про використання простору для журналу транзакцій. Також було досліджено та реалізовано використання первинних *(PK)* і зовнішніх ключів *(FK)*.

У дипломному проекті було описано склад файлів розробленого програмного засобу *AssemblyInfo.cs, Resources.Designers.cs, Settings.Designers.cs, App.config, Form1.cs, Form1.Designers, Diplom_bankDataSet.xsd* та *Program.cs*.

При розробці програмного засобу адміністрування бази даних банківських операцій було використано середовище *Microsoft Visual Studio* і шаблон додатку *Windows Forms (.NET Framework)*.

Також в дипломному проекті було описано розробку модулів програмного засобу адміністрування бази даних банківських операцій, а саме: модуль перегляду розмірів файлів журналів транзакцій бази даних банківських операцій, модуль додавання файлу в первинну файлову групу, модуль перегляду параметрів таблиці «Банк», модуль перегляду параметрів таблиці «Коди операцій», модуль перегляду зовнішніх ключів бази даних банківських операцій, модуль перегляду первинних ключів бази даних банківських операцій, модуль додавання журналу транзакцій. Головним завданням розроблених модулів є адміністрування бази даних банківських операцій.

Дипломний проект було виконано з дотриманням діючих стандартів та положень [13], [14].

СПИСОК БІБЛІОГРАФІЧНИХ ПОСИЛАНЬ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Автоматизація [Електронний ресурс] – режим доступу: *https://ru.wikipedia.org/wiki/Автоматизация* (дата звернення 17.05.2021)
2. Автоматизована банківська система [Електронний ресурс] – режим доступу: *https://ru.wikipedia.org/wiki/Автоматизированная_банковская_система* (дата звернення 17.05.2021)
3. Банківська енциклопедія / Під ред. проф. А. М. Мороза – К.: Ельтра, 1993. – 328с.
4. Коннолли Т. Базы данных. Проектирование, реализация и сопровождение. Бегг К. / Теория и практика – 3-е изд. – М.: «Вильямс», 2003. – 436с.
5. Култыгин, О. П. Администрирование баз данных. СУБД *MS SQL Server* учеб. пособие / О. П. Култыгин. - Москва : МФПА, 2012. - 232с.
6. *Microsoft SQL Server Management Studio* [Електронний ресурс] – режим доступу: *https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15* (дата звернення 19.05.2021)
7. *Database Engine* [Електронний ресурс] – режим доступу: *https://docs.microsoft.com/ru-ru/sql/database-engine/configure-windows/manage-the-database-engine-services?view=sql-server-ver1*5 (дата звернення 19.05.2021)
8. Подання [Електронний ресурс] – режим доступу: *https://metanit.com/sql/sqlserver/10.1.php* (дата звернення 20.05.2021)
9. Первинний ключ [Електронний ресурс] – режим доступу: *https://www.ibm.com/docs/en/ida/9.1.1?topic=entities-primary-foreign-keys* (дата звернення 21.05.2021)
10. Обмеження первинного ключа [Електронний ресурс] – режим доступу: *https://docs.microsoft.com/ru-ru/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver15* (дата звернення 21.05.2021)

11. Зовнішній ключ [Електронний ресурс] – режим доступу: *https://www.ibm.com/docs/en/ida/9.1.1?topic=entities-primary-foreign-keys* (дата звернення 21.05.2021)
12. Обмеження зовнішнього ключа [Електронний ресурс] – режим доступу: *https://docs.microsoft.com/ru-ru/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver15* (дата звернення 21.05.2021)
13. ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення / Держстандарт України. – Вид. офіц. – [Чинний від 1995-02-23]. – Київ, 2007. – 86с.
14. Слободян О. Положення про дипломні роботи (проекти) випускників Національного авіаційного університету. – К.: Видавництво НАУ, 2017. – 63с.